2013/14

KLUNE Alexander, SCHUSCHNIG Tobias

TGM

Aufgabe 02

SOA

Inhalt

[Aufgabenstellung 2](#_Toc371761735)

[Aufgabenaufteilung 4](#_Toc371761736)

[Zeiteinteilung 4](#_Toc371761737)

[Geschätzte Zeit 4](#_Toc371761738)

[Tatsächliche Zeit 5](#_Toc371761739)

[Designüberlegung 6](#_Toc371761740)

[Datenmodell 7](#_Toc371761741)

[Suche 7](#_Toc371761742)

[Speichern 7](#_Toc371761743)

[Laden 8](#_Toc371761744)

[Hinzufügen 8](#_Toc371761745)

[Erstellen eines Web Services 9](#_Toc371761746)

[Erstellen eines Web Service Clients 11](#_Toc371761747)

[Axis2 12](#_Toc371761748)

[Resultate 13](#_Toc371761749)

[Niederlagen 13](#_Toc371761750)

[GIT log 14](#_Toc371761751)

[Quellen 15](#_Toc371761752)

# Aufgabenstellung

Das neu eröffnete Unternehmen iKnow Systems ist spezialisiert auf Knowledgemanagement und bietet seinen Klienten die Möglichkeiten Daten und Informationen jeglicher Art (ähnlich wikipedia) in eine Wissensbasis einzupflegen und anschließend in der zentralen Wissensbasis nach Informationen zu suchen.  
  
Folgendes ist im Rahmen der Aufgabenstellung verlangt:

* Stellen Sie ein Datenmodell für die Wissensbasis (so einfach wiemöglich) auf.   
  [1 Punkte]
* Richten Sie 2 Services ein, implementieren und providen Sie diese:  
  - sucheEintrag  
  in: suchwort  
  out: List  
  [3 Punkte]  
    
  - erstelleEintrag  
  in: thema, content  
  out: success?  
  [3 Punkte]
* Implementieren Sie weiters einen Client, der Ihre Services verwendet.(Commandline/UI) [3 Punkte]
* Dokumentieren Sie im weiteren Verlauf den Datentransfer mit SOAP.  
  [2 Punkte]
* Der Einsatz eines Enterprise Service Bus (eg. openESB)  
  [4 Punkte]

Für die Umsetzung relevante Verweise:  
  
Apache Webservices Project  
<http://ws.apache.org/>  
  
Apache Axis  
<http://ws.apache.org/axis2/>  
  
JAX-WS Web Services and Axis2  
<http://axis.apache.org/axis2/java/core/docs/jaxws-guide.html>  
<http://www.ibm.com/developerworks/java/library/j-jws8/index.html>  
  
--------------  
Gruppengröße: 2  
Abgabedokument:   
Es sind alle allgemeinen Vorgaben zu erfüllen, sowie folgende spezielle Punkte  
- Datenmodell  
- Dokumentation der Services  
- Dokumentation der Service-Clients

Anmerkung: (T.Micheler, 2013-10-25)  
Bei der Abnahme DIESER Übung durch T.Micheler wird die Test-Coverage nicht berücksichtigt. Die Abnahme & der Test erfolgt im Unterricht!

# Aufgabenaufteilung

|  |  |  |
| --- | --- | --- |
| Tätigkeit | Klune | Schuschnig |
| Einlesen | X | X |
| Erstellen der Server Klasse |  | X |
| Erstellen der Client Klasse |  | X |
| GUI |  | X |
| Erstellen von Web Service | X |  |
| Erstellen von Web Service Client | X |  |
| Protokoll | X | X |
| Summe | 4 | 5 |

# Zeiteinteilung

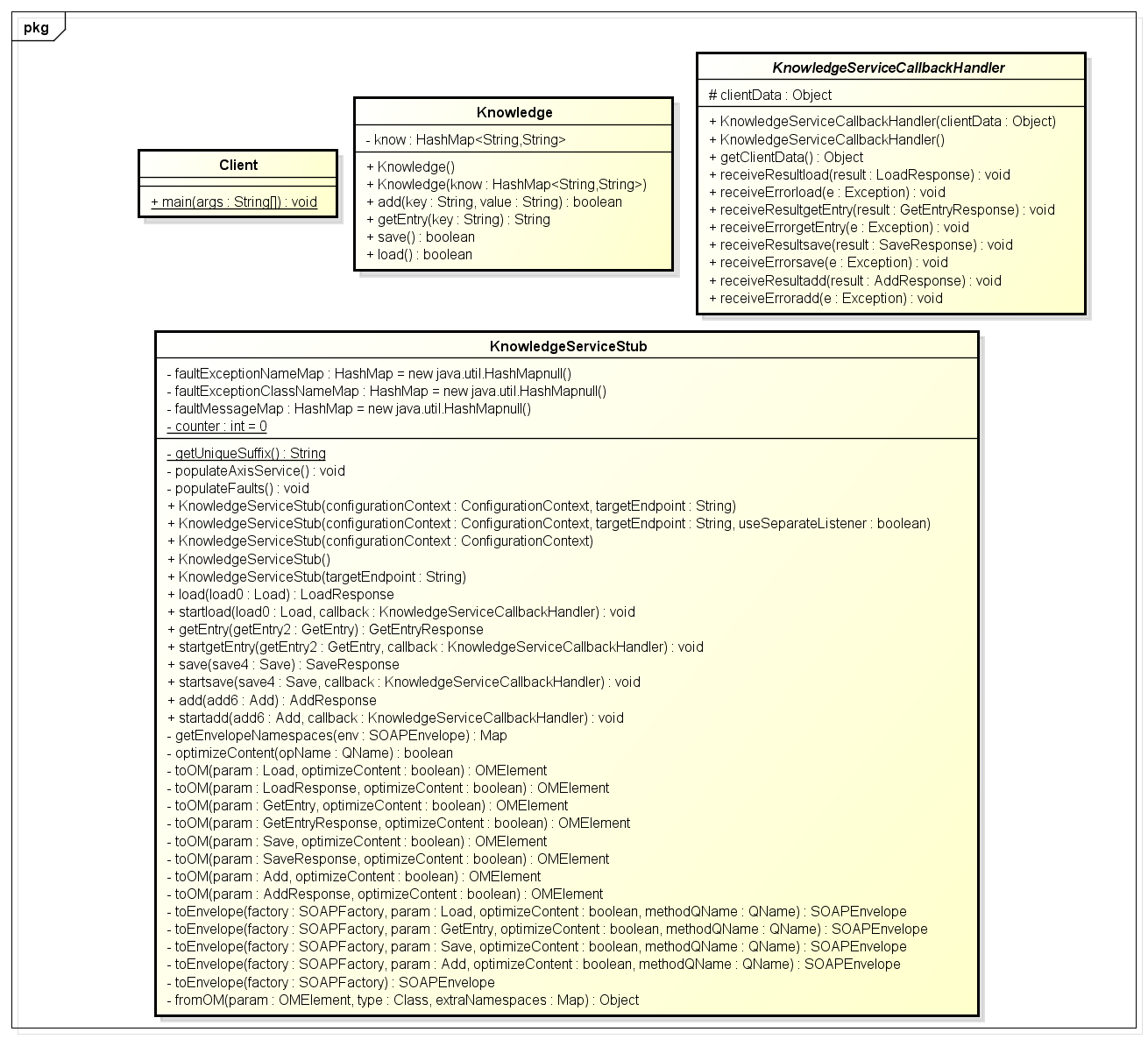
## Geschätzte Zeit

|  |  |  |
| --- | --- | --- |
| Tätigkeit | Klune | Schuschnig |
| Einlesen | 2 h | 2 h |
| Erstellen der Server Klasse |  | 1 h |
| Erstellen der Client Klasse |  | 30 min |
| GUI |  | 30 min |
| Erstellen von Web Service | 1 h |  |
| Erstellen von Web Service Client | 1 h |  |
| Protokoll | 45 min | 45 min |
| Summe | 4 h 45 min | 4 h 45 min |
| Gesamt | 9 h 30 min | |

## Tatsächliche Zeit

|  |  |  |
| --- | --- | --- |
| Tätigkeit | Klune | Schuschnig |
| Einlesen | 2 h | 2 h |
| Erstellen der Server Klasse |  | 1 h |
| Erstellen der Client Klasse |  | 30 min |
| GUI |  | 30 min |
| Erstellen von Web Service | 1 h |  |
| Erstellen von Web Service Client | 1 h |  |
| Protokoll | 45 min | 45 min |
| Summe | 4 h 45 min | 4 h 45 min |

# Designüberlegung



# Datenmodell

Das Datenmodell wurde realisiert über eine Hashmap in der das Topic als Key und der Content als Value gespeichert wurde. Dadurch gestaltet sich die Suche äußerst einfach durch eine Methode die von der Klasse Hashmap zur Verfügung gestellt wird. Auch das speichern gestaltet sich einfach durch von Java zur Verfügung gestellte Klassen.

## Suche

Die Methode zur Suche heißt: map.getValue(Key)

Zurückgegeben wird der Value also das Topic der Wissensdatenbank.

Die gesamte Such-Methode:

**public** String getEntry(String key) {

**return**(know.get(key));

}

Dabei muss beachtet werden, dass die Suche Casesensitive ist. (Also das genaue Topic)

## Speichern

Das Speichern einer Hashmap kann durch die von Java zur Verfügung gestellte Klasse Properties gelöst werden. Diese ist für das persistente speichern von Hashmaps ausgelegt.

Schritte:

1. Erstellen einer Properties Variable:

Properties properties = **new** Properties();

2. Setzen der einzelnen Properties aus der Hashmap:

properties.setProperty((String)pairs.getKey(), (String)pairs.getValue());

3. Mit einem Outputstream speichern hier wird ein FileWriter verwendet die Methode zum speichern von Properties ist store:

FileWriter writer;

writer = **new** FileWriter("conf.properties");

properties.store(writer, **null**);

## Laden

Das Laden kann ebenfalls mit einer Methode der Klasse Proberties gelöst werden diese ist load

Erster Schritt gleich wie bei speichern erstellen einer Proberties Variable.

2. Zweiter Schritt mittels eines Inputstreams lesen:

FileReader reader;

reader = **new** FileReader("conf.properties");

properties.load(reader);

3. Anschließend muss das Ergebnis noch in eine Hashmap gespeichert werden:

**for** (**final** String name: properties.stringPropertyNames())

know.put(name, properties.getProperty(name));

## Hinzufügen

Das Hinzufügen eines Eintrags geht ganz einfach mit der Methode put() der Klasse Hashmap wenn der Eintrag schon vorhanden ist wird der Content nur hinzugefügt das funktioniert folgendermaßen:

**if**(know.containsKey(key) == **false**) {

know.put(key,value)

}

**else** {

String hilf = know.get(key) + "\n" + value;

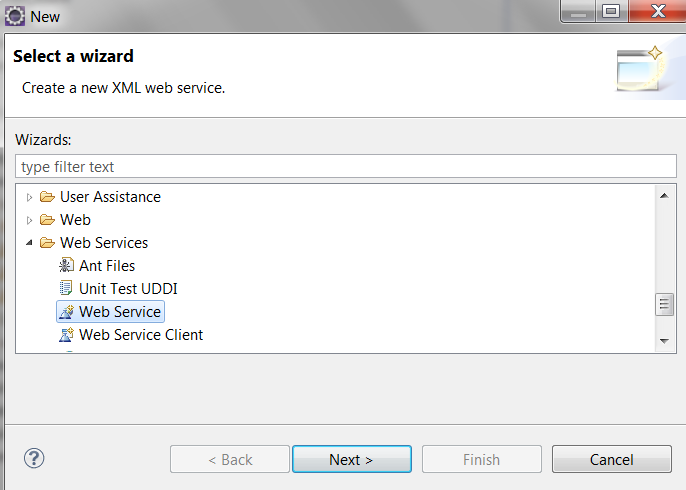
know.remove("key");

know.put(key, hilf);

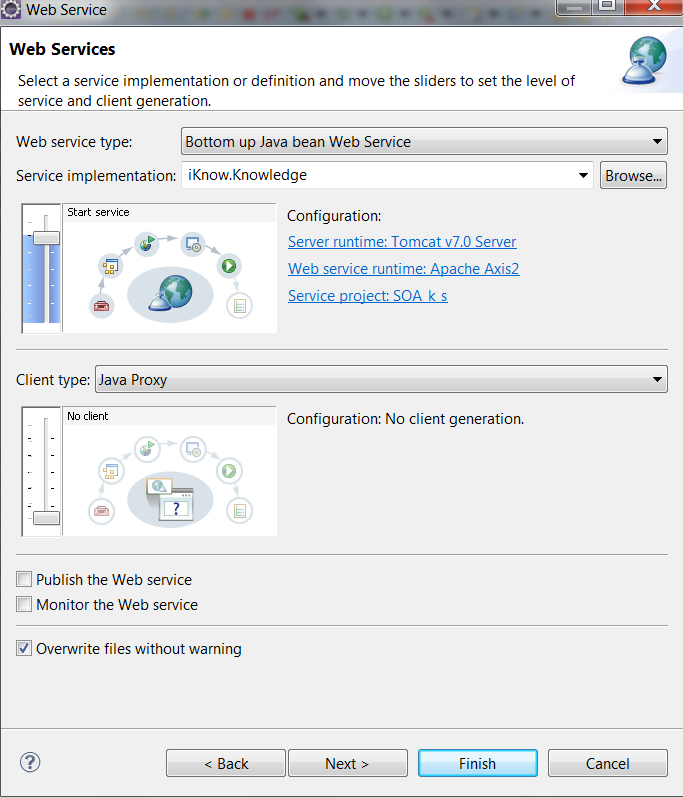
# Erstellen eines Web Services

Um einen Webservice anbieten zu können, braucht man ein .wsdl File. Dieses File wird generiert aus einer Java Klasse in der die einzelnen Services als Methoden Definiert werden.

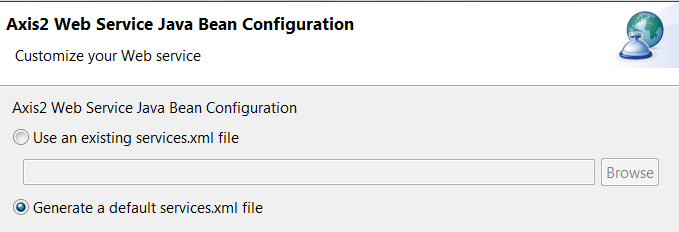
Rechtsklick auf das Projekt und dann auf „new“ -> other.

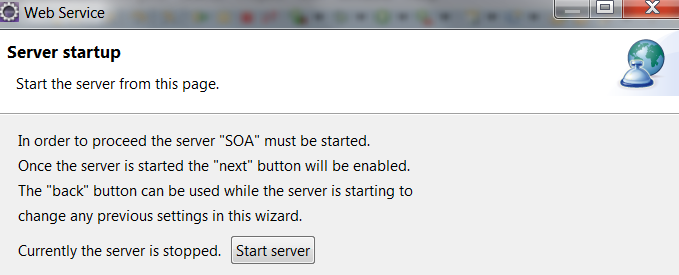


In diesem Auswahlfenster muss „Web Service“ Ausgewählt werden.



Im nächsten Schritt muss die Service Impl. Ausgewählt werden, sowie der gewünschte Server, Web Service runtime (Axis2(mehr dazu in einem anderen Kapitel)) und das Service Projekt.





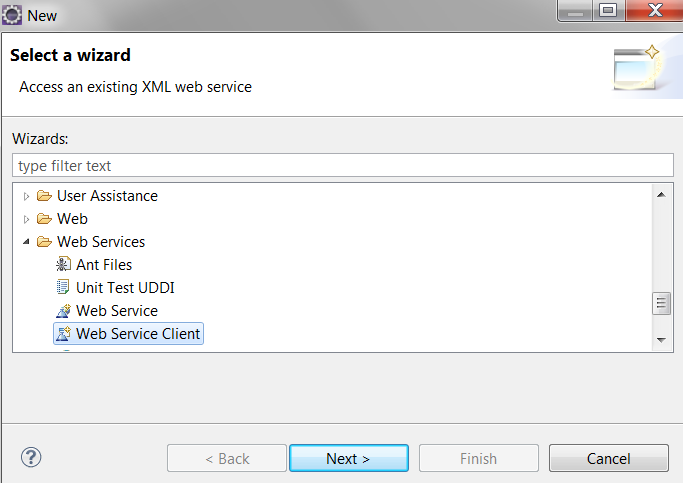
Danach muss noch der Server gestartet werden. Finish

Nach diesen Schritten wird ein .wsdl File aus der Service Implementierung erzeugt. Dieses File findet man unter WebContent/wsdl/*ServiceImpl*.wsdl

# Erstellen eines Web Service Clients

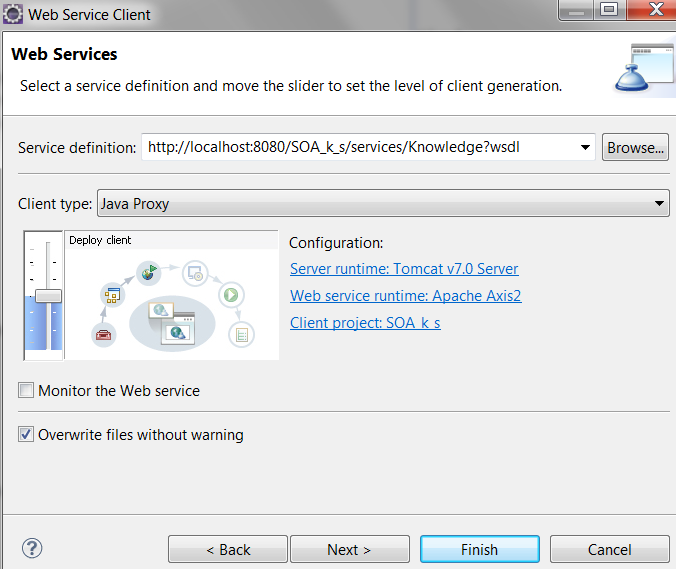
Um auch die Services die ein Server zu Verfügung steht wird ein Client benötigt. Dieser kann aber nur dann gerneriert werden, wenn bereits ein Service auf einem Server angeboten wird. Der Server muss laufen. Dieser Client muss über einen Stub verfügen, der die Kommunikation zwischen Server und Client herstellt.

Rechtsklick auf das Projekt und dann auf „new“ -> other.

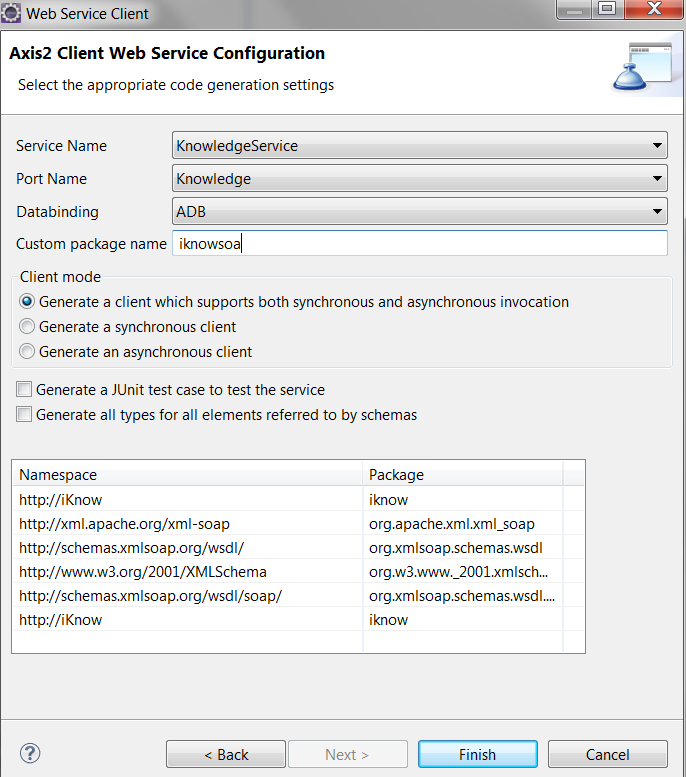


In „Service definition“ muss das .wsdl File welches auf dem Server liegt eingebunden werden. <http://localhost:8080/SOA_k_s/services/Knowledge?wsdl>

In Configuration muss wieder der Server und Axis2 ausgewählt werden.



Hier muss lediglich der „Custom Package Name“ geändert werden, da per Default das Package eingetragen ist in dem die Service Implementierung liegt.



Danach noch auf Finish und fertig!

# Axis2

Publiziert die Web Services als .wsdl

Dieses Programm ist Standardmäßig nicht in der Eclipse DIE vorhanden(nur Axis). Um dieses Programm nutzen zu können muss es eingebunden werden.

1. Herunterladen von der offiziellen Axis2 Homepage als binaries und ablegen in einem Folder der nicht bewegt wird.
2. Window -> Preferences
3. Web Services -> Axis2 Preferences
4. In Axis2 Runtime unter Axis2 runtime location muss der Pfad zu den Axis2 binaries eingegeben werden.
5. Danach müssen für das Projekt noch Einstellungen getroffen werden. Rechtsklick auf das Projekt -> Properties -> Project Facets
6. Dort muss die Version von Dynamic Web Module auf 2.5 gesetzt werden
7. Danach kann Axis2 Web Services aktiviert werden.

# Resultate / Niederlagen

Wir hatten bei der Service Impl keine großen Schwierigkeiten. Lediglich OpenESB stellte eine Hürde dar, weil wir für diese Übung Apache Tomcat Server und Eclipse verwendet haben, statt Glassfish und NetBeans. Dadurch mussten wir unsere DIE und Webserver umstellen.

Die realisierung von BPEL und ESB war nicht möglich.

# GIT log

# Quellen