Sample on how to create a web service consumer in Domino Designer 8.5.2 with out-of-the-box code element

Abstract

This sample shows how I used the web service consumer out-of-the-box code element in Domino Designer 8.5.2 to consume a SAP web service. I couldn't find a sample which was using the Lotus Notes "on board" elements, thats why I figured it out. The sample reads a SAP web service, which was generated on top of a custom RFC module.

Table of Contents

This sample shows how I used the web service consumer out-of-the-box code element in Domino Designer 8.5.2 to consume a SAP web service. I couldn't find a sample which was using the Lotus Notes "on board" elements, thats why I figured it out. The sample reads a SAP web service, which was generated on top of a custom RFC module. Because I heard sth like "it doesn't work, using the "onboard" elements", I had to try it of my own. Of course, there are still the possibility to write your own java soap client classes.

Create the SAP WebService

We were using an own SAP module called „ZFindeLieferant“. The module returns the supplierNo (Lieferantennummer) as well as the supplieraddress (Lieferantenadresse). Input is a String, searching for variants.

Important was, when creating the WebService in the SAP Developer environment, to choose the policy -option “standard” instead of the option “ws-policy”.

The WebService needs authentication, so credentials will be necessary.

The outcoming WSDL looked like this:

```
<?xml version="1.0" encoding="utf-8"?>
<wsdl:definitions targetNamespace="urn:sap-com:document:sap:soap:functions:mc-style"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
xmlns:tns="urn:sap-com:document:sap:soap:functions:mc-style"
xmlns:n1="urn:sap-com:document:sap:rfc:functions">
<wsdl:documentation>
<sidl:sidl
xmlns:sidl="http://www.sap.com/2007/03/sidl"/>
</wsdl:documentation>
<wsdl:types>
<xsd:schema attributeFormDefault="qualified" targetNamespace="urn:sap-com:document:sap:rfc:functions">
<xsd:simpleType name="char10">
<xsd:restriction base="xsd:string">
<xsd:maxLength value="10"/>
</xsd:restriction>
</xsd:simpleType>
<xsd:simpleType name="char80">
<xsd:restriction base="xsd:string">
<xsd:maxLength value="80"/>
</xsd:restriction>
</xsd:simpleType>
</xsd:schema>
<xsd:schema attributeFormDefault="qualified" targetNamespace="urn:sap-com:document:sap:soap:functions:mc-style"
xmlns:n0="urn:sap-com:document:sap:rfc:functions">
<xsd:import namespace="urn:sap-com:document:sap:rfc:functions"/>
<xsd:complexType name="Zlifareturn">
<xsd:sequence>
<xsd:element name="Lifnr" type="n0:char10"/>
<xsd:element name="Linek" type="n0:char80"/>
```

```
</xsd:sequence>
</xsd:complexType>
<xsd:complexType name="TableOfZlifareturn">
<xsd:sequence>
<xsd:element name="item" type="tns:Zlifareturn" minOccurs="0" maxOccurs="unbounded"/>
</xsd:sequence>
</xsd:complexType>
<xsd:element name="ZFindeLieferant">
<xsd:complexType>
<xsd:sequence>
<xsd:element name="Returnvalue" type="tns:TableOfZlifareturn"/>
<xsd:element name="Sortl" type="n0:char10"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
<xsd:element name="ZFindeLieferantResponse">
<xsd:complexType>
<xsd:sequence>
<xsd:element name="Returnvalue" type="tns:TableOfZlifareturn"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:schema>
</wsdl:types>
<wsdl:message name="ZFindeLieferant">
<wsdl:part name="parameters" element="tns:ZFindeLieferant"/>
</wsdl:message>
<wsdl:message name="ZFindeLieferantResponse">
<wsdl:part name="parameters" element="tns:ZFindeLieferantResponse"/>
</wsdl:message>
<wsdl:portType name="ZMM_PUR_FINDE_LIEFERANT">
<wsdl:operation name="ZFindeLieferant"><wsdl:input message="tns:ZFindeLieferant"/>
<wsdl:output message="tns:ZFindeLieferantResponse"/>
</wsdl:operation></wsdl:portType>
<wsdl:binding name="ZMM_PUR_FINDE_LIEFERANT" type="tns:ZMM_PUR_FINDE_LIEFERANT">
<soap:binding transport="http://schemas.xmlsoap.org/soap/http" style="document"/>
<wsdl:operation name="ZFindeLieferant">
<soap:operation soapAction="" style="document"/>
<wsdl:input>
<soap:body use="literal"/>
</wsdl:input><wsdl:output>
<soap:body use="literal"/>
</wsdl:output>
</wsdl:operation>
</wsdl:binding>
<wsdl:service name="ZMM_PUR_FINDE_LIEFERANTService">
<wsdl:port name="ZMM_PUR_FINDE_LIEFERANT" binding="tns:ZMM_PUR_FINDE_LIEFERANT">
<soap:address
location="http://sap.acme.com:8080/sap/bc/srt/rfc/sap/zmm_pur_finde_lieferant/200/zmm_pur_finde_lieferant/zmm_pur_finde_lieferant"/>
</wsdl:port>
</wsdl:service>
</wsdl:definitions>
```

CREATE THE WEB SERVICE CONSUMER IN LOTUS NOTES DESIGNER

- Web Service Consumer is an „Out-of-the-box“-component and will be found in the „code“-section.
- First we saved the SAP WSDl file locally.
- We were taking the action “New Web Service Consumer” from type “java” and give the local WSDL-file as input.
- The java SOAP client stub were created automatically.
- Now the question, how to use that.

CREATE A JAVA AGENT

Create a new agent of type java.

- Import the previously created web service consumer. Therefore is a standard selection available.
- First init the stub object like this:

```
ZMM_PUR_FINDE_LIEFERANT stub = new ZMM_PUR_FINDE_LIEFERANTServiceLocator()
                .getZMM_PUR_FINDE_LIEFERANT();
```

- This is what you actually find in the comment oft he generated classes
- After that, set the credentials, set the input object call the web service method and extract the returnTable.
- Hope this code give a more precise example.

Agent Source:

```
public class JavaAgent extends AgentBase {
    public void NotesMain() {
        try {
            System.out.println("START");
            Session session = getSession();
            //AgentContext agentContext = session.getAgentContext();
            // (Your code goes here)
            ZMM_PUR_FINDE_LIEFERANT stub = new ZMM_PUR_FINDE_LIEFERANTServiceLocator()
                    .getZMM_PUR_FINDE_LIEFERANT();
            stub.setCredentials("user", "password");
            TableOfZlifareturnHolder returnTable = new TableOfZlifareturnHolder();
            functions.rfc.sap.document.Char10 valueInput = new Char10("Companyname");

            //System.out.println(stub.getEndpoint());
            stub.ZFindeLieferant(returnTable, valueInput);
            //System.out.println("returnTable.getLength: " + returnTable.value.getItem().length);

            int pos = 0;
            while ((returnTable.value.getItem().length > pos) && (returnTable.value.getItem(pos) != null)) {
                // put your code here to work with the data returned from SAP
                System.out.println("returnTable.value: " + returnTable.value.getItem(pos).getLifnr());
                System.out.println("returnTable.value: " + returnTable.value.getItem(pos).getLinek());
                pos++;
            }


        } catch (Exception e) {
            System.out.println(e.getMessage());
            e.printStackTrace();
        }
    }
}
```

Best regards,
Peter Arndt, SIT GmbH