PROJECT REPORT

Programming Web-Services :: ID-2208

Members: Mohammad Omar Nasir
Xin Zhao

06-March-2017

1. PROJECT IMPLEMENTATION DIAGRAM

The whole procedure of this project is shown in Figure 1.

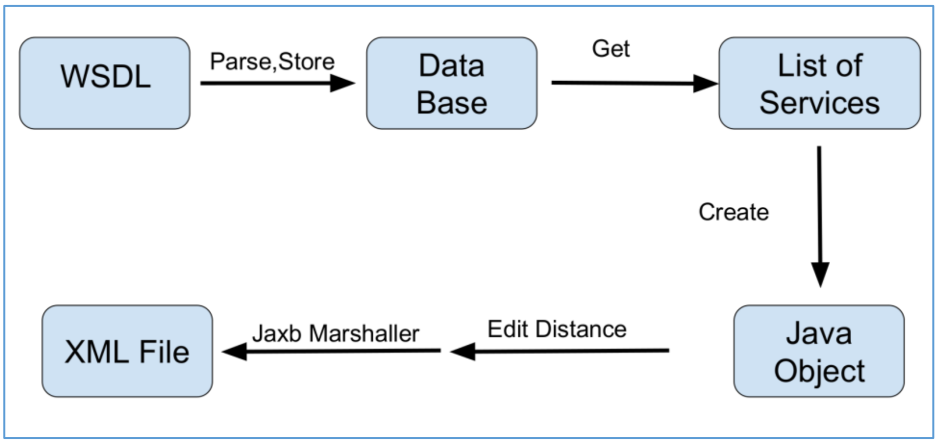


Figure 1. Flow chart of the whole procedure

2. WSDL PARSING

We analyzed the structure of all the wsdl files, and summarized the general hierarchy as shown in Figure 2.

2.2 DOM Parsing

The procedures are listed as follows:

1. Get list of all wsdl files in a folder and then parse them using DOM parser.
2. After parsing, we get the **<Root>** element, **<Service>,<PortType>,<Binding>,<Types>,<Message>** are all children of **<Root>** , as shown in Figure 2
3. We save all elements inside objects.

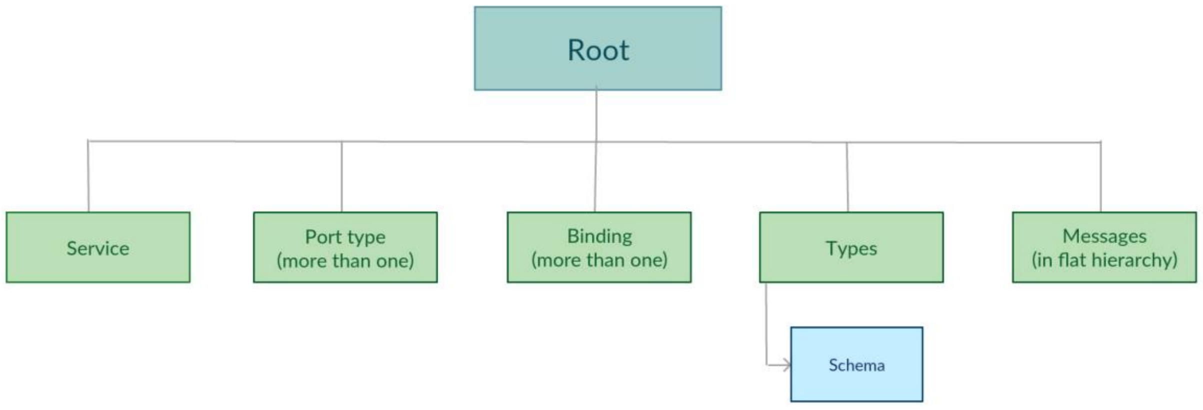


Figure 2. Common structure hierarchy of our wsdl fiels

2.2 Get the "Message" Object.

In order to get messages which are used for semantic or syntactic matching according to the requirements, we need following steps:

1. Get the "Service" Object. We extract both WSDL name and service name to avoid the case that two services have the same name.
2. Get the "PortType" Object. We iterate through "PortType" nodes list, and get all the nodes corresponding to "PortType" tag. Later, we go through all the children of "PortType" nodes, to search for all the "Operation" tags under <PortType>.
3. Get the "Binding" Object. Since headers of soap messages are only found inside operations as nodes, and these operations are only described under <Binding> tags, we need to go through all the "Binding" nodes to find all the operations there and confirm if header exists or not.
4. At this point, we handle extraction of element part differently, based on semantic or syntactic choice.
    a. **For syntactic matching**:

    Get the "Message" Object. After finding "message" tag, we iterate through its childNodes, get "Element" or "name" attribute in <part> tag. If it has "name" tag, it means there is no <element> associated with this <message>, we have to iterate through <element> inside Schema. If it has <element>, we need to pay attention to <complextype>, since there is no need to go through it.

    b. For Semantic Matching:

    Get the "Message" Object. After finding "message" tag, we iterate through its childNodes, get "type" or "element" attribute in <part> tag. For both values, we find the corresponding <node> in <schema>. This is recursively iterated until we find the lowest possible element containing the annotation attribute. If we reach the end, no child nodes exist and no annotation is found, another recursive function is called from inside the first function which iterates through the whole schema again to find corresponding types for last level elements. If found, this new function calls the first function again and sends the object to it. Now the first function will once again attempt to find annotation. If successful, all functions are exited successfully.
5. Store the useful information in a list.

    We mark input operation as 0 and output operation as 1, and store it with corresponding message name in a list followed the form <messagename_string,mark(0 or 1)>.

2.3. Data for matching

We can compare the items from output operations of one wsdl with items which belong to input operations of all the other wsdl files.

3. STORE IN DATABASE

We store the useful information in the database as shown in Figure 3. According to the explanation in Section 2, we first store list of Service Names in “Service” table. Then once parsing of <portTypes> and <bindings> is finished, we store all operations inside “Operation” table. Then once we successfully parse <message> nodes based on their relationship with <operation> nodes, we fetch corresponding <element> and/or “annotation” attribute and store it finally in “Element” table.

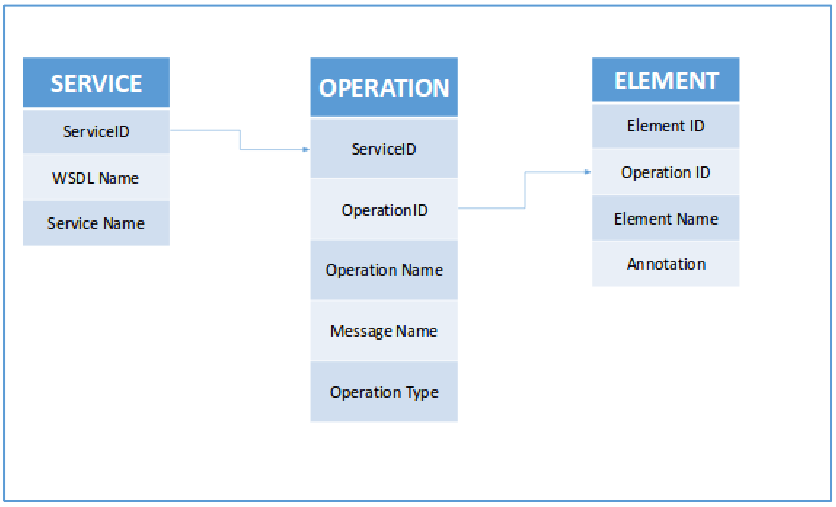
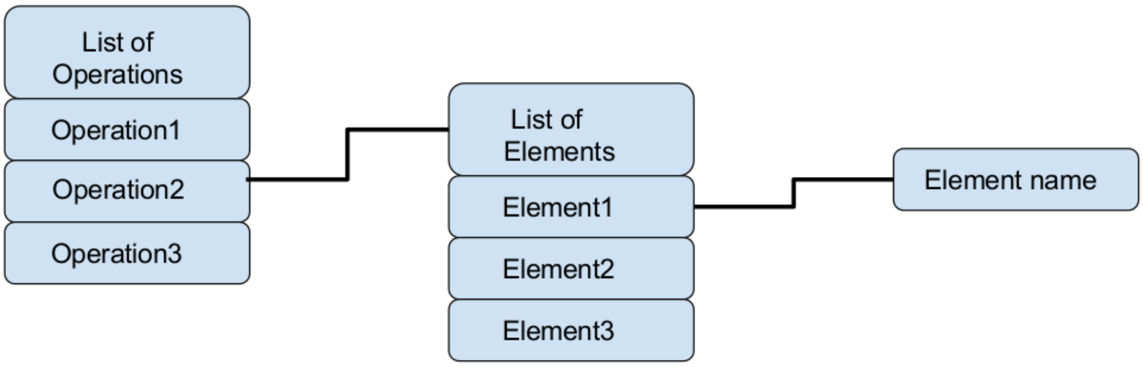


Figure 3. Structure of Database

4. CREATE JAVA OBJ

We get the list of services from database. Then we create Java object of services class which fit the structure of output xml example.



5. EDIT DISTANCE AND CALCULATE SCORES AND MATCH

For syntactic matching and scoring we do as follows: operations are marked by the average of the element scores. And service score should marked according to the average of the operation scores.

For semantic matching: we use proper OWLClasses from the Ontolog attributes which are offered in the complementary files.

6. MARSHALLER:

As mentioned in the 4th part, we store the data in the same structure and hierarchy as example output.xml. It makes the procedure here more convenient. We use Jaxb Marshaller to directly transform the output data in the xml files under the path: src/main/output/Output.xml

7. RUNNING INSTRUCTIONS

To setup DB: Install PostgreSQL. Inside the SQLScripts folder, the complete database schema is written in sql scripts. Simply run them after creating a database named “project” from PGAdmin4 terminal.

If you use intellij in linux, you need to open the programing in bin folder,
then run the program by using the command: ./idea.sh

and after that run database by command: service postgresql start