

WEB-SERVICES – Praktische Aufgaben

- (1) LZ WSDL - PA POIMANAGERSERVICE
- (2) LZ BPEL – PA WSORG-BPELPRO

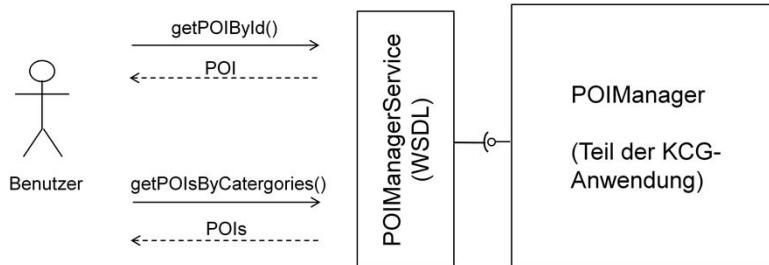
(1) In der Praktischen Aufgabe "POIMANAGERSERVICE" soll ein Web-Service mit der Entwicklungsumgebung Eclipse entwickelt werden. Ziel ist es, die verschiedenen Technologien und Standards, die zur Entwicklung von Web-Services beitragen, kennenzulernen und einen lauffähigen Web-Service zu erstellen.

(2) In dieser Praktischen Aufgabe (PA) ist der einer Workshop-Organisation (WSORG) auf dem KIT Campus zugrundeliegende Prozess durch einen BPEL-Prozess (BPELPRO) zu unterstützen. In dieser praktischen Aufgabe werden zwei Web-Services mit einander komponiert und verknüpft. Zur Beschreibung der Servicekomposition wird die XML-basierte Sprache Business Process Execution Language (BPEL) [OASIS-BPEL-V2] benutzt.

LZ WSDL – PA POIMANAGERSERVICE

- (1) Es soll unter Nutzung der Eclipse Web Tools Platform (WTP) ein Web-Service "POIManagerService" entwickelt werden, der bestehende Funktionalität der Komponente "POIManager" in Form von zwei Operationen bereitstellt

- (1) getPOIById()
- (2) getPOIsByCategoryId



2

21.04.2012

WASA - 3-2 WEB-SERVICE-KERNSTANDARDS

Cooperation & Management (C&M, Prof. Abeck)
Institut für Telematik, Fakultät für Informatik

In der Praktischen Aufgabe "POIMANAGERSERVICE" soll ein Web-Service mit der Entwicklungsumgebung Eclipse entwickelt werden. Ziel ist es, die verschiedenen Technologien und Standards, die zur Entwicklung von Web-Services beitragen, kennenzulernen und einen lauffähigen Web-Service zu erstellen.

- (1) Der zu entwickelnden Web-Service stellt zwei Operationen bereit, um POIs (PointOfInterest) abfragen zu können.
 - (1.1) Suche nach einer POI mit einer bestimmten ID
 - (2.1) Abfrage aller POIs, die zu der als Parameter übergebenen Kategorie gehören. Eine Kategorie ist z.B. Menschen, Cafeterien oder Bibliotheken auf dem Campus und wird in Form eines String übergeben.

(1.1) Web-Services können die Rolle eines Dienstnehmers oder eines Dienstgebers als auch beide Rollen spielen. In dieser praktischen Aufgabe spielt der Web-Service die Rolle eines Dienstgebers und definiert eine Schnittstelle nach außen.

(1.2) Ziel ist es, bestehende Funktionalität zur Verwaltung von Point of Interests (POI) durch Operationen bereitzustellen und bestehende Funktionalität der Komponente POIManager bereitzustellen und eine Wiederverwendung zu ermöglichen.

(1.3) Um die bereitgestellte Dienstfunktionalität aufrufen zu können, soll der Web-Service durch eine eindeutige Adresse in Form eines Uniform Ressource Identifier (URI) identifiziert werden.

POI	PointOfInterest
WTP	Web Tools Platform (Eclipse)

PA POIMANAGERSERVICE

Entwickeln von Web-Services unter Eclipse

- (1) Die Entwicklung von Web-Services wird von Eclipse durch die Web Tools Platform (WTP) unterstützt
 - (1) Enthaltene Werkzeuge sind u.a. Web-Services-Framework, WSDL-Editor, WS-I-Validator
 - (2) Wird ergänzt um JEE Standard Tools zur Entwicklung von Java-basierten Web-Services

- (1) Das Eclipse-WTP-Projekt erweitert die Eclipse-Plattform um Werkzeuge zur Entwicklung von Web- und JEE-Anwendungen.
 - (1.1) WS-I ist ein Standard zur Überprüfung bzw. Sicherstellung der Interoperabilität von Web-Services.
 - (1.2) Hierzu gehört ein erweiterbarer Web-Service-Wizard zur Erzeugung und Nutzung von Web-Services und Wizard-Erweiterungen zur Nutzung der Apache-Axis-Laufzeitumgebung

TODO

Wird eine Einführung in WTP benötigt?

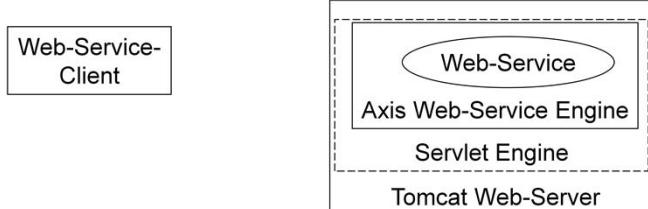
Was sollte diese beinhalten? (muss sehr kompakt auf einer Folie zusammengefasst werden)

JEE	Java Enterprise Edition
JST	JEE Standard Tools
WSDL	Web Service Description Language
WSIL	Web Service Inspection Language
WS-I	Web Services Interoperability
WST	Web Service Tools

PA POIMANAGERSERVICE

Installation von Apache Axis2

- (1) Download und Installation des Apache Tomcat Servers
 - (1) Kann direkt aus Eclipse erfolgen
- (2) Download und Installation der Apache Axis2 Web-Service-Engine
 - (1) Durch Importieren der entsprechenden WAR-Datei (Web ARchive)
- (3) Deployment von Axis2 auf Apache Tomcat
- (4) Überprüfen der Installation und des Deployments



4

21.04.2012

WASA - 3-2 WEB-SERVICE-KERNSTANDARDS

Cooperation & Management (C&M, Prof. Abeck)
Institut für Telematik, Fakultät für Informatik

TODO: Was muss zusätzlich zu der bereits in der Kurseinheit KOMPONENTEN beschriebenen Eclipse-Installation für diese PA installiert werden?

(1) Apache Tomcat ist ein (Open Source) Web-Server, der außerdem in der Lage ist, Java Servlets auszuführen (sog. "Java Servlet Container" [C&TD-EWTP:5]).

Der Java Servlet Container "Apache Tomcat" kann direkt über Eclipse heruntergeladen und installiert werden. Zum initialen Download und Installieren ist es notwendig, Eclipse als lokaler Administrator auszuführen.

- Zur Installation ist dann in die "Java EE"-Perspektive zu wechseln ("Window -> Open Perspective -> Other ... -> Java EE").

- Neue Server können in der View "Servers" angelegt werden (im unteren Frame zu finden). Über einen Rechtsklick kann in dieser Sicht das Kontextmenü aufgerufen werden. Der Befehl lautet "New -> Server".

- Hier ist "Apache -> Tomcat 6.0" zu wählen.

- Über "Browse..." muss dann das Installationsverzeichnis ausgewählt werden ("Tomcat installation directory").

- Der Tomcat Server sollte in den Ordner "C:\Programme\Apache Tomcat" installiert werden. Mit einem Klick auf "Download and Install..." werden die benötigten Dateien heruntergeladen und installiert.

- Mit "Finish" wird der Installationsprozess abgeschlossen.

- Eclipse kann dann geschlossen werden und nochmals als eingeschränkter Benutzer geöffnet werden. Nun ist der oben beschriebene Prozess nochmals zu durchlaufen. Es wird wieder das gleiche Installationsverzeichnis wie für den Administrator Account angeben, jedoch erkennt Eclipse die schon installierten Dateien und es ist nicht notwendig, auf "Download and Install..." zu klicken, sondern mit „Finish“ ist die Installation zu beenden.

(2) Die Axis2 WAR-Distribution ist unter der URL http://ws.apache.org/axis2/download/1_4_1/download.cgi zum Download erhältlich und muss dann noch entpackt werden. Die entpackte Axis2 WAR-Datei kann nun als Projekt in Eclipse importiert werden. Hierzu im "Project Explorer" über das Kontextmenü den Befehl "Import..." auswählen und dann "Web -> WAR file".

(3) Das importierte Axis2-Projekt kann auf Tomcat bereitgestellt (deploy) werden, indem in der Servers-Sicht (unterer Rahmen) auf dem Tomcat-Server rechts geklickt wird und der Befehl "Add and Remove Projects..." ausgewählt wird. Die verfügbaren Projekte werden im linken Teil gelistet, die bereits bereitgestellten (deployed) Projekte im rechten Teil. Das Axis2-Projekt ist links zu markieren und mit einem Klick auf "Add" auf dem Server bereitzustellen.

(4) Der Tomcat Server ist nun mittels Rechtsklick in der „Servers“-View und Klick auf „Start“ zu starten. In der View „Console“ werden Meldungen zum erfolgreichen oder fehlgeschlagenen Start des Servers angezeigt. Ist der Server erfolgreich gestartet, sollte unter der URL <http://localhost:8080/axis2/axis2-web/HappyAxis.jsp> die „Axis2 Happiness Page“ angezeigt werden, die die Installation von Axis2 validiert. Alternativer Aufruf: <http://localhost:8080/axis2/> -> „Validate“

WAR

Web Archive

PA POIMANAGERSERVICE – Top-Down-Ansatz ("Contract First")

- (1) Der Web-Service soll nach dem Top-Down-Ansatz sog. "Contract First" erstellt werden
 - (1) Benötigte Datentypen mit XML Schema Definition (XSD) beschreiben
 - (2) Die Schnittstelle des Web-Services durch eine WSDL-Datei beschreiben und erstellte Datentypen benutzen
 - (3) Mit dem Apache Axis2 Framework den Quellcode generieren und anpassen

(1) Für die Entwicklung von Web-Services stehen die zwei Ansätze "Code First" und "Contract First" zur Verfügung. Bei dem "Code First"-Ansatz handelt es sich um ein "Bottom Up"-Verfahren, das ausgehend vom Quellcode z.B. Java den "Contract" in Form einer WSDL-Datei erstellt. Der "Contract First"-Ansatz hingegen generiert aus der WSDL-Datei, die der Benutzer zum Beschreiben der Operationen erstellt hat, den Quellcode.

(1.1) Im ersten Schritt sollen die Datentypen mit XML Schema Definition (XSD) [W3C-XSD] und in Form von XSD-Dateien beschrieben werden. Die XSD-Dateien können von anderen XSD-Dateien importiert werden, um andere Datentypen zu referenzieren.

(1.2) Als zweiter Schritt ist eine WSDL-Datei mit Operationen und Datentypen anzulegen. Die WSDL-Datei definiert alle Operationen, die ein Web-Service bereitstellt, wobei Nachrichtentypen für die Eingabe- und Ausgabeparameter aus den vorher erstellen XSD-Dateien importiert werden.

(1.3) Im letzten Schritt soll mit dem Apache Axis2 Framework [Apache-Axis2] einen Quellcode generiert und mit der Funktionalität der XMLPOIMangers aus den vorherigen praktischen Aufgaben angepasst werden.

XSD XML Schema Definition

[Apache-Axis2] Apache, Apache Axis2, <http://axis.apache.org/axis2/java/core/index.html>

[W3C-XSD] W3C, XML Schema Definition, www.w3.org/XML/Schema

PA POIMANAGERSERVICE – Vorgehensbeschreibung

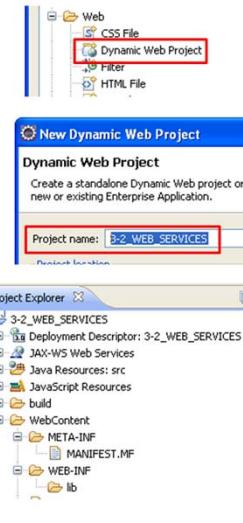
- (1) Es soll ein passendes Eclipse-Projekt für die Aufgabenstellung erstellt werden und die geeigneten Eclipse-Tools benutzt werden
- (2) Die Datentypen sollen mit XML Schema (XSD) beschrieben und in einzelne Dateien gespeichert werden
- (3) Die Schnittstelle des Web-Services soll mit der Web Services Description Language (WSDL) in Eclipse erstellt werden, wobei
 - (1) erstellte und existierende Datentypen in XML Schema importiert werden sollen
 - (2) die von außen zugängliche Operationen sowie ihre Parameter und Rückgabewerten definiert werden sollen
 - (3) das Protokoll zum Datenaustausch, der Endpunkt und die Adresse zum Erreichen des Web-Services festgelegt werden sollen

- (1) Für die Web-Service-Entwicklung bietet Eclipse eine Reihe von Tools an, die einfaches Erstellen und Editieren der Artefakte ermöglichen. Das Web Tool Project (WTP) [Eclipse-WTP] von Eclipse enthält eine Reihe von graphischen Editoren für verschiedene Programmiersprachen und erweitert die Eclipse-Plattform mit Tools und Werkzeuge, um J2EE- und Webanwendungen zu entwickeln.
- (2) Die im Nachrichtenaustausch benutzten Datentypen sollen in XML Schema beschrieben werden und in einzelne Dateien gespeichert werden, um sie von anderen Web-Services wiederverwenden zu können.
- (3) Zur Erstellung und Beschreibung der Dienstschnittstelle sollen die Web Services Description Language (WSDL) und die Entwicklungsumgebung Eclipse benutzt werden.
 - (3.1) Die vorher erstellten Datentypen, die in der Schnittstellenbeschreibung benutzt werden, sollen in die WSDL-Definition importiert werden.
 - (3.2) Mit Hilfe des WSDL-Editors in Eclipse lassen sich Operationen zur WSDL-Datei einfach hinzufügen sowie mit neuen als auch existierenden Datentypen festlegen.
 - (3.3) Informationen zum Erreichen des Web-Services und zum Datenaustausch sind entsprechend in der WSDL-Datei einzustellen.

[Eclipse-WTP] Eclipse Web Tool Project (WTP), <http://www.eclipse.org/webtools/>, 2011

PA POIMANAGERSERVICE – Web Project in Eclipse

- (1) Erstellen eines neuen Eclipse-Projekts
 - (1) Über den Menüpunkt "File"->"New"->"Other"->"Web"->"Dynamic Web Project" ein Projekt erstellen
 - (2) Als Name des Projektes "3-2_POIMANAGERSERVICE" eingeben und die Struktur des Projekts im Project Explorer ansehen



7

21.04.2012

WASA - 3-2 WEB-SERVICE-KERNSTANDARDS

Cooperation & Management (C&M, Prof. Abeck)
Institut für Telematik, Fakultät für Informatik

- (1) Die Entwicklung des Web-Services soll in einem separaten Eclipse-Projekt erfolgen. Eclipse bietet eine Reihe von Arten von Projekten an, um Web Anwendungen zu entwickeln. Dynamic Web Project ist eine Art von Projekten für Webanwendungen, die zusätzlich zu statischen HTML-Seiten auch dynamische J2EE-Inhalte und Ressourcen wie z.B. Servlets, JSP-Dateien und Filters enthalten können.
 - (1.1) In Eclipse über den Menüpunkt "New"->"Other"->"Web"->"Dynamic Web Project" ein neues Projekt anlegen. Der Arbeitsverzeichnis "workspace" von BPEL sollte in einem Pfad wie z.B. "C:\workspace" liegen, der keine Leerzeichen enthält. Der Axis2 Code-Generator stürzt ab, wenn der Pfad Leerzeichen enthält und verhindert eine korrekte Generierung der Software-Artefakte.
 - (1.2) Das Projekt hat nach der Konvention "<Kurseinheitennummer>_<PA-Kurzbezeichnung>" den Namen "3-2_POIMANAGERSERVICE". Nach erfolgreichem Anlegen des Projekts enthält das Projekt Java-Klassen und einen Ordner "WebContent", wo die XSDs und WSDL-Schnittstellen gespeichert werden sollen. Die Struktur eines dynamischen Web-Projekts sieht folgendermaßen aus:

(Web Deployment Descriptor) ist der standardisierte Deployment Descriptor (web.xml) für die Web-Anwendung.

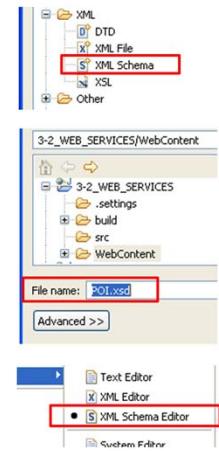
(Java Resources) enthält den Java-Quellcode für Klassen, Java-Beans und Servlets. Diese Ressourcen werden automatisch kompiliert und die generierten Dateien werden in den Ordner "WEB-INF/classes" kopiert, wenn sie zum Projekt hinzugefügt sind.

(WebContent) ist ein Ordner, der HTML, JSP und graphische Dateien beinhaltet. Dieser Ordner repräsentiert den Inhalt der WAR-Datei, die auf einen Server deployed wird.

(WEB-INF) dieser Ordner enthält die Klassen, Bibliotheken und eine web.xml Datei. Der Unterordner "classes" ist für Servlets, Utility-Klassen und enthält die vom Compiler generierten Klassen. Der Unterordner "lib" beinhaltet die von der Webanwendung referenzierten Klassen und Bibliotheken in Form von JAR-Dateien.

PA POIMANAGERSERVICE – XML Schema mit Eclipse

- (1) Datentypen mit Eclipse erstellen
 - (1) Unter "File"->"New"->"XML" das "XML Schema" auswählen
 - (2) Im Ordner "WebContent" eine neue Datei mit dem Namen "POI.xsd" anlegen
 - (3) Die Datei "POI.xsd" mit dem XML Schema-Editor öffnen



8

21.04.2012

WASA - 3-2 WEB-SERVICE-KERNSTANDARDS

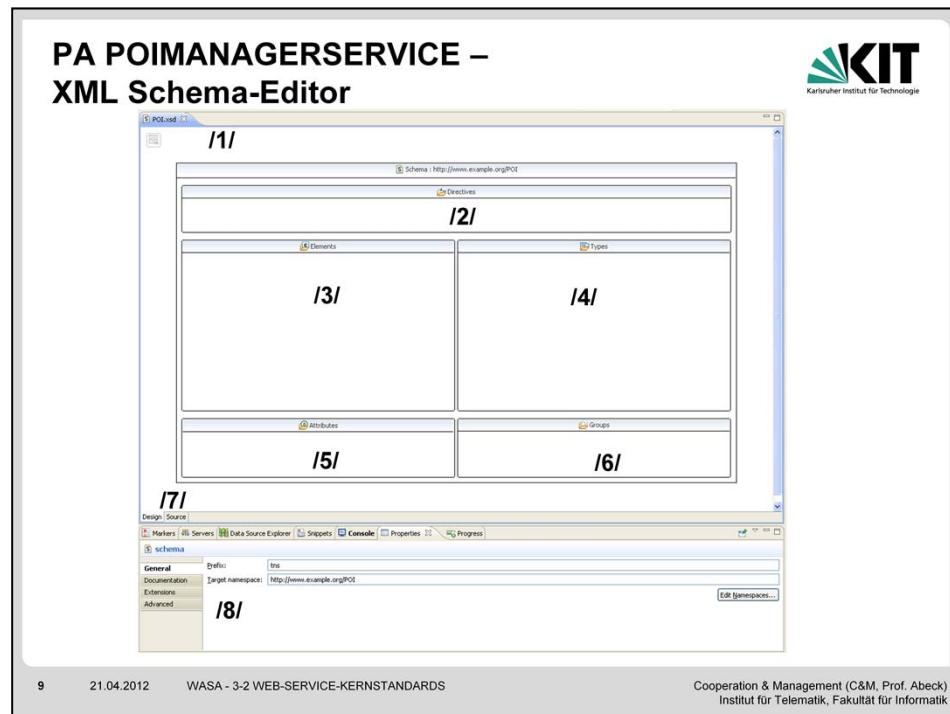
Cooperation & Management (C&M, Prof. Abeck)
Institut für Telematik, Fakultät für Informatik

XML Schema stellt eine formale Beschreibung zur Definition der Struktur und Validierungsbeschränkungen von XML-Dokumenten dar. WSDL ist XML-basiert, d.h. es besteht eine XSD (XML Schema Description), durch die der Aufbau einer WSDL-Beschreibung festgelegt ist.

- (1) XML Schemas sind ein Teil von Web-Service-Schnittstellen, die eine detaillierte Struktur der Nachrichten beschreiben. Um Datentypen in WSDL nutzen zu können, werden im aktuellen Web Project XSD-Dateien erstellt, die später importiert werden sollen. Der Vorteil der Erstellung von eigenständigen Dateien besteht in der Wiederverwendung und dem Referenzieren aus anderen XML-Dokumenten.
 - (1.1) Die Dateien sind vom Typ "XML Schema", der im Menü "File"->"New"->"XML" zu finden ist. XML Schema beschreibt in einer komplexen Schemasprache Datentypen, einzelne XML-Schema-Instanzen (Dokumente) und Gruppen solcher Instanzen. Ein konkretes XML Schema wird auch als eine XSD (XML Schema Definition) bezeichnet und hat als Datei üblicherweise die Endung .xsd" [Wikipedia-XSD].
 - (1.2) Im Ordner "WebContent" legen wir eine neue Datei mit dem Namen "POI.xsd" an. Diese Datei beinhaltet den Datentyp POI in XSD-Format.
 - (1.3) Die Datei kann mit dem "XML Schema Editor" durch doppelten Mausklick oder über den Menüpunkt "XML Schema Editor" im Kontextmenü geöffnet werden.

[Wikipedia-XSD] Wikipedia, XML Schema, http://de.wikipedia.org/wiki/XML_Schema, 2011

PA POIMANAGERSERVICE – XML Schema-Editor

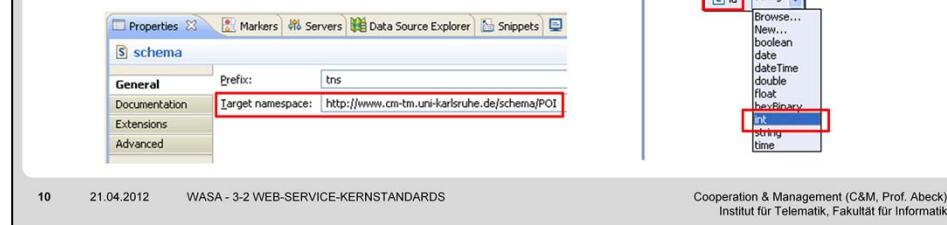


Nach dem doppelten Mausklick auf eine XSD-Datei in Eclipse, wird die Datei im XML Schema-Editor geöffnet. Die wichtigsten Elemente im XML Schema-Editor sind hier kurz eingeführt:

- /1/ Mit dem rechten Mausklick auf die verschiedenen Bereiche im Editor erscheint ein Kontextmenü, um neue Elemente wie Datentypen, Gruppen und Attribute zum XML Schema hinzuzufügen.
- /2/ Im Directives-Bereich werden importierte Datentypen aufgelistet.
- /3/ Der Elements-Bereich gibt einen Überblick über alle XSD-Elemente, die in der aktuellen XML Schema-Datei existieren.
- /4/ Komplexe als auch einfache Datentypen die zum XML Schema gehören, werden in diesem Bereich angezeigt.
- /5/ Die im Dokument gefundenen Attribute werden hier aufgelistet.
- /6/ Die im Dokument gefundenen Gruppen werden hier aufgelistet.
- /7/ Hier kann zwischen den XML Quellcode- und Model-Ansichten ausgewählt werden.
- /8/ Die Eigenschaften der ausgewählten Elemente können in der Properties-Ansicht geändert werden.

PA POIMANAGERSERVICE – Datentyp POI in XML Schema-Editor

- (1) Datentyp mit XML Schema-Editor bearbeiten
 - (1) Ein "ComplexType" mit dem Namen "POI" erstellen
 - (2) Durch doppelten Mausklick kann der Typ "POI" in einem anderen Fenster bearbeitet werden
 - (3) Elemente mit Name und Datentyp festlegen
- (2) "Target Namespace" in der Properties-Ansicht ändern und speichern



10

21.04.2012

WASA - 3-2 WEB-SERVICE-KERNSTANDARDS

Cooperation & Management (C&M, Prof. Abeck)
Institut für Telematik, Fakultät für Informatik

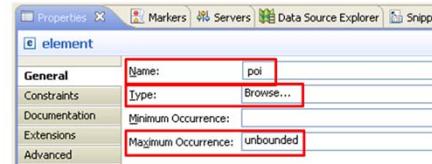
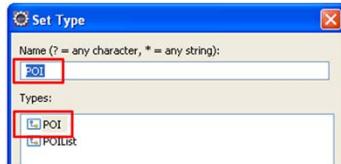
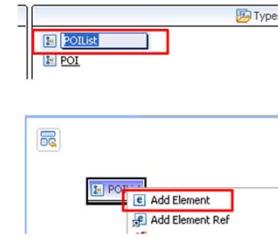
- (1) Nach dem ersten Erstellen einer XML Schema-Datei sollen die einzelnen Informationen zum Datentyp "POI" und zum XML Schema hinzugefügt werden.
 - (1.1) Für den Datentyp wird dem XML Schema durch rechten Mausklick im "Types"-Bereich für die Datei "POI.xsd" ein "ComplexType" namens "POI" hinzugefügt. Komplexe Typen werden für Elemente benutzt, die Kindelemente und Attribute enthalten können.
 - (1.2) (1.3) Durch doppeltes Klicken auf dem "ComplexType" können einzelne Elemente dem ComplexType hinzugefügt werden. Es sind folgende Elemente und Datentypen einzugeben:
 - id : integer
 - name : string
 - category : string
 - latitude : float
 - longitude : float
- (2) Nach der Konvention in [EK+08:174] hat die XSD die URL-Adresse "http://www.cm-tm.uni-karlsruhe.de/schema/POI" im "TargetNamespace". Durch die Wörter "schema" und "contract" lassen sich die zwei Geschäftsdomänen unterscheiden. Elemente, die das Wort "schema" im Namensraum enthalten, stellen Datentypen dar, die von anderen Web-Services verwendet werden können.

URL Uniform Resource Locator

[EK+08:174] T. Erl, A. Karmarkar, P. Walmsley, H. Haas. L. Umit Yalcinalp, K. Liu, D. Orchard, A. Tost and J. Pasley, Web-Service Contract Design and Versioning for SOA, Prentice Hall, 2008.

PA POIMANAGERSERVICE – Liste von Datentypen in XML Schema-Editor

- (1) Listen von Datentypen können mit XML Schema erstellt werden
 - (1) Einfache als auch komplexe Datentypen können benutzt werden
 - (2) Liste von POIs als auch von Kategorien sollen eine beliebige Anzahl an Elementen enthalten



- (1) Listen von Elementen werden im Rückgabeparameter der Operation "findPOIsByCategory" gebraucht. Diese Datenstruktur kann in XML Schema durch einen "ComplexType", der ein Array von Elementen enthält, definiert werden. Hierzu fügen wir dem XML-Schema einen "ComplexType" mit dem Namen "POIList" hinzu. Der "ComplexType" POIList enthält ein Element namens "poi", das durch das "maxOccurs"-Attribut eine Liste von Elementen definiert.
 - (1.1) Für den Datentyp einer Liste können einfache als auch komplexe Datentypen wie z.B. existierende Datentypen benutzt werden. Für die Liste der POIs soll der Datentyp "POI" ausgewählt werden.
 - (1.2) Durch Einstellen der Eigenschaften einer Liste in der Properties-Ansicht kann die Anzahl der enthaltenen Elemente festgelegt werden, wobei eine unbegrenzte Anzahl an Elementen auch möglich ist.

PA POIMANAGERSERVICE – Datentyp POI in Quellcode-Ansicht

```
1. <?xml version="1.0" encoding="UTF-8"?>
2. <schema xmlns="http://www.w3.org/2001/XMLSchema"
3. targetNamespace="http://www.cm-tm.uni-karlsruhe.de/schema/POI"
4. elementFormDefault="qualified"
5. xmlns:tns="http://www.cm-tm.uni-karlsruhe.de/schema/POI">
6.   <complexType name="POI">
7.     <sequence>
8.       <element name="id" type="integer"/>
9.       <element name="name" type="string"/>
10.      <element name="category" type="string"/>
11.      <element name="latitude" type="float"/>
12.      <element name="longitude" type="float"/>
13.    </sequence>
14.  </complexType>
15.  <complexType name="POIList">
16.    <sequence>
17.      <element name="poi" type="tns:POI" maxOccurs="unbounded"></element>
18.    </sequence>
19.  </complexType>
20.</schema>
```

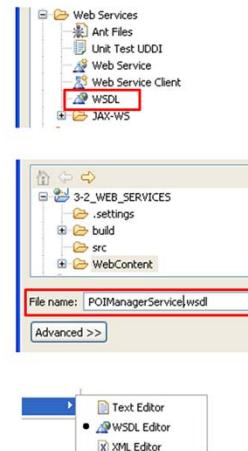
POI.xsd

Nach erfolgreichem Hinzufügen der notwendigen Elemente zum Datentyp "POI" erscheint in der Quellcode-Ansicht das auf der Folie angezeigte XML-Dokument. Das XML-Dokument fängt mit einem <schema>-Element an, das Angaben zu Namensräume macht.

- (1.) Die XML Schemas sind selbst XML-Dateien und am Anfang des Schemas sollte diese obligatorische Zeile auftauchen.
- (2.) Alle Elemente teilen den gemeinsamen Namensraum "<http://www.w3.org/2001/XMLSchema>".
- (3.) Der Attribut "targetNamespace" dient dazu, einen gemeinsamen Namensraum so zu definieren, dass alle global deklarierten Elemente und Typen Teil des Namensraums sein können.
- (6.) Komplexe Typen (complexTypes) werden für Elemente benutzt, die Kindelemente oder Attribute enthalten.
- (7.) Das <sequence>-Element bedeutet, dass die Elemente in der deklarierten Reihenfolge in der XML-Nachricht während einer Interaktion erscheinen sollen.

PA POIMANAGERSERVICE – WSDL in Eclipse

- (1) Anlegen einer WSDL-Datei
 - (1) Ordner "WebContent" auswählen und Name der Datei "POIManagerService.wsdl" eingeben
 - (2) "target namespace" definieren und Protokoll auswählen
 - (3) Datei zum Editieren mit dem WSDL-Editor öffnen



13

21.04.2012

WASA - 3-2 WEB-SERVICE-KERNSTANDARDS

Cooperation & Management (C&M, Prof. Abeck)
Institut für Telematik, Fakultät für Informatik

(1) Um die Dienstschnittstelle des Web-Services zu spezifizieren, soll eine WSDL-Datei generiert werden. Die Web Services Description Language (WSDL) ist als die wichtigste Technologie in der Entwicklung von Web-Services betrachtet. Mit Hilfe von WSDL lässt sich die Struktur der Schnittstelle definieren und mit anderen Technologien wie z.B. XSD kombinieren. Web-Service-Schnittstellen sind in einer Struktur organisiert, die eine klare Separierung folgender Fragen "was", "wie" und "wo" widerspiegelt [EK+08:51].

Die Aufgabe eines Web-Service und welche Operation er bietet stellen die abstrakte Sicht dar, die auf die "was"-Frage beantworten soll. In dieser Sicht werden Operationen und Nachrichten abstrakt beschrieben. Die andere Einteilung, die die anderen Fragen "wie" und "wo" beinhaltet soll, ist die konkrete Sicht. Hier können die vorher definierten Operationen und Nachrichten mit existierenden Protokollen und Formaten verbunden. Diese Unterteilung der WSDL-Schnittstelle in abstrakte und konkrete Sicht und die Verwendung von XML machen WSDL sehr flexibel.

(1.1) Im Ordner "WebContent" durch "New"->"Other"->"Web-Services"->"WSDL" eine neue WSDL Datei mit dem Namen "POIManagerService.wsdl" erstellen.

(1.2) Als target namespace ist URL-Adresse "http://www.cm-tm.uni-karlsruhe.de/contract/POIManagerService" einzugeben und als Protokoll SOAP auszuwählen. Der ausgewählte "target namespace" entspricht der Konvention in [EK+08:174].

(1.3) Die Datei zum Editieren mit dem WSDL-Editor öffnen. Der folgende XML-Ausschnitt aus der erstellen WSDL-Datei gibt ein Überblick über die Struktur einer WSDL-Schnittstelle:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<wsdl:definitions name="POIManagerService">
    <wsdl:types>
        ...
    </wsdl:types>
    <wsdl:message name="NewOperationRequest">
        <wsdl:part element="tns:NewOperation" name="parameters" />
    </wsdl:message>
    ...
    <wsdl:portType name="POIManagerService">
        <wsdl:operation name="NewOperation">
            ...
        </wsdl:operation>
    </wsdl:portType>
</wsdl:definitions>
```

```

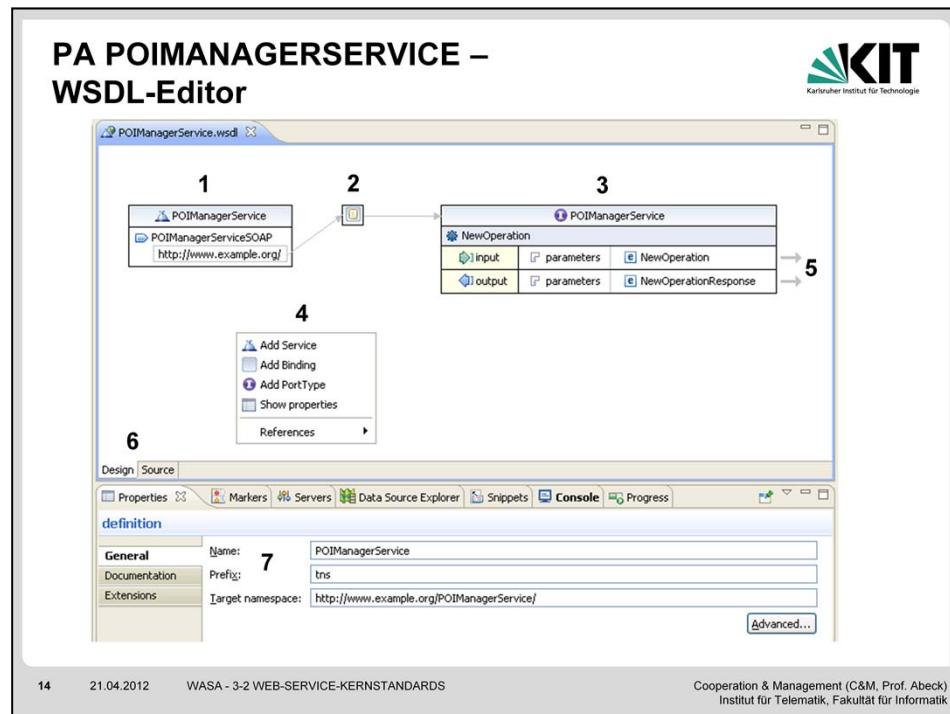
</wsdl:operation>
</wsdl:portType>

<wsdl:binding name="POIManagerServiceSOAP"
type="tns:POIManagerService">
    <soap:binding />
    <wsdl:operation name="NewOperation">
        ...
    </wsdl:operation>
</wsdl:binding>
<wsdl:service name="POIManagerService">
    <wsdl:port>
        ...
    </wsdl:port>
</wsdl:service>
</wsdl:definitions>
```

SOAP Simple Object Access Protocol

[EK+08:174] T. Erl, A. Karmarkar, P. Walmsley, H. Haas. L. Umit Yalcinalp, K. Liu, D. Orchard, A. Tost and J. Pasley, Web-Service Contract Design and Versioning for SOA, Prentice Hall, 2008.

PA POIMANAGERSERVICE – WSDL-Editor



Der WSDL-Editor stellt ein einfaches Werkzeug für die Bearbeitung einer WSDL-Datei dar.

- (1.) Dieses Element zeigt den Namen des Web-Services und seine Bindung an.
- (2.) Dieses Element beschreibt eine Bindung einer Operation in WSDL.
- (3.) Hier werden die angebotenen Operationen und Datentypen für die Eingabe- und Rückgabeparameter aufgelistet.
- (4.) Mit dem rechten Mausklick können neue Elemente wie z.B. Dienst, Bindung oder PortType zur WSDL-Definition hinzugefügt werden.
- (5.) Durch rechten Mausklick auf den Parametern der Operationen lassen sich existierende Datentypen finden und auswählen.
- (6.) Hier kann zwischen der XML Quellcode- und der Model-Ansicht ausgewählt werden.
- (7.) Die ausgewählten Elemente können in der Properties-Ansicht mit zusätzlichen Parametern eingestellt werden.

PA POIMANAGERSERVICE – Web-Service Operationen im WSDL-Editor

- (1) Name der bestehenden Operation "NewOperation" in der Properties-Ansicht ändern
- (2) Neue Operation durch "Add Operation" im Kontextmenü hinzufügen
- (3) "target namespace" der WSDL-Datei anpassen und speichern



15 21.04.2012

WASA - 3-2 WEB-SERVICE-KERNSTANDARDS

Cooperation & Management (C&M, Prof. Abeck)
Institut für Telematik, Fakultät für Informatik

- (1) Der WSDL-Editor legt beim ersten Erstellen einer WSDL-Datei eine Operation "NewOperation" an. Diese Operation soll in "getPOIById" umbenannt werden. Dazu stehen zwei Möglichkeiten zur Verfügung. Entweder in der Properties-Ansicht den Namen eingeben oder direkt im Editor durch doppeltes Klicken auf die Operation den neuen Namen eintragen.
- (2) Da der Web-Service eine andere Operation "findPOIsByCategory" in der Schnittstelle definieren soll, fügen wir eine neue Operation durch den Menüpunkt "Add Operation" im Kontextmenü zur Schnittstelle hinzu.
- (3) Falls der "target namespace" in der WSDL-Datei ein anderes Format hat, ist das Format entsprechend der Konvention in [EK+08:174] anzupassen. Die URL-Adresse "http://www.cm-tm.uni-karlsruhe.de/contract/POIManagerService" kann nun eingegeben werden.

Das Wort "contract" im Namensraum bedeutet, dass das XML-Dokument zur Geschäftsdomäne gehört, die die Schnittstellen Beschreibungen enthält. Durch den Namespace-Mechanismus wird im XML-Umfeld eine Modularität und Wiederverwendung erreicht [TEIA09]. XML Schema-Definitionen können einem Namespace zugeordnet werden, dem die Definitionen des Schemas zugeordnet werden. Dieser Namespace wird Target Namespace genannt.

[EK+08:174] T. Erl, A. Karmarkar, P. Walmsley, H. Haas. L. Umit Yalcinalp, K. Liu, D. Orchard, A. Tost and J. Pasley, Web-Service Contract Design and Versioning for SOA, Prentice Hall, 2008.

[TEIA09] TEIA AG - Internet Akademie und Lehrbuch Verlag, Konzepte Content-Repräsentation & Markup-Sprachen, <http://www.teialehrbuch.de/Kostenlose-Kurse/Markup-Sprachen/16512-XML-Namespace.html>, 2009.

PA POIMANAGERSERVICE – Web-Service Operationen in WSDL

```
1. <wsdl:portType name="POIManagerService">
2.   <wsdl:operation name="getPOIById">
3.     <wsdl:input message="tns:getPOIByIdRequest"/>
4.     <wsdl:output message="tns:getPOIByIdResponse"/>
5.   </wsdl:operation>
6.
7.   <wsdl:operation name="findPOIsByCategories">
8.     <wsdl:input message="tns:findPOIsByCategoriesRequest" />
9.     <wsdl:output message="tns:findPOIsByCategoriesResponse" />
10.    </wsdl:operation>
11. </wsdl:portType>
```

POIManagerService.wsdl

- (1) Die vom Web-Service bereitgestellten Operationen werden mit Hilfe des <portType>-Elements definiert
- (2) Für den POIManagerService werden request-response Operationen in den <portType>-Element benutzt

(1) Das wichtigste Element in einer WSDL-Datei ist das <portType>-Element. Es definiert den Web-Service, die Operationen und die Nachrichten, die beim Aufruf der Operation benutzt werden.

(2) Verschiedene Arten von Kommunikationstypen können während der Definition der Operationen verwendet werden. Da beim Aufruf des Web-Services POIManagerService POIs zurückgegeben werden, sind beide Operationen vom Typ Request-Response. Folgende Möglichkeiten sind auch verfügbar:

One-Way: Empfangen einer Nachricht

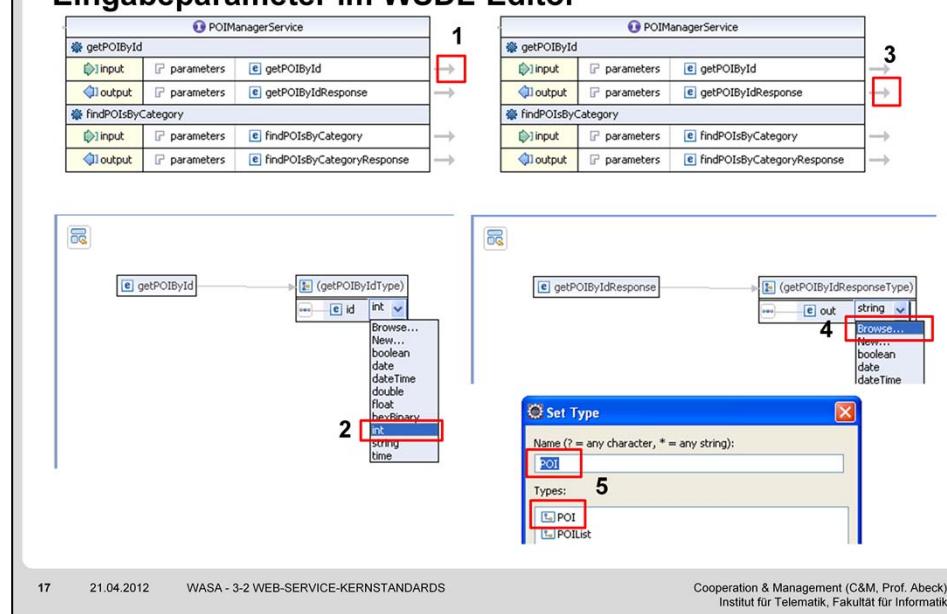
Request-Response: Empfangen einer Nachricht, dann Senden einer Nachricht

Solicit-Response: Senden einer Nachricht, dann Empfangen einer Nachricht

Notification: Senden einer Nachricht

Das Empfangen von Nachrichten wird dabei als <input>-Element dargestellt, das Senden als <output>-Element. Diese Elemente werden zusammen in das <operation>-Element eingefügt, das ein Unterelement des <portType>-Elements ist. Sowohl das <portType>-Element wie auch das <operation>-Element haben einen eindeutigen Namen.

PA POIMANAGERSERVICE – Eingabeparameter im WSDL-Editor



Da die Operationen bestimmte Informationen liefern sollen und eine Eingabe brauchen, sollen die Datentypen für die Eingabe- und Rückgabeparameter in WSDL festgelegt werden. Am Beispiel der Operation "getPOIById" soll gezeigt werden, wie Eingabe- und Rückgabeparameter mit bestimmten Datentypen konfiguriert werden können.

- (1) Durch Anklicken des ersten Pfeils wird das "getPOIById"-Element durchsucht und geändert werden.
- (2) Als Datentyp für den Eingabeparameter ID soll der einfache Datentyp "integer" ausgewählt werden.
- (3) Durch Anklicken des zweiten Pfeils lässt sich das "getPOIByIdResponse"-Element des Rückgabeparameters anpassen.
- (4) In diesem Fall soll nach einem existierenden Datentyp gesucht werden.
- (5) Im Suchfeld soll der Name "POI" eingegeben werden und den Typ "POI" ausgewählt werden.

Für die Operation "findPOIsByCategory" soll für den Eingabeparameter der Datentyp "string" und für den Rückgabeparameter den Datentyp "POIList" ausgewählt werden.

PA POIMANAGERSERVICE – Nachrichten in WSDL

```
1. <wsdl:message name="getPOIByIdRequest">
2.   <wsdl:part name="parameters" element="tns:getPOIById" />
3. </wsdl:message>
4.
5. <wsdl:message name="getPOIByIdResponse">
6.   <wsdl:part name="parameters" element="tns:getPOIByIdResponse" />
7. </wsdl:message>
8.
9. <wsdl:message name="findPOIsByCategoriesRequest">
10. <wsdl:part name="parameters" element="tns:findPOIsByCategories" />
11.</wsdl:message>
12.
13.<wsdl:message name="findPOIsByCategoriesResponse">
14. <wsdl:part name="parameters" element="tns:findPOIsByCategoriesResponse" />
15.</wsdl:message>
```

POIManagerService.wsdl

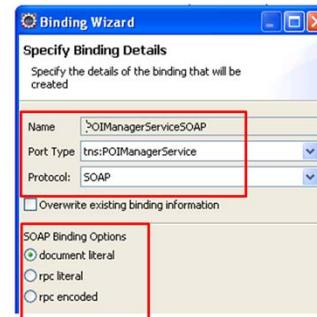
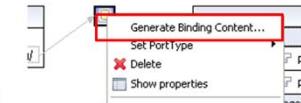
- (1) Für den Eingabe- und Rückgabeparameter werden Nachrichten mit eindeutigen Namen definiert
- (2) Nachrichten bestehen mindestens aus einem <part>-Element

(1) Nachrichten für die Eingabe und Rückgabeparameter haben eindeutige Namen innerhalb einer WSDL-Datei. Jedes <message>-Element (1.) muss einen eindeutigen Namen haben, wobei der Name nur innerhalb der definierten <message>-Elemente eindeutig sein muss.

(2) Ein <part>-Element definiert einen Nachrichtenparameter. Jedem <part> muss innerhalb des umgebenden <message>-Elements ein eindeutiger Name gegeben werden. Zusätzlich muss ein Datentyp angegeben werden, wobei man entweder in dem element-Attribut ein anderes XSD-Element, einen xsd:simpleType oder einen xsd:complexType in dem Attribut type angeben muss. Nachrichten bestehen aus einem oder mehreren <part>-Elementen (2.), die den Parametern der Nachricht darstellen.

PA POIMANAGERSERVICE – Bindung im WSDL-Editor

- (1) Informationen zum Netztransport und zur Kodierung eingeben
- (2) In WSDL 2.0 sind drei Transportarten und zwei Transportstile verfügbar



- (1) Mit Hilfe des Binding-Elements wird definiert wie die portTypes über das Netz transportiert und kodiert werden. Durch rechten Mausklick auf der Bindung im WSDL-Editor "Generate Binding Content" auswählen und einen Bindung erstellen. Durch Erstellen des Bindungsinhalt lassen sich neue Operationen in der Bindung hinzufügen und aktualisieren. In WSDL überführt eine Bindung die abstrakten Definitionen in eine konkrete Form. Daten- und Protokollformat für Nachrichten und Operationen werden bestimmt.
- (2) In WSDL 2.0. sind folgende Transportarten beschrieben: SOAP, direkt mit HTTP GET / POST oder mit Mime / SMTP. Transportstile sind: RPC oder Message / Document. Die Wahl der Nachrichtenformates und Protokolle ist frei. In WSDL ist SOAP als Nachrichtenformat und HTTP als Übertragungsprotokoll vordefiniert, es besteht aber die Möglichkeit beliebige Formate einzubinden.

PA POIMANAGERSERVICE – Bindung in WSDL

```
1.  <wsdl:binding name="POIManagerServiceSOAP" type="tns:POIManagerService">
2.    <soap:binding style="document"
3.      transport="http://schemas.xmlsoap.org/soap/http"/>
4.    <wsdl:operation name="getPOIById">
5.      <soap:operation soapAction="" />
6.      <wsdl:input>
7.        <soap:body use="literal" />
8.      </wsdl:input>
9.      <wsdl:output>
10.        <soap:body use="literal" />
11.      </wsdl:output>
12.    </wsdl:operation>
13.    ...
14.  </wsdl:binding>
```

POIManagerService.wsdl

- (1) Das Nachrichtenformat und die Details zum Protokoll eines Web-Services werden im <binding>-Element definiert
- (2) Einzelheiten zum SOAP-Protokoll und dessen Transport werden für jede Operation festgelegt

(1) Die WSDL Bindung definiert das Nachrichtenformat und die Protokolldetails für einen Web-Service. Ein <bindung>-Element hat zwei Attribute "name" und "type" und besitzt einen eindeutigen Namen und wird an ein portType-Element gebunden. Es sind mehrere Bindungen für ein portType-Element möglich. Dies erlaubt die Definition von mehreren Protokollen und Formaten für eine Operation.

(2.) Der Wert "document" des Attributs "style" gibt uns einen Hinweis über den Nachrichtenstil. In unserem Beispiel benutzen wir ein "document" Nachrichtenstil.

(3.) Der Wert des Attributes "transport" ist eine URI, die bestimmt, dass SOAP Nachrichten über HTTP übertragen werden sollen.

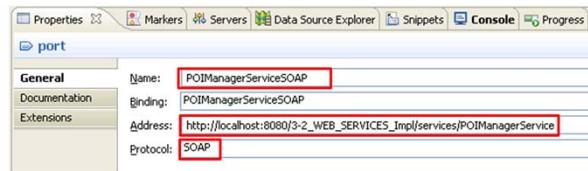
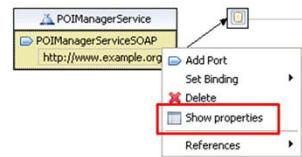
(2) In WSDL können für jede angebotene Operation bestimmte Bindungseigenschaften festgelegt werden.

(4.) Das <operation>-Element definiert den Port, der von der Operation bereitgestellt ist.

(7.) Der Wert "literal" des Attributs "use" aus den soap:body Elementen gibt an, welchen Nachrichtenstil die Operation beim Aufruf bzw. bei der Rückgabeantwort hat.

PA POIMANAGERSERVICE – Endpunkt und Adresse im WSDL-Editor

- (1) Ein Port stellt die Beziehung zwischen der Bindung und dem URL der Service-Implementierung her.
- (2) Von den verfügbaren Bindungen ist eine Bindung zum Zugriff auf den Web-Service und ein passendes Protokoll auszuwählen
- (3) Die Adresse des Web-Services in Form einer URL-Adresse eingeben



21 21.04.2012

WASA - 3-2 WEB-SERVICE-KERNSTANDARDS

Cooperation & Management (C&M, Prof. Abeck)
Institut für Telematik, Fakultät für Informatik

- (1) Die Eigenschaften des Ports wie z.B. Name, Bindung, Adresse und Protokoll lassen sich in der Properties-Ansicht eingeben und manuell einstellen.
- (2) Aus den vorher erstellten Bindungen ist eine Bindung auszuwählen, die beim Aufruf des Web-Services benutzt werden soll. Für Protokolle steht das SOAP- und das HTTP-Protokoll zur Verfügung.
- (3) Die Adresse, wo der Web-Service gefunden werden kann, ist entsprechend manuell einzugeben. Diese Adresse kann auf einen lokalen Host oder Server verweisen. In unserem Fall benutzen wir einen lokalen Host und der Web-Service ist unter der folgenden URL-Adresse erreichbar:
"http://localhost:8080/3-2_WEB_SERVICES_Impl/services/POIManagerService"

```
1. <wsdl:service name="POIManagerService">
2.   <wsdl:port name="POIManagerServiceSOAP" binding="tns:POIManagerServiceSOAP">
3.     <soap:address location="http://localhost:8080/3-
2_WEB_SERVICES_Impl/services/POIManagerService"/>
4.   </wsdl:port>
5. </wsdl:service>
```

POIManagerService.wsdl

- (1) Der Name des Web-Services, der durch die WSDL-Datei beschrieben ist, kann nun definiert werden
- (2) Einzelheiten zum Erreichen eines Web-Services werden im <port>-Element angegeben

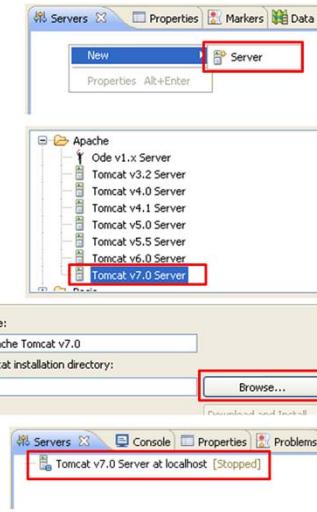
(1) Der Name des Web-Services lautet POIManagerService und ist im Element "service" im Attribut "name" eingegeben. Ein Service kann mehrere Ports haben und jeder Port beschreibt eine Möglichkeit den Service aufzurufen. In unserem Beispiel gibt es nur einen Port vom Typ SOAP 1.1.

(2) Das Kindelement "address" hat einen anderen XML Prefix als die anderen Elemente. Der Prefix soap ist an das SOAP 1.1 Binding aus diesem Dokument gebunden. Anstatt des SOAP Bindings können auch andere Bindings für den JMS oder Dateizugriff verwendet werden. Das Element "address" besitzt ein Attribut namens "location", welches auf Adresse des Web-Services zeigt.

(3.) Der Web-Service ist in unserem Fall unter der Adresse "http://localhost:8080/3-2_WEB_SERVICES_Impl/services/POIManagerService" erreichbar. Die WSDL-Datei kann nach dem erfolgreichen Deployen des Web-Services unter "http://localhost:8080/3-2_WEB_SERVICES_Impl/services/POIManagerService?wsdl" abgerufen werden.

PA POIMANAGERSERVICE – Apache Tomcat Server in Eclipse

- (1) Apache Tomcat herunterladen und installieren
- (2) In Eclipse einen Servertyp auswählen und einen Server einrichten

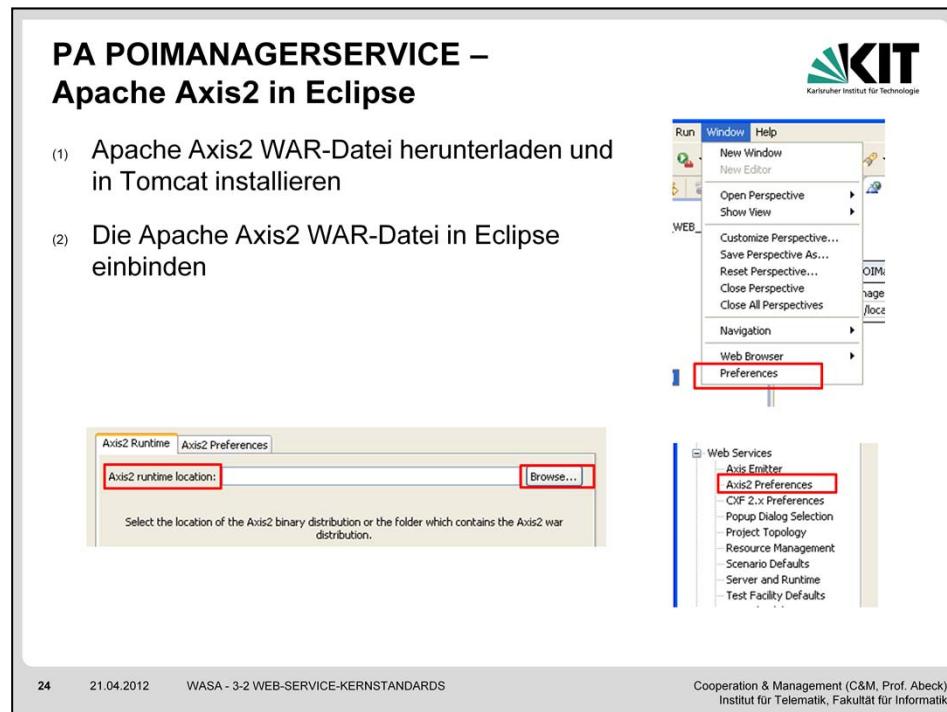


(1) Um Tomcat in Eclipse einbinden zu können, soll eine Version heruntergeladen und in einen Ordner entpackt werden. Folgende Version von Tomcat wurde ausgewählt: "http://apache.mirror.digionline.de/tomcat/tomcat-7/v7.0.16/bin/apache-tomcat-7.0.16.zip". Nach dem Herunterladen kann die Datei entpackt werden und Tomcat installiert werden.

(2) Nachdem die Apache Tomcat installiert wurde, soll in Eclipse die installierte Version von Tomcat als Server eingerichtet werden. Unter den Menüpunkt "Windows"->"Show View"->"Servers" die Ansicht "Servers" auswählen. In der Ansicht "Servers" mit dem rechten Mausklick durch "New"->"Server" einen neuen Server anlegen. Eine Reihe von Servern wie z.B. IBM WebSphere und JBoss sind in Eclipse unterstützt. In unserem Fall wählen wir den kostenlosen und weit bekannten Apache Tomcat Server aus. Ein Fenster zum Konfigurieren von Tomcat erscheint, wo der Pfad zum Tomcat Installationsordner eingegeben werden soll.

PA POIMANAGERSERVICE – Apache Axis2 in Eclipse

- (1) Apache Axis2 WAR-Datei herunterladen und in Tomcat installieren
- (2) Die Apache Axis2 WAR-Datei in Eclipse einbinden

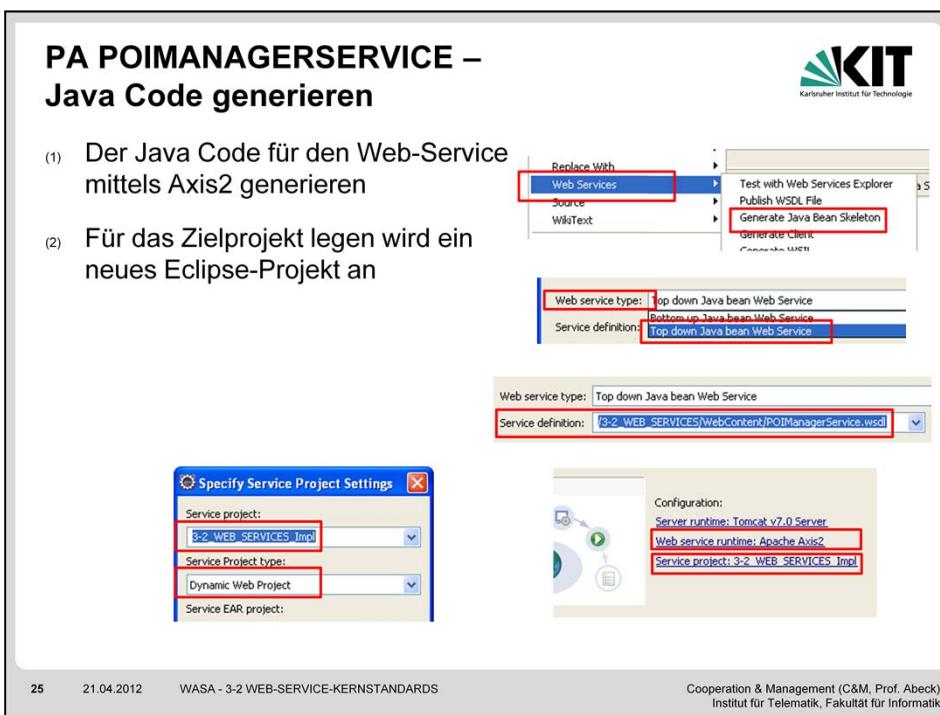


(1) Die Apache Axis2 unter "<http://archive.apache.org/dist/axis/axis2/java/core/1.5.4/axis2-1.5.4-war.zip>" herunterladen und entpacken. Nach dem entpacken die Datei "axis2.war" in den webapps-Ordner von Tomcat kopieren. WAR steht für Web Archive oder Web Application Archive und ist ein Dateiformat, das das Verpacken einer vollständigen Webanwendung nach der Java-Servlet-Spezifikation in Form von JAR- bzw. ZIP-Format beschreibt.

(2) Nachdem die Apache Axis2 WAR-Datei entpackt und in Tomcat kopiert wurde, kann nun Eclipse mit Axis2 konfiguriert werden. Unter "Window"->"Preferences" die Einstellung "Web-Services"->"Axis2 Preferences" auswählen und im Tab "Axis2 runtime location" durch den Browse-Button den Pfad zum webapps-Ordner von Tomcat, wo die WAR-Datei sich befindet, eingeben. Wenn dieser korrekt ist sollte die Meldung "Axis2 runtime loaded successfully" erscheinen. Klicken Sie auf OK.

PA POIMANAGERSERVICE – Java Code generieren

- (1) Der Java Code für den Web-Service mittels Axis2 generieren
- (2) Für das Zielprojekt legen wird ein neues Eclipse-Projekt an



(1) Der Java Code kann nun aus der WSDL-Datei nach dem Top-Down-Ansatz (Contract First) generiert werden. Durch rechten Mausklick auf die WSDL-Datei "POIManagerService.wsdl" den Menüpunkt "Web Services"->"Generate Java Bean Skeleton" auswählen. Es erscheint ein Dialog, um die Überführung zu konfigurieren. In "Web service type" den "Top Down"-Ansatz aus dem Dropdown auswählen und die WSDL-Datei in "Service definition" eingeben.

Nachdem der Ansatz und die WSDL-Schnittstelle festgelegt wurden, kann nun die Web-Service Laufzeitumgebung ausgewählt werden. Durch Anklicken auf den Eintrag "Web service runtime" unter den drei Web-Service Laufzeitumgebungen Apache Axis, Apache Axis2 und Apache CXF 2.x ausgewählt werden. In dieser praktischen Aufgabe benutzen wir die Apache Axis2 Laufzeitumgebung. Sie bietet eine Reihe von Optionen, um WSDL-Elemente in Java-Objekten zu überführen.

(2) Die generierten Dateien und Artefakte werden in ein separates Projekt erstellt. Das Workspace-Verzeichnis von Eclipse sollte in einem Pfad liegen, der keine Leerzeichen enthält. Der Axis2 Code Generator stürzt ab, wenn die WSDL-Datei in einem Pfad mit Leerzeichen liegt und verhindert eine vollständige Erzeugung der Software-Artefakte.

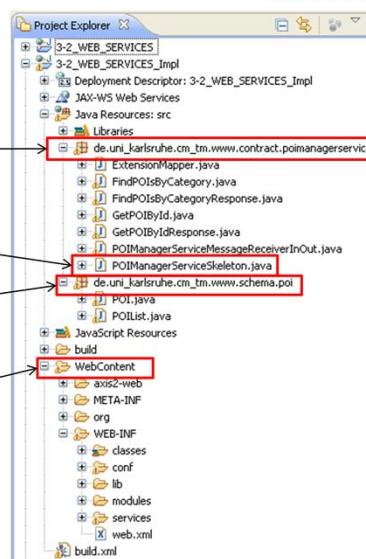
PA POIMANAGERSERVICE – Übersicht Code Stub

Package für die Web-Service-Schnittstelle

Web-Service Implementierung

Package für Datentyp POI

Ordner für Artefakte und Webinhalte



(Package für die Web-Service-Schnittstelle) Der Axis2 Code Generator erstellt dieses Package, um die Elemente der Schnittstelle hier zu speichern. Der Name des Packages stammt aus dem ausgewählten Namensraum in der WSDL-Datei. Dieses Package enthält neben den Klassen für Nachrichtenformate für jede Operation (FindPOIByCategory.java und FindPOIByCategoryResponse.java) andere Klassen. Die Klasse "ExtensionMapper.java" stellt ein Parser für eingelesene XML-Elemente dar, die anderen Datentypen erweitert haben. Das Ziel ist es, die entsprechende Klasse dieser XML-Elemente zu finden. Die Klasse "POIManagerServiceMessageReceiverInOut.java" stellt eine Brücke zwischen Axis2 Engine und der Dieninstanz und dient dazu eine Nachricht zu empfangen und eine Antwort zu generieren.

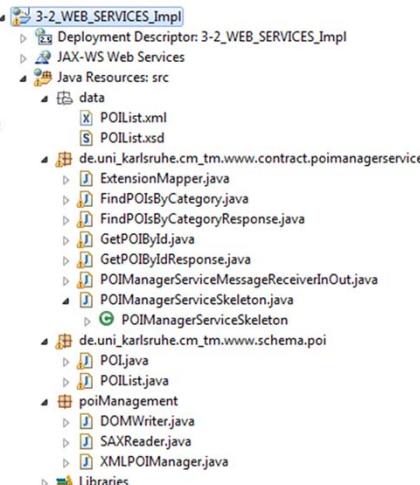
(Web-Service Implementierung) Diese Klasse beschreibt die Implementierungslogik des Web-Services und sollte erweitert werden.

(Package für Datentyp POI) Hier ist der Datentyp POI in Form einer Klasse mit Attributen beschrieben.

(Ordner für Artefakte und Webinhalte) Die kompilierten Klassen sowie notwendige Artefakte werden in diesem Ordner gespeichert.

PA POIMANAGERSERVICE – Einbinden der POIManagement-Funktionalität

- (1) Das von den Axis2-Tools erstellte POI.java ist zu benutzen
- (2) Ein Package "data" erstellen und die "POIList.xml" und "POIList.xsd"
- (3) Das Package "poiManagement" aus den vorherigen PAs in den "src"-Ordner kopieren
- (4) Die Methoden aus "POIManagerServiceSkeleton.java" sind zu vervollständigen



- (1) Die von den Axis2 Tools erstellte Klasse "POI.java" ist zu benutzen, da sie zu der manuell Erstellten Klasse "POI.java" aus den vorherigen PAs äquivalent ist.
- (2) Da der Classloader im Fall von Web-Services nicht auf den Inhalten im Ordner WEB-INF zugreifen kann wird die POI-Liste "POIList.xml" in das Package "data" verlagert.
Das XSD-Dokument ist nur da, um beim manuellen Verändern und Validieren der XML-POI-Liste zu dienen. Es wird in diesem Schritt nicht von Tomcat benötigt, da die darin enthaltenen Datentypen schon in der WSDL benutzt wurden.
- (3) Die Klassen aus der [C&M-PA-INFORMATION-11] können auch hier ohne Probleme eingesetzt werden, um die benötigte Funktionalität zu implementieren.

[C&M-PA-INFORMATION-11] Cooperation & Management: WASA - INFORMATION, Praktische Aufgaben, Karlsruher Institut für Technologie (KIT), C&M (Prof. Abeck).

PA POIMANAGERSERVICE – Implementierung der ersten Operation

- (1) "POIManagerServiceSkeleton" ist eine von Axis2 automatisch erstellte Klasse

- (2) Die "getPOIById"-Methode bietet folgende Funktionalität an

```
1. public class POIManagerServiceSkeleton{  
2.     public GetPOIByIdResponse getPOIById(GetPOIById getPOIById){  
3.         XMLPOIManager poiMan = new XMLPOIManager();  
4.         URL poiListLocation = this.getClass().getResource("/data/POIList.xml");  
5.         List<POI> poiList = poiMan.getAllPOIs(poiListLocation.toString());  
6.         GetPOIByIdResponse response = new GetPOIByIdResponse();  
7.         for(POI p : poiList){  
8.             if(p.getId() == getPOIById.getId()){  
9.                 response.setOut(p);  
10.            }  
11.        }  
12.    }  
13.    return response;  
14. }
```

POIManagerServiceSkeleton.java

(1.) Eine Instanz der Klasse "POIManagerServiceSkeleton" wird von Tomcat benutzt, um eingehende Anfragen zu beantworten.

(2.) Die Deklaration der "getPOIById"-Methode.

(4.) Instanziieren des XMLPOIMangers, um die benötigte XML-Funktionalität zu benutzen.

(5.) Mit der "getResource"-Methode kann man die Adresse der XML-POI-Liste erhalten, um von dort die POI-Daten auslesen zu können.

(6.) Aufruf des XMLPOIMangers, um eine Liste von POIs zu erhalten.

(8.) Um eine Antwort abschicken zu können, muss sie in die passenden Datentypen verpackt werden. Deshalb wird ein "GetPOIByIdResponse"-Objekt erstellt.

(8.-10.) Anhand einer For-Each-Schleife, die die vom XMLPOIManager erhaltene Liste iteriert, wird eine POI zum Antwortobjekt "response" hinzugefügt, falls sie eine gleiche ID zur übergebenen ID hat.

(13.) Die Antwort "response" wird als Objekt vom Typ "GetPOIByIdResponse" zurückgegeben.

PA POIMANAGERSERVICE – Implementierung der zweiten Operation

- (1) "POIManagerServiceSkeleton" ist eine von Axis2 automatisch erstellte Klasse

- (2) Die "findPOIsByCategory"-Methode bietet folgende Funktionalität an

```
1. public class POIManagerServiceSkeleton{  
2.     public FindPOIsByCategoryResponse findPOIsByCategory(FindPOIsByCategory  
3.         {  
4.             XMLPOIManager poiMan = new XMLPOIManager();  
5.             POIList responsePOIList = new POIList();  
6.             URL poiListLocation = this.getClass().getResource("/data/POILIST.xml");  
7.             List<POI> poiList = poiMan.getAllPOIs(poiListLocation.toString());  
8.             for (POI p : poiList) {  
9.                 if (p.getCategory().equalsIgnoreCase(  
10.                     findPOIsByCategory.getCategory())) {  
11.                         responsePOIList.addPoi(p);  
12.                     }  
13.                 }  
14.             FindPOIsByCategoryResponse response = new FindPOIsByCategoryResponse();  
15.             response.setOut(responsePOIList);  
16.             return response;  
17.         }  
    }
```

POIManagerServiceSkeleton.java

29 21.04.2012 WASA - 3-2 WEB-SERVICE-KERNSTANDARDS

Cooperation & Management (C&M, Prof. Abeck)
Institut für Telematik, Fakultät für Informatik

(1.) Eine Instanz der Klasse "POIManagerServiceSkeleton" wird von Tomcat benutzt um eingehende Anfragen zu beantworten.

(2.) Die Deklaration der "findPOIByCategory" Methode.

Als Parameter wird ein Objekt des Typs "de.uni_karlsruhe.cm_tm.www.contract.poimanagerservice.FindPOIsByCategory" eingegeben.

Als Rückgabe liefert die Methode ein Objekt des Typs "de.uni_karlsruhe.cm_tm.www.contract.poimanagerservice.FindPOIsByCategoryResponse" zurück.

(4.) Instanziieren des XMLPOIMangers, um die benötigte XML-Funktionalität zu benutzen.

(5.) Hier wird ein "POIList"-Objekt instanziert.

(6.) Mit der "getResource"-Methode kann man die Adresse der XML-POI-Liste erhalten und von dort die POI-Daten auslesen.

(7.) Aufruf der "getAllPOIs"-Methode des XMLPOIMangers, um eine Liste von POIs zu erhalten.

(8.-13.) Es wird mit einer For-Each-Schleife durch die vom XMLPOIManager erhaltene Liste iteriert und die passenden POIs werden mit Hilfe der "addPoi"-Methode des "POIList"-Objekts in das "responsePOIList"-Objekt eingefügt.

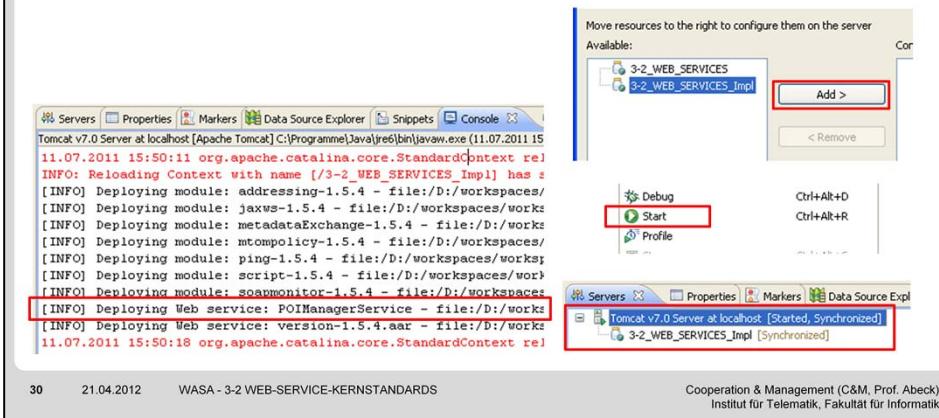
(14.) Um eine Antwort abschicken zu können, muss sie in die passenden Datentypen verpackt werden. Deshalb wird ein "FindPOIsByCategoryResponse"-Objekt erstellt.

(15.) Die Liste von POIs, die im "POIList"-Objekt "responsePOIList" gekapselt sind, wird dem "response"-Antwortobjekt mit der Methode "setOut" hinzugefügt.

(16.) Die Antwort wird als Objekt "response" vom Typ "FindPOIsByCategoryResponse" zurückgeliefert.

PA POIMANAGERSERVICE – Web-Service deployen

- (1) Das von Axis2 erstellte Eclipse-Projekt in Tomcat einbinden
- (2) Tomcat Server starten und den Web-Service aufrufen



(1) Um den Web-Service auffindbar zu machen, soll das von Axis2 erstellte Eclipse-Projekt in Tomcat eingebunden werden. Durch die Option "Add and Remove" lässt sich das aktuelle Projekt zum Server einbinden und gleichzeitig deployen. Eine andere Möglichkeit zum deployen wäre eine WAR-Datei aus dem aktuellen Projekt zu exportieren und in den webapps-Ordner von Tomcat zu kopieren. Die WAR-Datei sollte den Namen "3-2_WEB_SERVICES_Impl.war" haben. Bei Änderungen in Quellcode soll diese WAR-Datei nochmal generiert und in Tomcat kopiert werden.

(2) Tomcat aus der Servers-Ansicht anschließend starten. In der Konsole werden Debug- und Startinformationen angezeigt. Wenn die Web-Services mit Hilfe von Eclipse deployed wurden, werden Änderungen nach dem Speichern automatisch publiziert. Diese Alternative ist in Entwicklungsumgebungen zum Testen von Änderungen sehr hilfreich.

PA POIMANAGERSERVICE – Web-Service im Browser aufrufen

- (1) Nach erfolgreichem Deployen des Web-Services kann die WSDL-Schnittstelle des Web-Services nun im Browser aufgerufen werden

The screenshot shows a web browser with two tabs open. The left tab displays the Apache Software Foundation logo and URL. The right tab shows the WSDL definition for the POIManagerService. The XML code is as follows:

```
<wsdl:definitions name="POIManagerService" targetNamespace="http://www.cm-tm.uni-karlsruhe.de/POIManagerService">
  <wsdl:types>
    <xsd:schema targetNamespace="http://www.cm-tm.uni-karlsruhe.de/POIManagerService">
      <xsd:import namespace="http://www.cm-tm.uni-karlsruhe.de/POIManagerService"/>
      <xsd:element name="getPOIById">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="id" type="xsd:int"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
      <xsd:element name="getPOIByIdResponse">
        <xsd:complexType>
          <xsd:sequence>
            <xsd:element name="POI" type="tns:POI"/>
          </xsd:sequence>
        </xsd:complexType>
      </xsd:element>
    </xsd:schema>
  </wsdl:types>
  <wsdl:message name="getPOIByIdRequest">
    <wsdl:part name="parameters" type="tns:POI"/>
  </wsdl:message>
  <wsdl:message name="getPOIByIdResponse">
    <wsdl:part name="body" type="tns:POI"/>
  </wsdl:message>
  <wsdl:operation name="getPOIById">
    <wsdl:input message="getPOIByIdRequest"/>
    <wsdl:output message="getPOIByIdResponse"/>
  </wsdl:operation>
</wsdl:definitions>
```

Below the tabs, the browser status bar shows the URLs: http://localhost:8080/3-2_WEB_SERVICES_Impl/services/listServices and http://localhost:8080/3-2_WEB_SERVICES_Impl/services/POIManagerService?wsdl.

31

21.04.2012

WASA - 3-2 WEB-SERVICE-KERNSTANDARDS

Cooperation & Management (C&M, Prof. Abeck)
Institut für Telematik, Fakultät für Informatik

- (1) Der Web-Service kann mit dem Web-Browser aufgerufen werden. Axis2 bietet eine Seite um aktive Dienste anzuzeigen und sich über den Status der einzelnen Dienste zu informieren. Die WSDL-Datei kann im Browser durch Hinzufügen des Kürzels "?wsdl" am Ende der URL-Adressen angezeigt werden.

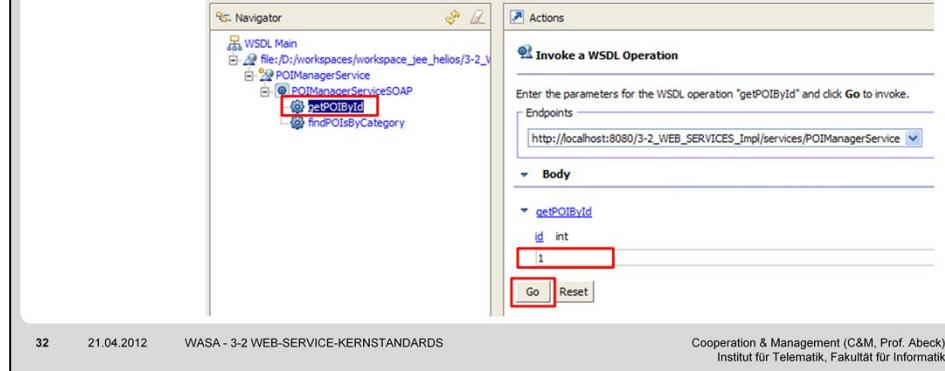
Der Port und Server in den folgenden URL-Adressen kann entsprechen den Einstellungen von Tomcat angepasst werden:

http://<server>:<port>/3-2_WEB_SERVICES_Impl/services/listServices

http://<server>:<port>/3-2_WEB_SERVICES_Impl/services/POIManagerService?wsdl

PA POIMANAGERSERVICE – Web-Service mit Web Services Explorer

- (1) Tool in Eclipse um Web-Service-Aufrufe (Request) zu erstellen
- (2) Eingabe- als auch Rückgabeparameter können auch in Form von XML angezeigt werden.



32

21.04.2012

WASA - 3-2 WEB-SERVICE-KERNSTANDARDS

Cooperation & Management (C&M, Prof. Abeck)
Institut für Telematik, Fakultät für Informatik

(1) In Eclipse ist ein Tool enthalten, um Web-Services besser testen zu können. Durch rechten Mausklick auf die WSDL-Datei kann im Kontextmenü über den Menüpunkt "Web Services"->"Test with Web Services Explorer" die WSDL-Datei mit dem Web Services Explorer geöffnet werden. Der Web Services Explorer ist eine JSP Webanwendung, die auf dem Apache Tomcat Servlet Engine gehostet ist. Diese Anwendung ist in Eclipse integriert und läuft als Thread in Eclipse JRE. Der Web Services Explorer bietet drei folgenden Dienste an:

Umfassende Unterstützung für das Auffinden von Web-Services in WS-Inspection 1.0 Dokumenten und in UDDI-Registern mit Hilfe des UDDI-Protokolls.

Umfassende Unterstützung für die Veröffentlichung von Web-Services zu UDDI-Registern mit dem UDDI-Protokoll.

Umfassende Unterstützung für das Aufrufen von Web-Services über ihre WSDL und XSDs.

(2) Der Web Services Explorer bietet neben der Eingabe in Form-Ansicht auch die Möglichkeit, Eingabe- als auch Ausgabeparameter in Quellcode-Ansicht anzuzeigen. Die Eingabe in Form eines XML-Ausschnitts ist bei der Übergabe von mehreren Daten wie z.B. Listen von Datentypen sehr hilfreich.

UDDI

Universal Description, Discovery and Integration (UDDI)

PA POIMANAGERSERVICE – Test der Web-Service-Operationen

/1/

Web Services Explorer

getPOIById

id int
1

Go Reset

Status

Body

getPOIByIdResponse

out

id (int): 1

name (string): Mensa am Adenauerring
category (string): mensa
longitude (float): 8.41678
latitude (float): 49.011925

/2/

Mit dieser XML-Datei sind anscheinend keine Style-Informationen verknüpft. Dokuments angezeigt.

```
<ns2:findPOIsByCategoryResponse>
  - <out>
    - <poi>
      <id>2</id>
      <name>Chemie Bibliothek</name>
      <category>bibliothek</category>
      <longitude>8.413655</longitude>
      <latitude>49.01128</latitude>
    </poi>
    - <poi>
      <id>3</id>
      <name>24h Bibliothek</name>
      <category>bibliothek</category>
      <longitude>8.416523</longitude>
      <latitude>49.011013</latitude>
    </poi>
```

33

21.04.2012

WASA - 3-2 WEB-SERVICE-KERNSTANDARDS

Cooperation & Management (C&M, Prof. Abeck)
Institut für Telematik, Fakultät für Informatik

Diese Folie zeigt das Ergebnis der Web-Service-Aufrufe für die beiden Operationen mit dem Web Services Explorer und im Browser an.

/1/ Bei der Eingabe im Eingabefeld des Web Services Explorers für die erste Operation, können Parameter dem Web-Service-Aufruf übergeben werden. Als Beispiel die Eingabe id = 1 liefert eine POI mit der richtigen ID, die aus einer POI-Liste eingelesen wurde.

/2/ Eine andere Möglichkeit eine Operation des Web-Services aufrufen zu können ist der Webbrower. Bei zweiten Aufruf werden alle POIs für eine bestimmte Kategorie zurückgegeben. Die Übergabe der Parameter im Webbrower wird durch folgende Form durchgeführt:

`http://<server>:<port>/Service/Operation?param1=wert1¶m2=wert2`

Für unseren Fall rufen wir die Operation "findPOIsByCategory" mit dem Parameter "category", das den Wert "bibliothek" hat, auf. Die URL-Adresse sieht dann folgendermaßen aus:

`http://localhost:8080/3-2_WEB_SERVICES_Impl/services/POIManagerService/findPOIsByCategory?category=bibliothek`

PA POIMANAGERSERVICE – Ausblick

- (1) Die Web-Service Technologie stellt eine Lösung zur Bereitstellung von Funktionalität der einzelnen Komponenten
- (2) Web-Services können durch ihre Komposition gemeinsam zur Lösung einer Aufgabe beitragen
- (3) In einer Praktischen Aufgabe in der Kurseinheit WEB-SERVICE-KOMPOSITION wird eine Komposition von Web-Services erstellt und am Beispiel des Geschäftsprozesses zur Workshop-Organisation veranschaulicht

(1) Web-Service stellen eine robuste Technologie, um Funktionalität bestehender Komponenten bereitzustellen. Web-Service können dann Funktionalität von z.B. Legacy-Systemen durch Erstellen einer Fassade und einen Adapter breitstellen. Dadurch können neue Anwendungen mit einander interagieren.

(2) Web-Services können durch ihre Komposition gemeinsam eine Aufgabe Lösen und dadurch tragen sie zur Bildung eines Geschäftsprozesses bei. Für die Komposition von Web-Services existieren verschiedene Technologien und Standards.

(3) Die nächste praktische Aufgaben zeigt an, wie eine Komposition von Web-Services unter der Entwicklungsumgebung erstellt werden kann. Das Szenario ist die Organisation eines Arbeitstreffen auf dem Campus. Hier wird ein Raum für Anzahl der Teilnehmer gesucht und alle relevanten Orte wie z.B. Menschen, Cafeterien und Bibliotheken in der Umgebung ermittelt.

Es ist ein Geschäftsprozess zur Organisation eines Workshops auf dem KIT Campus durch einen BPEL-Prozess mittels der Apache Orchestration Director Engine (ODE) zu unterstützen. Der BPEL-Prozess benötigt die folgenden zwei Web-Services:

- (1) Web-Service "POIManagerService" stellt Operationen bereit, um POIs abzufragen
 - (1) Die Operation "getNextPOIByCategory" gibt den nächstgelegenen POI für eine bestimmte Kategorie zurück
- (2) Web-Service "RoomManagerService" dient zur Verwaltung und Reservierung von Räumlichkeiten auf dem Campus
 - (1) Die Operation "getRoom" sucht einen Raum, der für die Anzahl der Teilnehmer passt

In dieser Praktischen Aufgabe (PA) ist der einer Workshop-Organisation (WSORG) auf dem KIT Campus zugrundeliegende Prozess durch einen BPEL-Prozess zu unterstützen. In dieser praktischen Aufgabe werden zwei Web-Services mit einander komponiert und verknüpft. Zur Beschreibung der Servicekomposition wird die XML-basierte Sprache Business Process Execution Language (BPEL) [OASIS-BPEL-V2] benutzt.

(1) Die Verwaltung einer PointOfInterest-Menge (POI) wird mit Hilfe des Services "POIManagerService" unterstützt. Der Service ist ein vom KITCampusGuide-System bereitgestellter Web-Service.

(1.1) Die Operation sucht nach dem nächstgelegenen POI für die ausgewählte Kategorie. Hier wird der Abstand zwischen dem Raum und allen POIs in dieser Kategorie berechnet und anschließend der POI mit dem kleinstem Abstand zurückgegeben. Kategorien sind relevante Arten von Punkten auf dem Campus, wie z.B. Menschen, Bibliotheken, Cafeterien und Parkplätze. In dieser Aufgabe werden die Längen- und Breitengrad der POIs für die verschiedenen Kategorien mit zufälligen Zahlen so befüllt, dass bei erneuter Abfrage der Operation POIs mit neuen Positionen zurückgegeben werden können.

(2) Der Dienst "RoomManagerService" ist ein vom Facility-Management-System bereitgestellter Web-Service und stellt Funktionalität bereit, um Räume auf dem Campus zu verwalten.

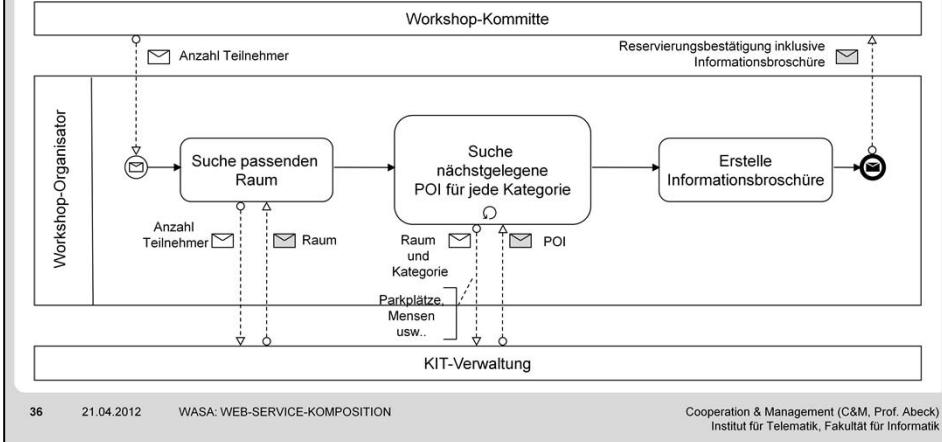
(2.1) Bei der Suche nach einem bestimmten Raum ist die Anzahl der Teilnehmer als Parameter für die Operation "getRoom" einzugeben (die Kapazitäten der Räume ist in Schritten von jeweils 10 Teilnehmern angegeben, d.h. Räume mit einer Kapazität von 1 bis 11, von 11 bis 20, ...).

[OASIS-BPEL-V2] OASIS, Business Process Execution Language, <http://docs.oasis-open.org/wsbpel/2.0/wsbpel-specification-draft.html>, 2006.

BPEL	Business Process Execution Language
PA	Praktische Aufgabe
POI	PointOfInterest
WSORG	Workshop-Organisation

PA WSORG-BPELPRO – BPMN-Prozess zur Workshop- Organization

- (1) Der BPMN-Prozess stellt den komponierten Web-Service dar, der andere Web-Service-Operationen aufruft
 - (1) Der logische Fluss als auch die Reihenfolge der Web-Service-Aufrufe wird durch den modellierten Geschäftsprozess bestimmt



Der in dieser Aufgabe umzusetzende Geschäftsprozess ist eine vereinfachte Variante des in der Kurseinheit SERVICEORIENTERTE ARCHITEKTUREN UND ENTWICKLUNG angegebenen BPMN-Prozesses zur Workshop-Organisation

- (1) Der komponierte Dienst entsteht durch die Kombination von bestehenden Services (sog. Komponentenservices). Die Komponentenservices können ihrerseits elementar oder komplex sein.
- (1.1) Das BPMN-Modell in der Folie stellt den Prozess zur Organisation eines Workshops dar. Der Prozess enthält zwei Aktivitäten zum Abrufen der notwendigen Informationen, die erste Aktivität sucht einen Raum für die angegebene Anzahl von Teilnehmern. Die zweite Aktivität sucht für jede Kategorie die nächstgelegenen POIs (PointOfInterest), wobei die Kategorien vom Benutzer festgelegt und ausgewählt werden sollen. Die letzte Aktivität ist eine Aktivität, die die Teilergebnisse zusammenfügt und zurückgibt. Der logische Fluss wird durch den modellierten Geschäftsprozess bestimmt. Zur Modellierung des Geschäftsprozesses wird die Modellierungssprache Business Process Model and Notation (BPMN) benutzt [OMG-BPMN-2.0]. Die Modellierungssprache BPMN bietet Elemente an, um die Arbeitsabläufe eines Geschäftsprozesses und die beteiligten Partner dargestellt in Form von Rollen modellieren zu können.

BPMN

Business Process Model and Notation

[OMG-BPMN-2.0] OMG, Business Process Model and Notation, www.omg.org/spec/BPMN/2.0/, 2011

PA WSORG-BPELPRO – Vorgehen

- (1) Die beiden beteiligten Web-Services werden nach dem Top-Down Ansatz erstellt
 - (1) Die WSDL-Schnittstellen mit den notwendigen Operationen werden mit Hilfe von Axis2 in Quellcode überführt
 - (2) Der Quellcode wird angepasst, um die Funktionalität der Services zu implementieren
 - (3) Beide Web-Services werden als WAR-Datei bereitgestellt und auf dem Tomcat verteilt
- (2) Für den komponierten Dienst soll ein BPEL-Prozess zur Orchestrierung erstellt und in Apache ODE verteilt werden
- (3) Aufruf des BPEL-Prozesses im Browser und Test des gesamten Prozessablaufs

(1) Die zu komponierenden Web-Services sollen nach dem Top-Down-Ansatz entwickelt werden. Für weitere Informationen zum Erstellen von Web-Service-Schnittstellen und ihren Operationen mittels Eclipse wird auf die in der Kurseinheit WEBSERVICE-KERNSTANDARDS enthaltene Praktische Aufgabe POIMANAGERSERVICE verwiesen.

(1.1) Die Operationen für die Web-Services werden durch die Web Service Description Language (WSDL) beschrieben und mit Hilfe des Codegenerators von Apache Axis2 [Apache-Axis2] in Quellcode überführt. Dadurch werden Java-Klassen generiert, die ein Web-Service konstruieren und in Tomcat verteilt (engl. deploy) werden können.

(1.2) Der automatisch erstellte Quellcode ist um die notwendige Funktionalität zu erweitern. Hier können nicht nur bestehende Klassen ergänzt werden, sondern auch neue Bibliotheken eingebunden werden.

(1.3) Die Web-Services werden in eine WAR-Datei (Web Archive) exportiert und anschließend auf den Apache Tomcat verteilt. Axis2 bietet im Vergleich zu Axis 1.x ein einfaches Deployment, das dem archivbasierten J2EE-Deployment ähnelt. Axis2 bringt auch zwei neue Funktionen mit, um eine hohe Verfügbarkeit zu erreichen. "Hot Deployment" dient dazu, einen Web-Service zu verteilen, während das System läuft und "Hot Update" ist eine in einer Entwicklungsumgebung als besonders nützliche Funktion zu betrachten, da es Änderungen am Web-Service erlaubt, ohne das System anzuhalten.

(2) Der komponierte Service soll mit BPEL erstellt werden und in die Orchestrations-Maschine (engl. orchestration engine) Apache ODE (Orchestration Director Engine, [Apache-ODE]) verteilt werden. Apache ODE führt Geschäftsprozesse aus, die auf dem WS-BPEL-Standard basieren und mit anderen Web-Services kommunizieren.

(3) Der erstellte BPEL-Prozess ist im Web-Browser aufzurufen, um die einzelnen Prozessabläufe zu testen.

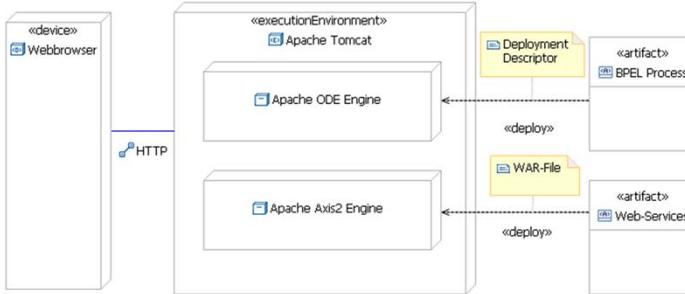
ODE	Orchestration Director Engine
WAR	Web (Application) Archive

[Apache-Axis2] Apache Axis2, <http://axis.apache.org/axis2/java/core/>, 2010

[Apache-ODE] Apache ODE, <http://ode.apache.org/>, 2011

PA WSORG-BPELPRO – Deployment

- (1) Die erstellten Artefakte sind in lokalen Ausführungsumgebungen zu verteilen
 - (1) Der BPEL-Prozess und die Web-Services werden unabhängig voneinander verteilt, um separate Änderungen bei den Artefakten zu ermöglichen



38

21.04.2012

WASA: WEB-SERVICE-KOMPOSITION

Cooperation & Management (C&M, Prof. Abeck)
Institut für Telematik, Fakultät für Informatik

- (1) Es sind die erstellten Artefakte in Ausführungsumgebungen verteilt (engl. deploy) werden. Um die Aufrufe lokal zu testen, werden lokale Ausführungsumgebungen für den BPEL-Prozess und die Web-Services benutzt.
 - (1.1) Da der BPEL-Prozess auf die beteiligten Dienste per Web-Service-Aufrufe, die durch eindeutige URL-Adressen identifiziert sind, zugreift, können wir die Web-Services in eine separate Ausführungsumgebung deployen. Dies ermöglicht eine einfache Übernehmen der Änderungen auf Seite der Web-Services, ohne dass der BPEL-Prozess unterbrochen wird.

Das Verteilungs-Diagramm (engl. deployment diagram) gibt ein Überblick über die Ausführungsumgebungen, in denen die erstellten Artefakte verteilt werden.

(Webbrowser) Läuft auf der Seite des Klienten, der den ersten Aufruf des Web-Services durch eine HTTP-Abfrage initiiert. Der Webbrowser kann auch durch einen Service ersetzt werden, der den Aufruf z.B. in einer Servicekomposition erstellt.

(Apache Tomcat) Stellt die Umgebung zur Ausführung von Java-Code auf Seite des Servers dar. In Produktionsumgebungen wird der HTTP-Server des Tomcat nicht benutzt; stattdessen wird meist ein Apache Web-Server vor den Apache Tomcat geschaltet.

(Apache ODE Engine) Ist eine freie Laufzeitumgebung für WS-BPEL und unterstützt ein "Hot Deployment" von WS-BPEL-Paketen. Apache ODE besteht aus einem internen Compiler, der WS-BPEL-Dateien nach Java kompiliert, einer Ausführungsmaschine, einer Integrationsschicht, die dazu benutzt wird, um mit externen Services kommunizieren zu können.

(Apache Axis2 Engine) Wird als Java-Servlet innerhalb des Apache Tomcat betrieben, der Web-Services für Java-Klassen anbietet.

(Deployment Descriptor) Definiert primär die Prozesse und die zu verwendenden Bindungen (engl. bindings) für Partnerverbindungen (engl. partner links), um Prozesse auszurufen oder um in BPEL auf Nachrichten zu warten. Der untenstehende Ausschnitt stellt einen Deployment Descriptor für den BPEL-Prozess "Process1" dar, der den Web-Service "Service" durch den PartnerLink "servicePL" aufruft.

```

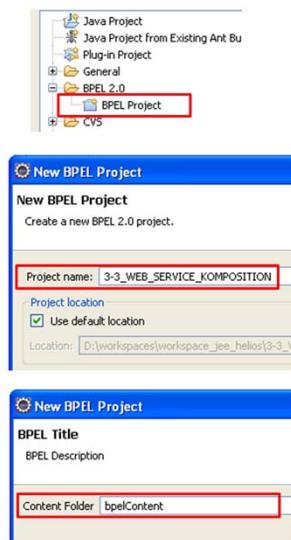
<deploy xmlns="http://www.apache.org/ode/schemas/dd/2007/03"
       xmlns:pns="http://ode/bpel/unit-test" xmlns:wns="http://ode/bpel/unit-test.wsdl">
    <process name="pns:Process1">
        <active>true</active>
        <provide partnerLink="servicePL">
            <service name="wns:Service" port="OperationPort"/>
        </provide>
    </process>
</deploy>

```

(WAR-File) Steht für Web Archive oder Web Application Archive und ist ein Dateiformat, das das Verpacken einer vollständigen Webanwendung nach der Java-Servlet-Spezifikation in Form von JAR- bzw. ZIP-Format beschreibt. Diese Datei kopieren wir in den webapps-Ordner von Tomcat, wo sie nach Start von Tomcat automatisch entpackt und deployed wird.

PA WSORG-BPELPRO – BPEL-Projekt

- (1) Erstellen eines neuen BPEL-Projektes
 - (1) Der Name des Projektes ist "3-3_WSORG-BPELPRO"
 - (2) Der "Content Folder" enthält die BPEL-Prozesse und alle referenzierten Dateien



39

21.04.2012

WASA: WEB-SERVICE-KOMPOSITION

Cooperation & Management (C&M, Prof. Abeck)
Institut für Telematik, Fakultät für Informatik

Eclipse unterstützt die Entwicklung von BPEL-Prozessen durch das BPEL Designer Project [Eclipse-BPEL-Project]. BPEL-Prozesse können dann in Eclipse erstellt, editiert und getestet werden, wobei zum Erstellen von Prozessen das BPEL Designer Tool benutzt wird [Eclipse-BPEL-Designer]. Der BPEL Designer ist ein Editor basierend auf dem Graphical Editing Framework (GEF) und hat neben der Darstellung der BPEL-Prozess in Form von Modellen die Möglichkeit den XML-Code zu validieren und auf bestimmten Fehler und Warnungen hinzuweisen. Der Prüfer funktioniert auf ein im BPEL-Designer integriertes EMF-Model, das die WS-BPEL 2.0 Spezifikation darstellt.

- (1) Erstellen Sie ein neues Projekt in Eclipse. Als Projekttype wählen Sie "BPEL 2.0"->"BPEL Project".
 - (1.1) Der Projektname folgt der Konvention "<Kurseinheitennummer>_<PA-Kurzbezeichnung>" Die "Target Runtime" und "Configuration" sollen in diesem Schritt nicht gesetzt werden und stattdessen nur die Voreinstellungen ausgewählt werden.
 - (1.2) Der Ordner, in dem der BPEL-Prozess und alle referenzierten Dateien wie XSDs gespeichert werden, kann im nächsten Schritt eingegeben werden. Der Standardwert kann hierzu verwendet werden.

GEF

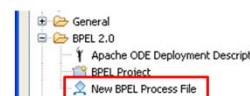
Graphical Editing Framework

[Eclipse-BPEL-Project] BPEL Project, <http://www.eclipse.org/bpel>

[Eclipse-BPEL-Designer] BPEL Designer Update Site, <http://download.eclipse.org/technology/bpel/update-site/>

PA WSORG-BPELPRO – BPEL-Prozess

- (1) Anlegen eines BPEL-Prozesses im Ordner "bpelContent"
- (2) Auswahl eines synchronen Prozesses mit dem Namen "WorkshopOrganization"
- (3) WSDL-Informationen eingeben



Process Details	
BPEL Process Name: <input type="text" value="WorkshopOrganization"/>	
Namespace: <input type="text" value="http://www.cm-tm.uni-karlsruhe.de/contract/WorkshopOrg..."/>	
Template: <input type="text" value="Synchronous BPEL Process"/>	
Generates an empty BPEL process. Only receive and reply activities are placed in the process body. The caller will block until all the steps in the process have completed. A client interface is generated.	
WSDL Details	
Service Name: <input type="text" value="WorkshopOrganization"/>	
Port Name: <input type="text" value="WorkshopOrganizationPort"/>	
Service Address: <input type="text" value="http://localhost:8080/ode/processes/WorkshopOrganization"/>	
Binding Protocol: <input type="text" value="SOAP"/>	

40

21.04.2012

WASA: WEB-SERVICE-KOMPOSITION

Cooperation & Management (C&M, Prof. Abeck)
Institut für Telematik, Fakultät für Informatik

(1) Im Ordner "bpelContent" ist ein neuer BPEL-Prozess über den Menüpunkt "New"->"Other"->"BPEL 2.0"->"New BPEL Process File" anzulegen. Der Assistent erstellt zwei neue Dateien mit den Erweiterungen ".wsdl" und ".bpel", die eine WSDL-Schnittstelle und einen BPEL-Prozess beschreiben.

(2) Beim Anlegen eines BPEL-Prozesses kann neben dem Erstellen eines leeren Projektes zwischen zwei anderen Arten von Vorlagen ausgewählt werden. Ein synchroner Prozess definiert eine Kommunikationsart, bei der der Klient den BPEL-Prozess aufruft und solange wartet, bis die Antwort zurückkommt. Bei asynchronen BPEL-Prozessen wird auf Seite des Klienten ein Aufruf durchgeführt und der Fluss unabhängig vom aufgerufenen Prozess weiterverarbeitet. Im vorliegenden Fall der Workshop-Organisation wird ein synchroner BPEL-Prozess angelegt.

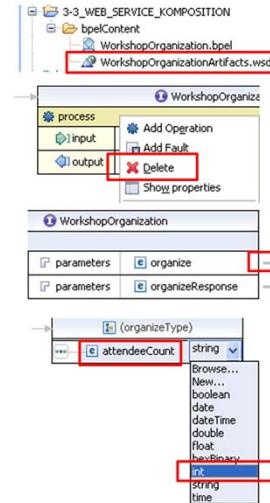
(3) Die WSDL-Informationen können im nächsten Menü eingegeben werden. Es wird folgender Namensraum gewählt: "http://www.cm-tm.uni-karlsruhe.de/contract/WorkshopOrganization". Da der resultierende BPEL-Prozess von einer Orchestrierungsmaschine ausgeführt wird, wird im Adressenfeld die URL-Adresse "http://localhost:8080/ode/processes/WorkshopOrganization" eingegeben. Für das Bindungsprotokoll (engl. binding protocol) wird das Simple Object Access Protocol (SOAP) verwendet.

SOAP

Simple Object Access Protocol

PA WSORG-BPELPRO – WSDL-Operation erstellen

- (1) Bearbeitung der erstellten WSDL-Datei mit dem WSDL-Editor
 - (1) Die automatisch erstellte Operation löschen und eine neue Operation "organize" hinzufügen
 - (2) Als Eingabeparameter soll die Teilnehmeranzahl als ganzzahlige Zahl verwendet werden
 - (3) Im Ausgabeparameter soll für der Datentyp "string" ausgewählt werden



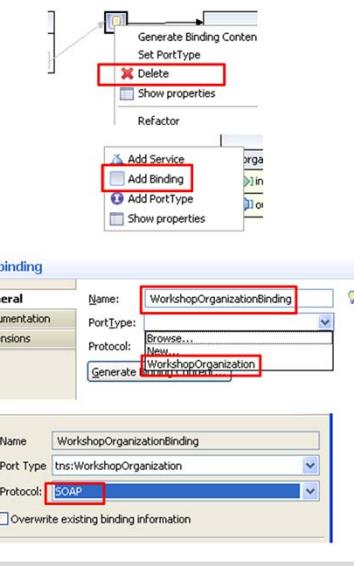
- (1) Nach Anlegen eines neuen BPEL-Prozesses liegen die BPEL- und WSDL-Artefakte in der Projektstruktur vor. Die automatisch generierte WSDL-Schnittstelle "WorkshopOrganizationArtifacts.wsdl" ist an den Geschäftsprozess anzupassen. Die vorhandene WSDL-Schnittstelle enthält eine Operation "process", die durch eine neue Operation ersetzt werden soll. Dazu wird die Datei mit dem WSDL-Editor geöffnet, der einfaches Editieren und Bearbeiten von WSDL-Dateien ermöglicht.
- (1.1) Um eine konsistente Anpassung der WSDL-Schnittstelle durchzuführen, soll die vorhandene Operation "process" gelöscht und eine neue Operation "organize" erstellt werden. Dadurch werden Nachrichtentypen mit den richtigen Namen erstellt und diese können im BPEL-Prozess verwendet werden. Eine Umbenennung der vorhandenen Operation ist aufwändig, weil einzelne Nachrichtentypen und Datentypen auch manuell umbenannt werden sollen.
- (1.2) Die Operation "organize" hat als Eingabeparameter die Anzahl der Teilnehmer "attendeeCount" in Form einer ganzzahligen Zahl.
- (1.3) Für den Rückgabeparameter soll Text zurückgegeben werden, der alle Details zum reservierten Raum und zu den ermittelten POIs enthält.

PA WSORG-BPELPRO – WSDL-Binding erstellen

- (1) Erstellen eines neuen WSDL-Binding für die neue Operation

(1) Die alte Bindung löschen und eine neue Bindung mit dem Namen "WorkshopOrganizationBinding" erstellen

(2) In der Properties-Ansicht den PortType auswählen und den Bindungsinhalt generieren

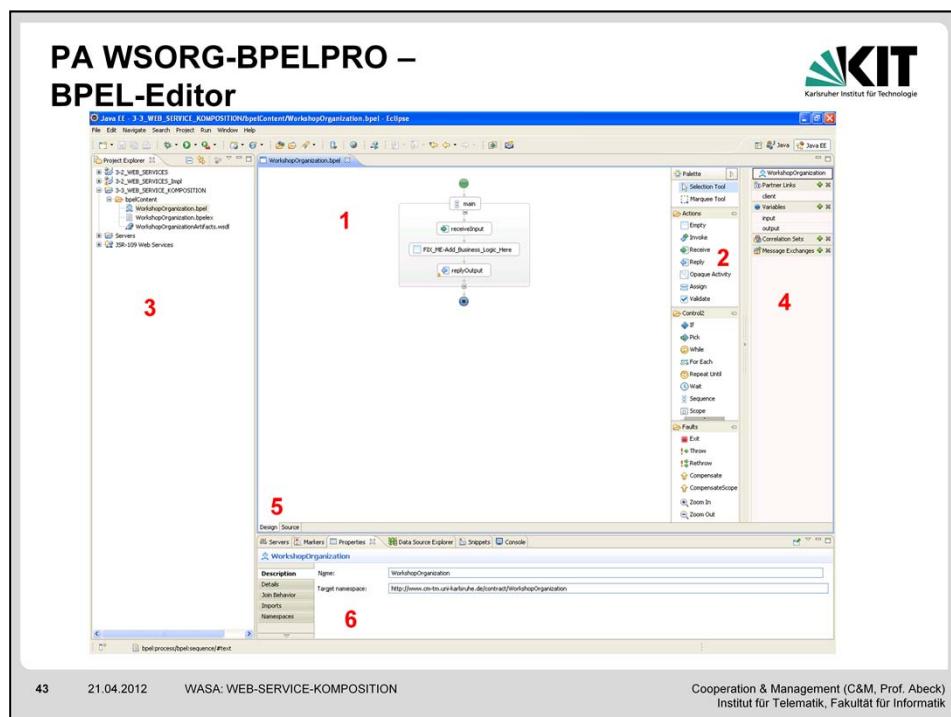


(1) Die neue Operation soll nun mit einer neuen WSDL-Bindung versehen werden.

(1.1) Die vorhandene Bindung kann gelöscht werden und mit einer neuen ersetzt werden. Die neue Bindung trägt den Namen "WorkshopOrganizationBinding".

(1.2) In der Eigenschaften (engl. property) der Bindung wird der vorhandene PortType "WorkshopOrganization" ausgewählt. Durch das Auswahlfeld "Generate Bindings Content" kann der Bindungsinhalt generiert werden und das SOAP-Protokoll ausgewählt werden.

PA WSORG-BPELPRO – BPEL-Editor

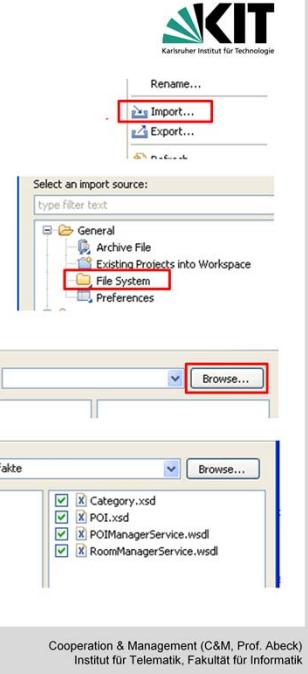


Der BPEL-Editor ist eine graphische Oberfläche, die ein einfaches Erstellen, Editieren und Bearbeiten von BPEL-Prozessen ermöglicht.

- (1) In dieser Ansicht wird der Prozess angezeigt, wobei einzelne Elemente ausgewählt und verschoben werden können.
- (2) Die Palette enthält alle BPEL-Elemente eingeordnet nach ihren Kontexten wie z.B. Fluss- oder Kontrollelemente.
- (3) Der "Project Explorer" zeigt die erstellten Projekte und ihre Strukturen an.
- (4) Neue Variablen als auch die im Prozess enthalten sind, werden in dieser Ansicht angezeigt und bearbeitet.
- (5) Hier kann zwischen der Model- und Quellcode-Ansicht ausgewählt werden. Die Quellcode-Ansicht zeigt den BPEL-Prozess in Form einer XML-Datei. Die QuellCode-Ansicht kann z.B. benutzt werden, um Warnungen und Fehlern in den einzelnen Zeilen zu identifizieren.
- (6) Die Properties-Ansicht ermöglicht die Einstellung der ausgewählten Elemente durch Änderung ihrer Eigenschaften.

PA WSORG-BPELPRO – Dateien in Eclipse-Projekt importieren

- (1) Importieren der vorhandenen XSDs und WSDL-Schnittstellen in den Ordner "bpelContent"
- (2) Überführen der zwei WSDL-Schnittstellen mit Axis2 in Code und Speichern in ein neues Projekt mit dem Namen "3-3_WSORG-BPELPRO_Impl"



44

21.04.2012

WASA: WEB-SERVICE-KOMPOSITION

Cooperation & Management (C&M, Prof. Abeck)
Institut für Telematik, Fakultät für Informatik

(1) Um in Eclipse weitere Datentypen und WSDL-Schnittstellen benutzen zu können, sollen Artefakte mit der Import-Funktion von Eclipse importiert werden. Durch rechten Mausklick auf dem "Project Explorer" ist die Option "Import" anzuklicken und der Menüpunkt "General"->"File System" auszuwählen, um Dateien aus einem Ordner auf der Festplatte zu importieren. Den Ordner, in dem die Artefakt liegen, ist auszuwählen und alle Dateien sind zu selektieren, die dann in den aktuellen Projekt-Ordner kopiert und gespeichert werden. Dieser Praktischen Aufgabe ist ein ZIP-Archiv mit den notwendigen Artefakte beigelegt worden, das in einen Ordner entpackt und in Eclipse importiert werden soll. Das Archiv beinhaltet folgende Artefakte:

(Category.xsd) Der Datentyp "Category" in Form einer XSD-Datei.

(POI.xsd) Der Datentyp "POI" in Form einer XSD-Datei.

(POIManagerService.wsdl) Die WSDL-Schnittstelle des Web-Services "POIManagerService".

(RoomManagerService.wsdl) Die WSDL-Schnittstelle des Web-Services "RoomManagerService".

(2) In dieser Praktischen Aufgabe sollen Web-Services-Schnittstellen, die in Form von "Java Bean Skeletons" vorliegen, mit Hilfe von Axis2 gemäß dem Top-Down-Ansatz überführt werden. Das Zielprojekt trägt den um "_Impl" ergänzten Projektnamen und wird später in eine WAR-Datei exportiert.

PA WSORG-BPELPRO – Implementierung der Web-Services

- (1) Generierung des Codes mit Hilfe von Axis2 in einem dynamischen Projekt und Ergänzung der notwendigen Funktionalität
- (2) Erstellung einer Dummy-Implementierung für die zwei Web-Services
 - (1) Der "RoomManagerService" gibt einen Raum als POI mit zufälligen Werten für die Längen- und Breitengrad zurück
 - (2) Der "POIManagerService" sucht in einer Dummy-Datenbank von POIs den nächstgelegenen POI für eine Kategorie, die den minimalen Abstand zum Raum hat
- (3) Exportieren des Projekts als WAR-Datei und Kopieren in den "webapps"-Ordner von Tomcat

(1) Wie in der Praktischen Aufgabe "POIMANAGERSERVICE" beschrieben wurde, kann mit Hilfe von Axis2 Java-Quellcode aus den WSDL-Schnittstellen generiert werden. Das Zielprojekt der Konvertierung soll den Namen "3-3_WSORG-BPELPRO_Impl" tragen. Um den gesamten BPEL-Prozess und die einzelnen Web-Service-Aufrufe testen zu können, wird der generierte Code um zusätzliche Funktionalität ergänzt.

(2) Die zwei Web-Services haben eine Dummy-Implementierung, um das Ergebnis zu simulieren. Die Klassen, die die Operationen der Web-Services enthalten und erweitert werden sollen, sind im Zielprojekt im Ordner "src" enthalten:

(de.uni_karlsruhe.cm_tm.www.contract.poimanagerservice.POIManagerServiceSkeleton) Java-Klasse für den Web-Service "POIManagerService".

(de.uni_karlsruhe.cm_tm.www.contract.roommanagerservice.RoomManagerServiceSkeleton) Java-Klasse für den Web-Service RoomManagerService.

(2.1) Der Web-Service "RoomManagerService" hat die Operation "getRoom", die einen Raum als POI mit Längen- und Breitengrad zurückgibt. Um auch dynamisch Räume nach der Anzahl der Teilnehmer zu vergeben, wird definiert, dass sich jeder Raum um eine Anzahl von 10 Teilnehmern im Vergleich zum kleineren und größeren Raum unterscheidet. Als Beispiel kann Raum 1 bis zu 10 Personen enthalten und Raum 2 bis zu 20 usw.

(2.2) Der Web-Service "POIManagerService" erstellt eine Dummy-Datenbank, die POIs sortiert nach Kategorien enthält, und sucht beim Aufruf der Operation "findNextPOIByCategory" für die übergebene Kategorie und den Raum den nächstgelegenen POI.

(3) Das erstellte dynamische Web-Projekt ist in eine WAR-Datei zu exportieren und in den webapps-Ordner von Tomcat zu kopieren, um die Web-Services beim Start des ODE Servers zu deployen. Bitte beachten Sie dass der Name der WAR-Datei in Kleinbuchstaben eingegeben werden soll, sonst können die Web-Service-Aufrufe aus dem BPEL-Prozess nicht richtig ausgeführt werden. Die WAR-Datei hat dann den Namen "3-3_WSORG-BPELPRO_impl.war".

PA WSORG-BPELPRO – Web-Service-Operation "getRoom"

```
1. public GetRoomResponse getRoom(GetRoom getRoom){  
2.     POI poi = new POI();  
3.     poi.setId(((getRoom.getAttendeeCount() / 10) + 1));  
4.     poi.setCategory("Seminarraum");  
5.     poi.setLatitude((float)(Math.random() * 100) % 49);  
6.     poi.setLongitude((float)(Math.random() * 100) % 80);  
7.     poi.setName("Raum " + poi.getId());  
8.     GetRoomResponse getRoomResponse = new GetRoomResponse();  
9.     getRoomResponse.setOut(poi);  
10.    return getRoomResponse;  
11.}  
12.}
```

RoomManagerServiceSkeleton.java

(1. public ...) Die Methode stellt die Implementierung des Web-Services beim Aufruf der Operation "getRoom" aus dem BPEL-Prozess dar. Die Nachrichtentypen für die Eingabe- und Ausgabeparameter aus der WSDL-Datei wurden in den Klassen "GetRoom" und "GetRoomResponse" umgewandelt.

(2. POI poi ...) Hier wird ein POI-Objekt erstellt und in den nachfolgenden Befehlen mit notwendigen Informationen anhand der Setter-Methoden gefüllt.

(3. poi.setID ...) Die ID wird aus der Anzahl der Teilnehmer ermittelt, was eine dynamische Zuweisung von Räumen und als auch eine Überprüfung der zurückgegebenen POI ermöglicht.

(4. poi.SetCategory) Der Workshop findet in einem POI der Kategorie "Seminarraum" statt.

(5. poi.setLatitude ..., 6. poi.setLongitude ...) Der POI (hier der Seminarraum) erhält zufällige Werte für den Längen- und Breitengrad.

(7. poi.setName ...) Hierdurch erhält der POI einen eindeutigen Namen.

(8. GetRommResponse ...) Erstellung des Objekts für den Rückgabeparameter erstellt.

(9. getRoomResponse ...) Zuweisung des POIs zum Objekt mit Hilfe einer Setter-Methode.

PA WSORP-BPELPRO – Web-Service-Operation "findNextPOIByCategory"



```
1. public FindNextPOIByCategoryResponse findNextPOIByCategory(FindNextPOIByCategory
                                                               findNextPOIByCategory){
2.     String category = findNextPOIByCategory.getCategory();
3.     POI room = findNextPOIByCategory.getRoom();
4.     Map<String, List<POI>> poiDB = buildPOIDatabase();
5.     POI poi = findPOIByCategoryInDB(poiDB, category, room);
6.     FindNextPOIByCategoryResponse findNextPOIByCategoryResponse = new
                                                               FindnextPOIByCategoryResponse();
7.     findNextPOIByCategoryResponse.setOut(poi);
8.
9.     return findNextPOIByCategoryResponse;
10.}

11.private Map<String, List<POI>> buildPOIDatabase(){ ... }

12.private POI findPOIByCategoryInDB(Map<String, List<POI>> poiDB, String categoryUser, POI
room){ ... }

13.private POI findNextPOI(List<POI> pois, POI room){ ... }

14.private double distanceTwoPOIs(POI poi1, POI poi2){ ... }
```

POIManagerServiceSkeleton.java

47 21.04.2012

WASA: WEB-SERVICE-KOMPOSITION

Cooperation & Management (C&M, Prof. Abeck)
Institut für Telematik, Fakultät für Informatik

(1. public ...) Die Operation "findNextPOIByCategory" führt eine Reihe von Operationen durch, um den nächstgelegenen POI für eine bestimmte Kategorie zurückzugeben.

(2. String category ..., 3. POI room ...) Aus dem Eingabeparameter "findNextPOIByCategory" werden die Kategorie und der Raum in entsprechende Variablen übernommen.

(4. Map<...>) Die Methode "buildPOIDatabase" erstellt eine Dummy-Datenbank bestehend aus einer Reihe von POIs sortiert nach Kategorien.

(5.) Durch die Methode "findPOIByCategoryInDB" wird nach dem nächstgelegenen POI für die Kategorie gesucht.

(6. FindNextPOIByCategoryResponse ... - 8. return ...) Der Rückgabeparameter wird erstellt und mit der ermittelten POI zurückgegeben.

(11.) Die Methode "buildPOIDatabase" legt die Anzahl der POIs pro Kategorie fest. In dem untenstehenden Code gibt es auf dem Campus zwei Menschen, drei Cafeterien, vier Bibliotheken und 5 Parkplätze. Für jede Kategorie werden dann die einzelnen POIs mit zufälligen Zahlen für den Längen- und Breitengrad erstellt. Die Methode zum Erstellen der Datenbank sieht folgendermaßen aus:

```
private Map<String, List<POI>> buildPOIDatabase(){
    Map<String, List<POI>> mapPOI = new LinkedHashMap<String, List<POI>>();
    Map<String, Integer> mapCategories = new LinkedHashMap<String, Integer>();
    mapCategories.put("Menschen", 2);
    mapCategories.put("Cafeterien", 3);
    mapCategories.put("Bibliotheken", 4);
    mapCategories.put("Parkplätze", 5);

    for(String category : mapCategories.keySet()){
        List<POI> pois = new ArrayList<POI>();
        for(int id = 0 ; id < mapCategories.get(category); id++){
            POI poi = new POI();
            poi.setId(id + 1);
            poi.setCategory(category);
            poi.setLatitude((float)(Math.random() * 100) % 49);
            poi.setLongitude((float)(Math.random() * 100) % 80);
            poi.setName(category.toCharArray()[0] + " " + poi.getId());

            pois.add(poi);
        }
        mapPOI.put(category, pois);
    }
    return mapPOI;
}
```

```

    }

    mapPOI.put(category, pois);
}

return mapPOI;
}

```

(12.) Die Methode "findPOIByCategoryInDB" erhält als Parameter die Datenbank, die Kategorie und den Raum. Die Methode erstellt einen leeren POI, der bei unbekannter Kategorie zurückgegeben wird. Falls die gewünschte Kategorie in der Datenbank gefunden wurde, wird das POI-Objekt mit der nächstgelegenen POI ersetzt und zurückgegeben.

```

private POI findPOIByCategoryInDB(Map<String, List<POI>> poiDB, String categoryUser, POI room){
    POI poi = new POI();
    poi.setName("");
    poi.setId(0);
    poi.setCategory("");
    poi.setLatitude(0.0f);
    poi.setLongitude(0.0f);

    for(String categoryDB : poiDB.keySet()){
        if(categoryDB.toLowerCase().equals(categoryUser.toLowerCase())){
            poi = findNextPOI(poiDB.get(categoryDB), room);
            break;
        }
    }

    return poi;
}

```

(13.) Die nächstgelegene POI wird mit Hilfe der Funktion "findNextPOI" ermittelt, wobei diese Methode alle POIs der Kategorie und den Raum im Eingabeparameter übergeben bekommt. Diese Methode berechnet den Abstand zwischen dem Raum und jeder POI und gibt die POI mit dem kleinsten Abstand zurück.

```

private POI findNextPOI(List<POI> pois, POI room){
    POI min = pois.get(0);
    for(int i = 1 ; i < pois.size() ; i++){
        POI poi = pois.get(i);
        if(distanceTwoPOIs(poi,room) <= distanceTwoPOIs(min,room)){
            min = poi;
        }
    }

    return min;
}

```

(14.) Der Abstand zwischen zwei POIs ist mit Hilfe der Methode "distanceTwoPOIs" ermittelt.

```

private double distanceTwoPOIs(POI poi1, POI poi2){
    float la1 = poi1.getLatitude();
    float la2 = poi2.getLatitude();

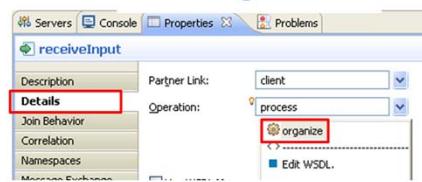
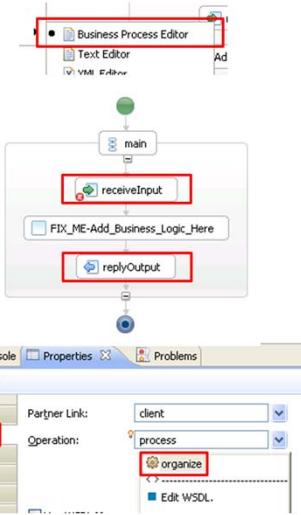
```

```
float lo1 = poi1.getLongitude();
float lo2 = poi2.getLongitude();

return Math.sqrt(Math.pow((la1 - la2), 2) + Math.pow((lo1 - lo2), 2));
}
```

PA WSORG-BPELPRO – WSDL-Operation im BPEL-Prozess

- (1) Öffnen der BPEL-Datei mit dem BPEL-Editor
- (2) Öffnen der Aktivitäten "Receive" und "Reply" in der "Properties"-Ansicht und Ergänzung der neuen Operation aus der WSDL-Schnittstelle



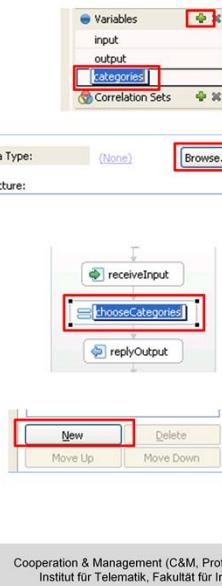
(1) Der erstellte BPEL-Prozess soll nun mit dem WSDL-Editor geöffnet und bearbeitet werden, um ihn um die neue Operation zu ergänzen. Im BPEL-Projekt von Eclipse gibt es bis jetzt keine Möglichkeit, die WSDL-Datei mit dem BPEL-Prozess automatisch zu synchronisieren. Änderungen in beiden Artefakten können nur manuell synchronisiert werden. Jeder BPEL-Prozess enthält am Anfang eine "Receive"-Aktivität, die die empfangenen Parameter in einer Variable speichert. Die "Receive"-Aktivität nimmt eine besondere Stellung im Lebenszyklus eines Geschäftsprozesses ein, da dieser durch das "createInstance"-Attribut der "Receive"-Aktivität instanziert wird. Die "Reply"-Aktivität wird im Falle einer synchronen Interaktion benutzt, um die Antwort auf eine zuvor akzeptierte Anfrage zu senden.

In den Eigenschaften-Ansicht der "Receive"- bzw. "Reply"-Aktivität können die Operation, Variablen und PartnerLinks der Kommunikationspartner ausgewählt werden.

(2) Nach der Änderung der WSDL-Schnittstelle befindet sich nun der BPEL-Prozess in einem inkonsistenten Zustand und zeigt einen Fehler bei den "Receive"- und "Reply"-Aktivität an. Nun soll der BPEL-Prozess auch angepasst und mit der WSDL-Schnittstelle synchronisiert werden. Die "Receive"- bzw. "Reply"-Aktivität ist mit der neuen Operation in der "Properties"-Ansicht anzupassen. Die Aktivität "FIX_ME-Add_Business_Logic_Here" kann nun auch gelöscht werden.

PA WSORG-BPELPRO – Kategorien als Variable in BPEL

- (1) Speicherung der Kategorien, die beim Aufruf der gelegenen POIs benutzt werden, im BPEL-Prozess in Form von XML-Literalen
- (2) Erstellung der Variablen mit dem Namen "categories" und dem Datentyp "CategoryList"
- (3) Verwendung der "Assign"-Aktivität, um XML-Literale einer Variable zuzuweisen



(1) Eine Kategorie ist im BPEL-Prozess durch ein XML-Literal des Datentyps "Category" beschrieben. Das Array "CategoryList" enthält eine Menge von Kategorien und kann dann in einem weiteren Schritt durchlaufen werden, um einzelne Kategorien beim Aufruf des Web-Services "RoomManagerService" zu übergeben.

(2) Eine Variable "categories" mit dem Datentyp "CategoryList" anlegen. In der Properties-Ansicht lassen sich durch den "browse"-Button die vorher importierten Datentypen finden. Bitte wählen Sie den Datentyp "CategoryList" aus.

(3) Das Zuweisen von XML-Literalen einer Variable ist mit der Assign-Aktivität von BPEL möglich. Hierfür wird dem Prozess eine neue "Assign"-Aktivität hinzugefügt. Das "Assign"-Element wird in der Palette markiert und zwischen der "Receive"- und "Reply"-Aktivität platziert. Um die "Assign"-Aktivität so zu konfigurieren, dass sie ein XML-Literal einer Variable zuweist, wird per rechtem Mausklick auf das "Assign"-Element dessen Kontextmenü geöffnet und hier "Show in Properties" gewählt. In der nun erscheinenden "Properties"-Ansicht wird links der Reiter "Details" geöffnet und mit "New" eine neue Kopieraktion aus einem "Fixed Value" in die Variable "categories" eingefügt, wobei folgender XML-Ausschnitt in der "From"-Seite eingegeben werden soll. Sobald mittels "Steuerung-S" der BPEL-Prozess gespeichert wird, fragt Eclipse, ob ein Initialisierer für das <assign>-Element erzeugt werden soll, was zu bestätigen ist.

```

<ns1:CategoryList
  xmlns:ns1="http://www.cm-tm.uni-karlsruhe.de/schema/Category"
  xmlns:tns="http://www.cm-tm.uni-karlsruhe.de/contract/WorkshopOrganization"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <ns1:category>
    <ns1:name>Mensen</ns1:name>
  </ns1:category>

  <ns1:category>
    <ns1:name>Cafeterien</ns1:name>
  </ns1:category>

  <ns1:category>
    <ns1:name>Bibliotheken</ns1:name>
  </ns1:category>

  <ns1:category>
    <ns1:name>Parkplätze</ns1:name>
  </ns1:category>
</ns1:CategoryList>

```

PA WSORG-BPELPRO – Import der Schemas und WSDL-Schnittstellen

- (1) Importieren der existierenden Datentypen in Form von XSDs in den BPEL-Prozess
- (2) Importieren der WSDL-Schnittstellen, um die existierenden Web-Services aufrufen zu können
- (3) Herunterladen der für die Praktische Aufgabe benötigten Artefakte und Importieren in das Projekt

The screenshot shows the Eclipse BPEL Project Explorer interface. In the center, there's a table-like view under the 'Imports' tab, listing various files with their locations and namespaces. A red box highlights the 'Imports' tab itself and the list of files. Below this, a modal dialog box titled 'Browse for a WSDL to Import' is open, showing options for 'Import Source' (Resource, File, URL, WSDL) with 'Resource' selected. Another red box highlights the 'Resource' radio button.

50

21.04.2012

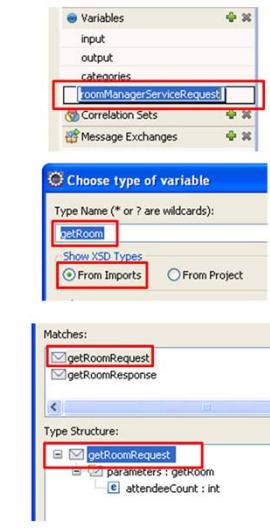
WASA: WEB-SERVICE-KOMPOSITION

Cooperation & Management (C&M, Prof. Abeck)
Institut für Telematik, Fakultät für Informatik

- (1) In dieser Praktischen Aufgabe werden existierende Datentypen benutzt. Diese können in BPEL in Form von XSD- oder WSDL-Dateien im Reiter "Imports" importiert werden. Die existierenden WSDL- und XSD-Dateien befinden sich nach dem Eclipse-Import im aktuellen Projekt und können nun auch in den BPEL-Prozess importiert werden.
- (2) Nachdem die zwei WSDL-Schnittstellen "POIManagerService.wsdl" und "RoomManagerService.wsdl" erfolgreich importiert sind, können in BPEL die Aufrufe der Web-Services erstellt werden. Diese WSDL-Schnittstellen werden vom BPEL-Editor mit "PartnerLinks" ergänzt, falls sie diese nicht beinhalten.
- (3) Die notwendigen Artefakte sind beigefügt und können heruntergeladen werden und im Projekt in den Ordner "bpelContent" gespeichert werden.

PA WSORG-BPELPRO – Eingabe- und Ausgabeparameter der Web-Services

- (1) Erstellung der Eingabe- und Rückgabeparameter in Form von Variablen, um einen Web-Service in BPEL aufrufen zu können
- (2) Auswahl der Datentypen aus den importierten WSDL-Schnittstellen
- (3) Erstellung einer "Assign"-Aktivität zur Initialisierung der Variablen



51

21.04.2012

WASA: WEB-SERVICE-KOMPOSITION

Cooperation & Management (C&M, Prof. Abeck)
Institut für Telematik, Fakultät für Informatik

(1) Bevor ein Dienstauftrag in BPEL erfolgen kann, müssen die Eingabe- und Ausgabeparameter in Form von Variablen erstellt werden. Als Beispiel wird im Web-Service "RoomManagerService" für den Eingabeparameter die Variable "roomManagerServiceRequest" und für den Rückgabeparameter die Variable "roomManagerServiceResponse" angelegt. Für den zweiten Web-Service-Aufruf werden zwei weitere Variablen mit den Namen "poiManagerServiceRequest" und "poiManagerServiceResponse" erstellt. Diese Variablen dienen zur Speicherung von Nachrichten und Daten und machen den Zustand des Prozesses zur Laufzeit aus. Variablen können auch nur für den Zustand des Prozesses benötigt werden, ohne jemals mit Partnern ausgetauscht zu werden.

(1.1) Die Datentypen bzw. Nachrichtentypen der Parameter befinden sich in den importierten WSDL-Schnittstellen und können mit Hilfe des BPEL-Editors gefunden werden. Die erstellten Variablen sollen mit den richtigen Nachrichtentypen angepasst werden. Für die Variablen "roomManagerServiceRequest" und "roomManagerServiceResponse" werden die WSDL-Nachrichtentypen "getRoomRequest" bzw. "getRoomResponse" ausgewählt. Für die Variablen des zweiten Web-Services werden "findNextPOIByCategoryRequest" und "findNextPOIByCategoryResponse" ausgewählt.

(2) Die Variablen eines Geschäftsprozesses werden durch eine "Assign"-Aktivität initialisiert. Eine Initialisierung ist die Zuweisung eines XML-Literals zu einer Variable. Dazu wird zum BPEL-Prozess zwischen der "Receive"- und "Reply"-Aktivität ein "Assign"-Element mit dem Namen "initializeVariables" hinzugefügt.

PA WSORG-BPELPRO – Web-Service-Aufruf erstellen

- (1) Erstellung einer "Invoke"-Aktivität für jeden Web-Service-Aufruf und Anpassung mit den notwendigen Parametern aus der WSDL-Schnittstelle des Web-Services
- (2) Setzen eines PartnerLink über die Eigenschaften-Ansicht der "Invoke"-Aktivität
- (3) Auswahl der Operation und der vorher erstellten Variablen für die Eingabe- und Rückgabeparameter



(1) Für den Web-Service-Aufruf wird nun nach der "Assign"-Aktivität eine "Invoke"-Aktivität hinzugefügt, die dazu dient, einen externen Web-Service aufzurufen. Das Aufrufen mittels einer "Invoke"-Aktivität kann ein synchroner Anforderungsauftrag oder eine asynchroner, einseitiger Methodenauftrag sein. BPEL nutzt die gleiche Syntax für beide Fälle mit einigen zusätzlichen Optionen für den synchronen Aufruf.

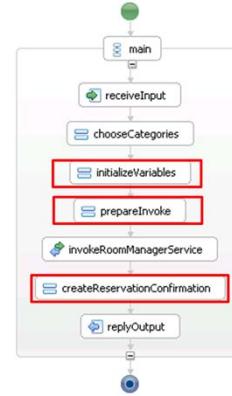
(2) In der Eigenschaften-Ansicht wird der PartnerLink gesetzt, über den der Web-Service aufgerufen wird. Ein PartnerLink modelliert in BPEL einen Service, mit dem ein Geschäftsprozess interagiert. Dabei ist anzugeben, dass ein neuer, globalen PartnerLink erstellt werden soll. Als Name ist "RoomManagerServicePL" festzulegen. Im darauffolgenden Dialog sind PortTypes im Filter zu erlauben und es ist "RoomManagerService" auszuwählen. Jeder PartnerLink wird dabei mit einem PartnerLinkType gekennzeichnet, wobei mehrere PartnerLinks durch denselben PartnerLinkType gekennzeichnet werden können. Ein PartnerLinkType bezeichnet dabei das Verhältnis des Nachrichtenaustausches zwischen zwei Services. Er spezifiziert die Rolle der Kommunikationspartner im Nachrichtenaustausches und legt fest, welcher PortType vom jeweiligen Service für den Austausch bereitgestellt wird.

Weil für den angelegten PortType noch kein PartnerLinkType existiert, legt Eclipse einen neuen an. Als Name ist "RoomManagerServicePLT" zu wählen. Die Rolle wird "RoomManagerServiceProvider" genannt und als PortType wird "RoomManagerService" gewählt. Die zweite Rolle wird leer gelassen und der Dialog wird mit "Finish" bestätigt.

(3) Die Operation kann nun in der "Properties"-Ansicht der "Invoke"-Aktivität festgelegt werden. Dazu wird der vorher erstellte PartnerLink "RoomManagerServicePL" gewählt und der "Invoke"-Aktivität werden die Variablen zugeordnet. Dazu dwir die Option "Use WSDL Message Parts Mapping" deaktiviert und die entsprechenden Variablen "roomManagerServiceRequest" und "roomManagerServiceResponse" werden aus den Dropdowns ausgewählt.

PA WSORG-BPELPRO – Web-Service-Aufruf anpassen

- (1) Initialisierung der Variablen vor dem Aufruf von externen Services durch die "Invoke"-Aktivität
 - (1) Die Initialisierung der Variablen mit XML-Literalen erfolgt durch eine "Assign"-Aktivität
- (2) Erstellung einer "Assign"-Aktivität zur Vorbereitung des "Invokes", um gespeicherte Informationen beim Aufruf eines anderen Services zu übergeben
- (3) Bearbeitung des Ergebnisses in einer "Assign"-Aktivität und Speicherung als Text nach dem Aufruf des Web-Services durch die "Invoke"-Aktivität



53

21.04.2012

WASA: WEB-SERVICE-KOMPOSITION

Cooperation & Management (C&M, Prof. Abeck)
Institut für Telematik, Fakultät für Informatik

- (1) Die im BPEL-Prozess erstellten Variablen sollen initialisiert werden, bevor sie in einer "Invoke"-Aktivität benutzt werden können.
- (1.1) Eine "Assign"-Aktivität mit dem Namen "initializeVariables" enthält alle Initialisierungen der Variablen. Hier sollen drei Kopieraktionen erstellt werden, wobei auf der "From"-Seite einen festen Wert in Form eines XML-Literals eingegeben wird. Für die folgenden Variablen sollen folgende XML-Literale eingegeben werden:

(output)

```

<tns:organizeResponse
xmlns:tns="http://www.cm-tm.uni-karlsruhe.de/contract/WorkshopOrganization"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <tns:out/>
</tns:organizeResponse>
  
```

(roomManagerServiceRequest)

```

<tns:getRoom
xmlns:tns="http://www.cm-tm.uni-karlsruhe.de/contract/RoomManagerService"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <tns:attendeeCount>0</tns:attendeeCount>
</tns:getRoom>
  
```

- (2) Beim Aufruf von einem externen Dienst können Informationen aus den existierenden Variablen übergeben werden. In unserem Fall ist die Anzahl der Teilnehmer aus dem Prozess zum externen Dienst zu übergeben. Die Übergabe findet in einer "Assign"-Aktivität statt, wobei eine neue Kopieraktion erstellt werden soll. Dazu wird eine Kopieraktion erstellt, die den Inhalt des Elements "input.parameters/attendeeCount" in die Variable "roomManagerServiceRequest.parameters/attendeeCount" kopiert.

- (3) Nach dem Service-Aufruf werden Ergebnisse mittels der Assign-Aktivität "createReservationConfirmation" zusammengestellt. Das Ergebnis des Web-Service-Aufrufs ist einer String-Variablen angehängt. Für die "Assign"-Aktivität wird auf der "From"-Seite einen XPath-Ausdruck erstellt, wobei für das Anhängen folgender XPath-Ausdruck eingegeben wird und für die Zielvariable der Konkatenation die Variable "output.parameters/out" auf der "To"-Seite ausgewählt:

```

concat('Vortragsraum:
',bpel:getVariableData('roomManagerServiceResponse','parameters','/ns1:out/ns3:name')
,bpel:getVariableData('output','parameters','/tns:out'))
  
```

Im obigen XPath-Ausdruck wurden folgende Namensräume benutzt (falls andere Namensräume im BPEL-Prozess eingestellt sind, soll der XPath-Ausdruck angepasst werden):

```
tns: http://www.cm-tm.uni-karlsruhe.de/contract/WorkshopOrganization  
ns1: http://www.cm-tm.uni-karlsruhe.de/contract/RoomManagerService  
ns3: http://www.cm-tm.uni-karlsruhe.de/schema/Category
```

BPEL bietet eine Reihe von Typen von Ausdrücken, wie z.B. Boolesche Ausdrücke, Ausdrücke mit einer Dauer als Wert und Zuweisungen. Für die Definition von Ausdrücken wird standardmäßig XPath benutzt, um Eigenschaften von Elementen extrahieren zu können. XPath kann auch benutzt werden, um auf Daten des Prozesses zuzugreifen. Im obigen Ausdruck wird mit Hilfe der XPath-Funktion `bpel:getVariableData('output', 'parameters', '/tns:out')` auf dem Inhalt der Variable "ouput" und das innere Element "out" gelesen.

XPath XML Path Language

[W3C-XPath-2.0] W3C, XML Path Language (XPath) 2.0, <http://www.w3.org/TR/xpath20/>

PA WSORG-BPELPRO – Web-Service POIManagerService aufrufen

- (1) Erstellung einer "ForEach"-Schleife, die das Array der Kategorien durchläuft
 - (1) Eine Zählvariable mit Startwert und dynamischem Endwert erstellen
 - (2) Das Ergebnis ist bei jedem Durchlauf durch eine "Assign"-Aktivität an den Ausgabeparameter des BPEL-Prozesses anzuhängen



(1) BPEL stellt Kontrollfluss-Elemente bereit, um die verschiedenen Abläufe eines Geschäftsprozesses darzustellen. Diese strukturierende Elemente legen fest, wie eine Menge von Aktivitäten ausgeführt wird. Sie können rekursiv verwendet, beliebig verschachtelt und kombiniert werden. In dieser praktischen Aufgabe wird eine ForEach-Schleife erstellt, die das Array der Kategorien durchläuft. Im Schleifenrumpf können mehrere Aktivität innerhalb einer "Sequence"-Aktivität durchgeführt werden. Eine "Sequence"-Aktivität gilt als beendet, wenn die letzte Aktivität innerhalb der Sequenz beendet ist.

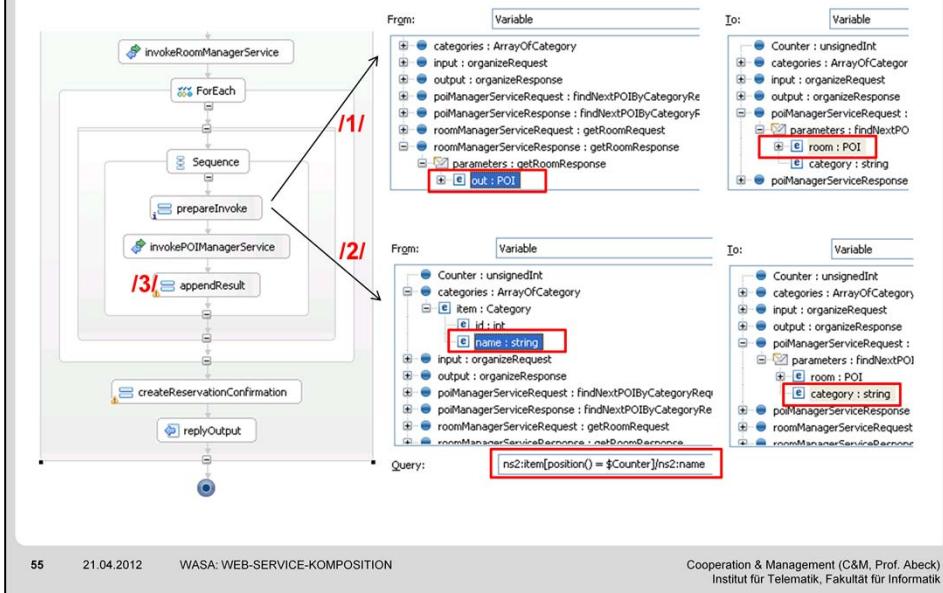
(1.1) Eine "ForEach"-Schleife definiert eine Variable, um die Anzahl der Durchläufe zu begrenzen. Im vorliegenden Fall besteht eine dynamische Anzahl an Durchläufen, die von der Anzahl der Kategorien im Array abhängig ist. Die Größe des Arrays kann mittels der XPath-Funktion "count" ermittelt werden und als Endwert der Counter-Variable zugewiesen werden.

(1.2) Die Zählvariable erhöht sich bei jedem Durchlauf und dadurch können nun einzelne Kategorien durch Indizierung des Arrays anhand der Counter-Variable einem "Invoke"-Element zugewiesen werden. Das Ergebnis des Web-Services-Aufrufes ist dann an den Ausgabeparameter des gesamten Prozesses anzuhängen. Der folgende Ausdruck zeigt eine Indizierung eines Arrays in einem BPEL-Prozess:

```
<! [CDATA[ns1:item[position()=$Counter]/ns1:name ]]>
```

Hier wird das Array "item" mit der Zählvariable "Counter" indiziert. In Apache ODE soll die Funktion "position" benutzt werden.

PA WSORG-BPELPRO – Aufruf des Web-Services "POIManagerService"



Da der Eingabeparameter beim zweiten Aufruf auch initialisiert werden soll, wird in der "Assign"-Aktivität "initializeVariables" eine Kopieraktion eingefügt, die dem Eingabeparameter "poiManagerServiceRequest" einen festen Wert in Form eines XML-Literals zuweist. Das XML-Literal lautet

(poiManagerServiceRequest)

```
<tns:findNextPOIByCategory
  xmlns:tns="http://www.cm-tm.uni-karlsruhe.de/contract/POIManagerService"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <tns:room></tns:room>
  <tns:category></tns:category>
</tns:findNextPOIByCategory>
```

/1/ Der gefundene Raum soll auch durch eine zweite Kopieraktion an den Eingabeparameter übergeben werden. Der Raum ist als POI im Ausgabeparameter des ersten Web-Service-Aufrufes enthalten.

/2/ Zur Vorbereitung des zweiten Web-Services wird als erstes die aktuelle Kategorie übergeben. Dies geschieht durch eine Kopieraktion, wobei das Array nun als Index den Ausdruck "position() = \$Counter" verwendet.

/3/ Bei jedem Durchlauf liefert der Aufruf des zweiten Web-Services die nächstgelegene POI für die übergebene Kategorie. Das Ergebnis wird an den Ausgabeparameter des gesamten Prozesses angehängt. Der folgende XPath-Ausdruck führt eine Konkatenation in Form einer String-Operation aus:

(output.parameters/out)

```
concat(bpel:getVariableData('output','parameters','/tns:out'),' Kategorie:
',bpel:getVariableData('poiManagerServiceResponse','parameters','/ns2:out/ns4
:
    c      a      t      e      g      o      r      y      '      )      ,
      '      n      a      m      e      '      )      ,      '
```

```
@( ' ,bpel:getVariableData('poiManagerServiceResponse' , 'parameters' , '/ns2:  
out/ns4:latitude') , ' ,bpel:getVariableData('poiManagerServiceResponse' ,  
'parameters' , '/ns2:out/ns4:longitude')) )'
```

Im obigen XPath-Ausdruck wurden folgende Namensräume benutzt. Falls andere Namensräume im BPEL-Prozess eingestellt sind, soll der XPath-Ausdruck angepasst werden :

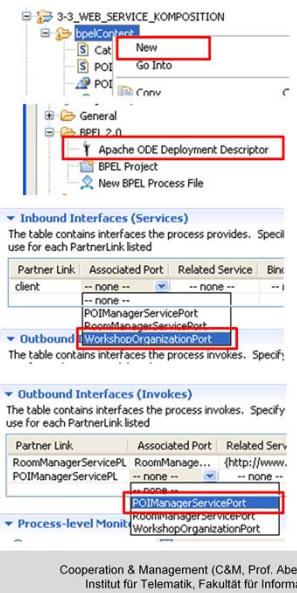
tns: http://www.cm-tm.uni-karlsruhe.de/contract/WorkshopOrganization
ns2: http://www.cm-tm.uni-karlsruhe.de/contract/POIManagerService
ns4: http://www.cm-tm.uni-karlsruhe.de/schema/POI

Der XPath-Ausdruck gibt als Beispiel folgende Ausgabe im Browser aus, wobei den fettgedruckten Text dynamisch generiert werden.

Kategorie: **Mensen M1 @(1,2345, 1,23456)**

PA WSORG-BPELPRO – Erstellen des Deployments für den BPEL-Prozess

- (1) Das Deployment des BPEL-Prozesses wird durch einen Deployment Descriptor beschrieben
 - (1) Neue Datei erstellen und mit dem "ODE Deployment Descriptor Editor" öffnen
 - (2) Angeschlossene Ports für die Prozess-Instanz und die PartnerLinks auswählen



56 21.04.2012

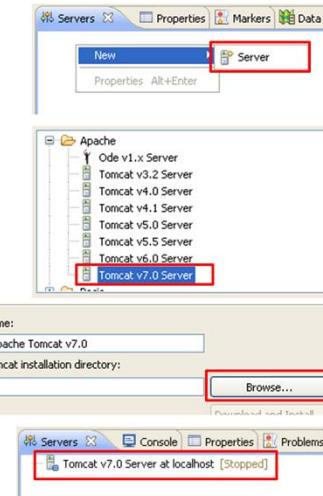
WASA: WEB-SERVICE-KOMPOSITION

Cooperation & Management (C&M, Prof. Abeck)
Institut für Telematik, Fakultät für Informatik

- (1) Nun ist der Deployment Descriptor über den Menüpunkt "New"->"Apache ODE Deployment Descriptor" zu erzeugen. Eine Datei "deploy.xml", die Informationen zum Deployment des BPEL-Prozesses enthält, wird erstellt. Der Deployment Descriptor ähnelt der Datei "web.xml", die sich im WEB-INF-Ordner einer Web-Anwendung befindet.
 - (1.1) Zum Konfigurieren des Deployment Descriptors bietet Eclipse einen Editor an. Die Datei ist durch rechten Mausklick auf "deploy.xml" und über den Menüpunkt "Open With"->"ODE Deployment Descriptor Editor" zu öffnen.
 - (1.2) Im Editor wird unter "Inbound Interfaces (Services)" für den PartnerLink "client" des BPEL-Prozesses der Port "WorkshopOrganisationPort" angegeben. Für die beiden PartnerLinks der Services "RoomManagerServicePL" und "POIManagerServicePL" sind die entsprechenden Ports auszuwählen.

PA WSORG-BPELPRO – Apache Tomcat Server in Eclipse

- (1) Herunterladen und Entpacken des Apache Tomcat
- (2) Auswählen eines Servertyp und Einrichten eines Servers in Eclipse



(1) Um Tomcat in Eclipse einbinden zu können, soll eine Version heruntergeladen und in einen Ordner entpackt werden. Folgende Version von Tomcat wurde ausgewählt: "<http://apache.mirror.digionline.de/tomcat/tomcat-7/v7.0.16/bin/apache-tomcat-7.0.16.zip>". Nach dem Herunterladen kann die Datei entpackt werden.

(2) Nachdem der Apache Tomcat installiert wurde, soll Eclipse die installierte Version als Server einrichten. Unter "Windows"->"Show View"->"Servers" ist die Ansicht "Servers" auswählen. In der Ansicht "Servers" mit dem rechten Mausklick durch "New"->"Server" einen neuen Server anlegen. Eine Reihe von Servern wie z.B. IBM WebSphere und JBoss werden von Eclipse unterstützt. In diesem Fall wird der frei verfügbare und weit bekannten Apache Tomcat Server ausgewählt. Ein Fenster zum Konfigurieren von Tomcat erscheint, in dem der Pfad zum Tomcat-Installationsordner einzugeben ist.

PA WSOR-BPELPRO – BPEL-Ausführungsumgebung Apache ODE

- (1) Die "Orchestration Director Engine" (ODE) der Apache Group dient zur Ausführung von WS-BPEL 2.0 standardisierten Geschäftsprozessen
 - (1) Apache ODE ist ein Open-Source-Projekt (Java)
 - (2) Benötigt Tomcat als Web-Server
 - (3) Integration in Eclipse in Verbindung mit dem Eclipse BPEL-Designer Plugin möglich
- (2) Schritte zur Erstellung eines BPEL-Prozesses
 - (1) Erzeugen eines neuen BPEL-Projekts
 - (2) Festlegen des Typs des zu erstellenden BPEL-Prozesses
 - (3) Entwurf des BPEL-Prozesses mit dem BPEL Designer
 - (4) Erstellen des konkreten Teils der WSDL-Beschreibung
 - (5) Erstellen eines ODE Deployment Descriptor
 - (6) Deployment auf dem Server
 - (7) Aufruf des Dienstes zur Ausführung des BPEL-Prozesses

(1) Unter "http://ode.apache.org/" wird die Ausführungsumgebung wie folgt beschrieben: "Apache ODE (Orchestration Director Engine) executes business processes written following the WS-BPEL standard. It talks to web services, sending and receiving messages, handling data manipulation and error recovery as described by your process definition. It supports both long and short living process executions to orchestrate all the services that are part of your application".

(1.1) Unter [ODE_Mail] befinden sich Mailing-Listen zum Projekt. Der Code kann mit Subversion (unter http://svn.apache.org/repos/asf/ode/branches/APACHE_ODE_1.X) abgerufen werden.

(1.2) Durch Kopieren der ODE WAR-Distribution (Web ARchive) in das "webapps"-Verzeichnis des installierten Tomcat [C&M-ODE:5/Einbindung von Apache Tomcat und Apache ODE]

(1.3) Apache ODE wird als Server in Eclipse integriert. Geschäftsprozesse können mit dem Eclipse BPEL-Designer Plugin definiert und direkt in Eclipse auf dem ODE-Server deployed werden.

Um Apache ODE und das BPEL-Plugin in Eclipse nutzen zu können, muss zuerst die Eclipse WTP (Web Tools Platform) erfolgreich installiert werden. Als Version der Eclipse-Plattform wird die Eclipse IDE 3.4 benötigt.

(2) In [C&M-ODE] wird der Erstellungsprozess mit ausführlichen Bildschirmausschnitten anhand eines einfachen "Hello World"-BPEL-Prozesses gezeigt.

(2.1) Neben der Auswahl des Projekttyps ("BPEL2.0" und "BPEL2.0 Facet") und des Projektnamens ist die Ziel-Laufzeitumgebung (Target Runtime, "Apace Ode 1.x Runtime") anzugeben.

(2.2) Hier wird "Synchronous BPSEL Process" gewählt, woraufhin eine zunächst leere BPEL-Datei ("name.bpel") vom Plug-in erzeugt wird.

(2.3) Mittels Drag-and-Drop lässt sich der Prozess um entsprechende BPEL-Elemente (z.B. Assign, Invoke, Receive, Reply, Opaque Activity) ergänzen.

(2.4) Das betrifft die Angabe des "Port Type" und des "Binding" an das zu nutzende Protokoll.

(2.5) Das Ergebnis ist eine Datei "deploy.xml", in der im Wesentlichen die implementierungsbezogenen Informationen (konkreter Teil von WSDL) enthalten sind.

(2.6) Im "Server View" wird das erstellte Projekt auf den Server ausgeliefert und der Server wird gestartet.

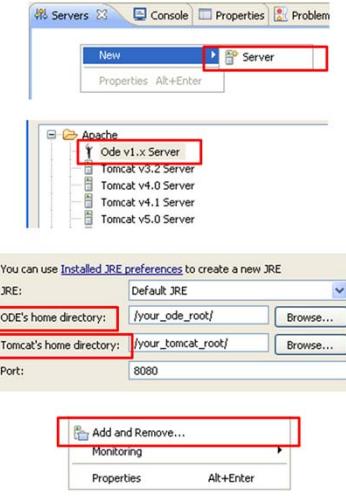
(2.7) Der Test des BPEL-Prozesses erfolgt in Eclipse mittels des "Webservice Explorer".

BPEL Business Process Execution Language

WTP Web Tools Platform

PA WSORG-BPELPRO – Einrichten von Apache ODE in Eclipse

- (1) Kopieren der WAR-Datei der Web-Services und Apache ODE WAR-Datei in den webapps-Ordner von Tomcat
- (2) Anlegen eines neuen ODE-Servers und Konfigurieren einer Tomcat-Instanz mit der Apache ODE-WAR-Datei
- (3) Hinzufügen des Projekts im Server und Starten des Servers



(1) Die Web-Services sind in einer WAR-Datei enthalten. Diese Datei kann nun nach Installation von Tomcat in den "webapps"-Ordner von Tomcat kopiert werden. Parallel wird eine Apache-ODE-Version in Form einer WAR-Datei unter der URL-Adresse "http://www.apache.org/dyn/closer.cgi/apache-ode-war-1.3.5.zip" heruntergeladen. Die WAR-Datei ist aus dem ZIP-Archiv zu entpacken und in den "webapps"-Ordner zu kopieren. Um die nächsten Schritte durchführen zu können, ist es erforderlich, einen Start des Tomcat-Servers durchzuführen. Durch diesen Start werden einzelne Ordner aus den WAR-Dateien erstellt, die zum Einrichten des Apache ODE Servers in Eclipse eingegeben werden sollen.

(2) In Eclipse wird nun ein Apache-ODE-Server eingerichtet. Dazu wird der Servertyp "Ode v1.x" ausgewählt. In den Einstellungen für das ODE-Heimatverzeichnis (ODE's home directory) wird der Pfad zum Ordner "webapps\ode" und für das Tomcat-Heimatverzeichnis (Tomcat's home directory) der Pfad zum Tomcat eingegeben.

(3) Durch rechten Mausklick ist auf dem erstellten Server und über den Menüpunkt "Add and Remove" das aktuelle Projekt "3-3_WEB_SERVICE_KOMPOSITION" hinzuzufügen und der Server ist zu starten. Falls beim Start des ODE-Servers die untenstehende Fehlermeldung erscheint, öffnet sich durch doppelten Mausklick auf dem Server-Eintrag in der Server-Ansicht die Konfiguration des ODE-Servers. In der Option "Open launch configuration" und im Tab "Classpath" ist die Datei "tomcat-juli.jar" aus dem bin-Ordner im Tomcat-Verzeichnis als externe JAR-Datei zu importieren.

```
java.lang.NoClassDefFoundError: org/apache/juli/logging/LogFactory
at org.apache.catalina.startup.Bootstrap.<clinit>(Bootstrap.java:60)
Caused by: java.lang.ClassNotFoundException:
org.apache.juli.logging.LogFactory
at java.net.URLClassLoader$1.run(Unknown Source)
at java.security.AccessController.doPrivileged(Native Method)
at java.net.URLClassLoader.findClass(Unknown Source)
at java.lang.ClassLoader.loadClass(Unknown Source)
at sun.misc.Launcher$AppClassLoader.loadClass(Unknown Source)
at java.lang.ClassLoader.loadClass(Unknown Source)
... 1 more
```

PA WSORG-BPELPRO – Aufruf des BPEL-Prozesses im Browser

The screenshot shows two separate browser windows side-by-side.

Left Window (Step 1): The URL is <http://localhost:8080/ode/processes/WorkshopOrganization?wsdl>. The page displays the WSDL code for the 'organize' operation. A red box highlights the line: `<element name="attendeeCount" type="xsd:int"/>`.

Right Window (Step 2): The URL is <http://localhost:8080/ode/processes/WorkshopOrganization/organize?attendeeCount=20>. The page displays the response from the BPEL process. A red box highlights the output: "Vortragssaal Raum 3 Kategorie: Mensen M 2 @ (44.582542, 37.235363) Kategorie: Cafeterien C 3 @ (29.436102, 55.122963) Kategorie: Bibliotheken B 4 @ (19.813179, 75.45811) Kategorie: Parkplätze P 5 @ (19.485596, 61.35345)".

60

21.04.2012

WASA: WEB-SERVICE-KOMPOSITION

Cooperation & Management (C&M, Prof. Abeck)
Institut für Telematik, Fakultät für Informatik

/1/ Die WSDL-Schnittstelle kann im Browser unter der URL-Adresse: "<http://localhost:8080/ode/processes/WorkshopOrganization?wsdl>" aufgerufen werden.

/2/ Als Testaufruf wir im Browser folgende URL-Adresse eingegeben:

"<http://localhost:8080/ode/processes/WorkshopOrganization/organize?attendeeCount=20>".

Hierdurch ist festgelegt, dass 20 Personen am Workshop teilnehmen. Es wird "Raum 3" für das Treffen vorgeschlagen und reserviert. Die nächstgelegenen POIs für die ausgewählten Kategorien sind aufgelistet, wobei die Position in Form von Breiten- und Längengrad für jede POI angezeigt ist. Bei erneutem Aufruf des BPEL-Prozesses lassen sich neue Positionen für die nächstgelegenen POIs ermitteln.