Java Access to Oracle CRM On Demand

By: Joerg Wallmueller – Melbourne, Australia

Introduction

The CRM On Demand Web Services interfaces are a very important piece of our integration story, and utilized by a lot of customers. That integration usually takes place by using either EAI middleware, or using Java code.

This document guides you in setting up a little Java application, which accesses your CRM On Demand account using Web Service interface.

Java skills are not required to follow the tutorial, just solid know-how in copy-and-paste!

Requirements

You will need the following two components:

a) Java Development Environment

For this tutorial we are going to use Oracle JDeveloper. JDeveloper can be downloaded here: http://www.oracle.com/technology/software/products/jdev/index.html. Of course you can use any other popular Java Development Environment as Eclipse or NetBeans.

b) Apache Axis

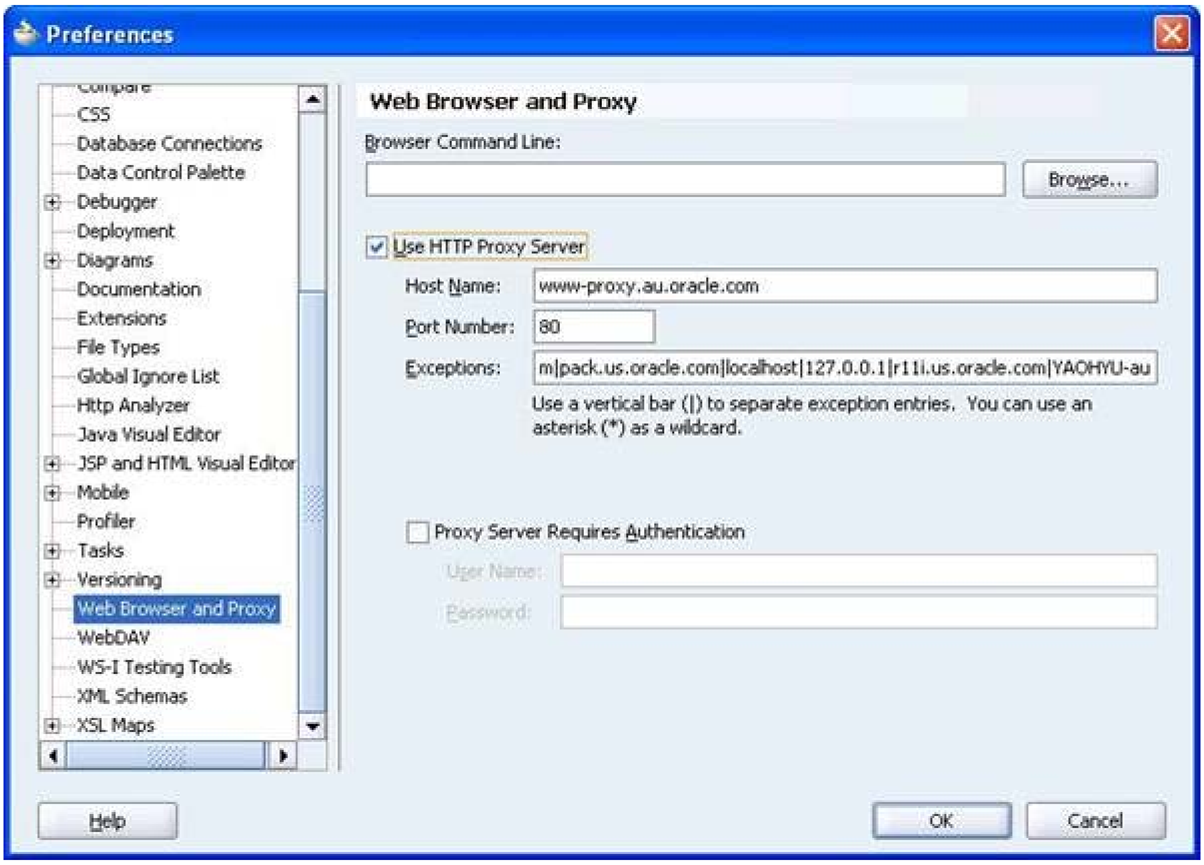
We will use Apache Axis to automatically generate “Java Proxy classes” based on our CRM On Demand Web Services. You need to download the Apache Axis 1.4 libraries (axis-bin-1_4.zip) from http://ws.apache.org/axis/ to a local folder on your disk. For this tutorial, extract the zip file to the root of your D\: drive, and rename it for simplicity from “D:\axis-1_4” to “D:\Axis” afterwards.

Step 1: Generate the skeleton Java program

First we are going to create a basic Java program which logs into CRM On Demand, prints out the session id and logs off again. As we don’t invoke a Web Service yet, we don’t need to download any WSDL file.

1) Set JDeveloper Proxy settings

To ensure connectivity to the CRM OD server, add the Oracle proxy server in the screen “Tools→Preferences → Web Browser and Proxy” as shown below.



Web Browser and Proxy

Browser Command Line:

Browse...

[✓] Use HTTP Proxy Server

Host Name: www-proxy.au.oracle.com

Port Number: 80

Exceptions: m|pack.us.oracle.com|localhost|127.0.0.1|r11i.us.oracle.com|YAOHYU-au

Use a vertical bar (|) to separate exception entries. You can use an asterisk (*) as a wildcard.

[ ] Proxy Server Requires Authentication

User Name:

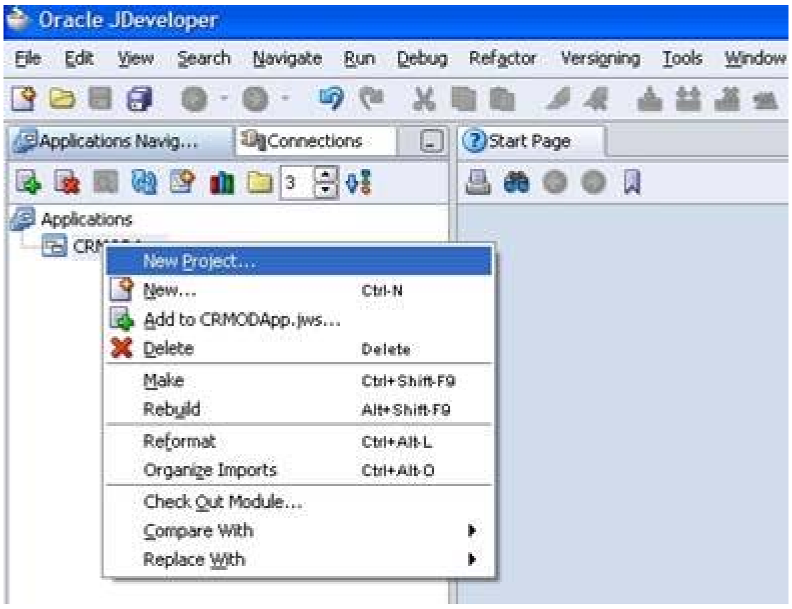
Password:

2) Create a new Java application

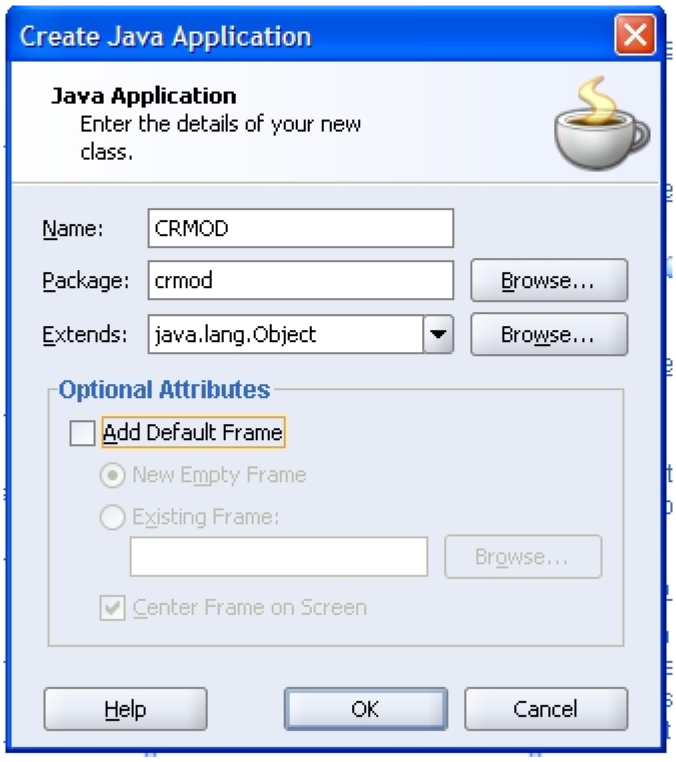
Start JDeveloper, and create a new Application (File/New/General →Application). Give it the name "CRMODApp" and stay with the other defaults. Cancel the creation of the initial project as part of the wizard.

3) Create a new Java project

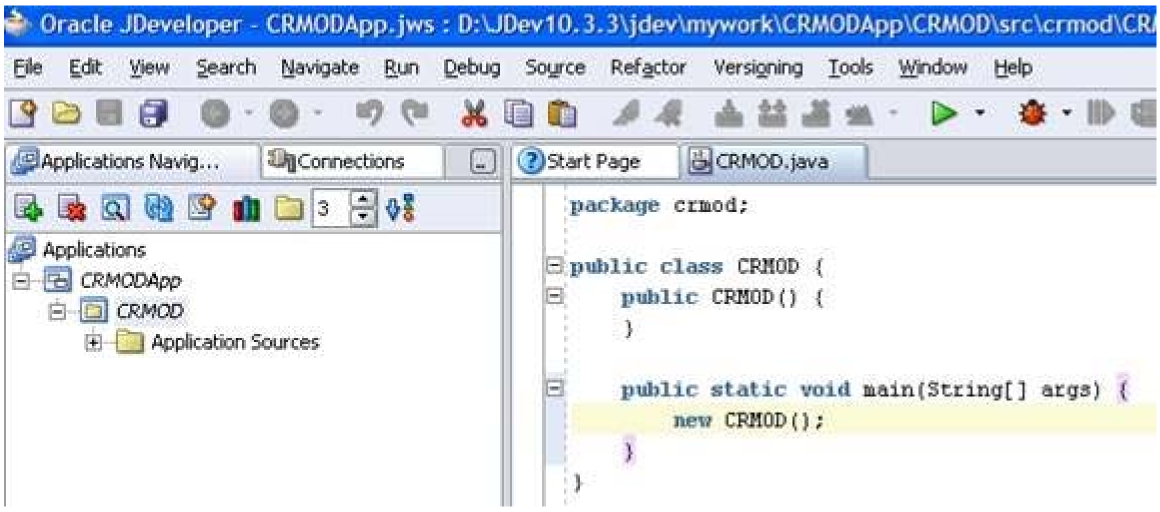
Select the newly created CRMODApp in the Applications navigator, and select from the context menu "New Project".



Choose "General/Projects → Java Application Project". In the wizard, assign the name "CRMOD" and stay with all the defaults. After creating the new project, the wizard will ask you about initial information about the Java application. Name you new application "CRMOD" and de-select "Add Default Frame".



You should result in the following source code being generated.


Oracle JDeveloper - CRMODApp.jws : D:\JDev10.3.3\jdev\mywork\CRMODApp\CRMOD\src\crmod\CR
File Edit View Search Navigate Run Debug Source Refactor Versioning Tools Window Help
Applications Navig...
Connections
Start Page
CRMOD.java
Applications
CRMODApp
CRMOD
Application Sources

```
package crmod;

public class CRMOD {
    public CRMOD() {
    }

    public static void main(String[] args) {
        new CRMOD();
    }
}
```

4) Import required packages

We need to add a few Java libraries. Paste the following three lines of code at the second line of the source code (below the “Package crmod;” line):

```
import java.net.HttpURLConnection;
import java.net.URL;
import java.util.StringTokenizer;
```

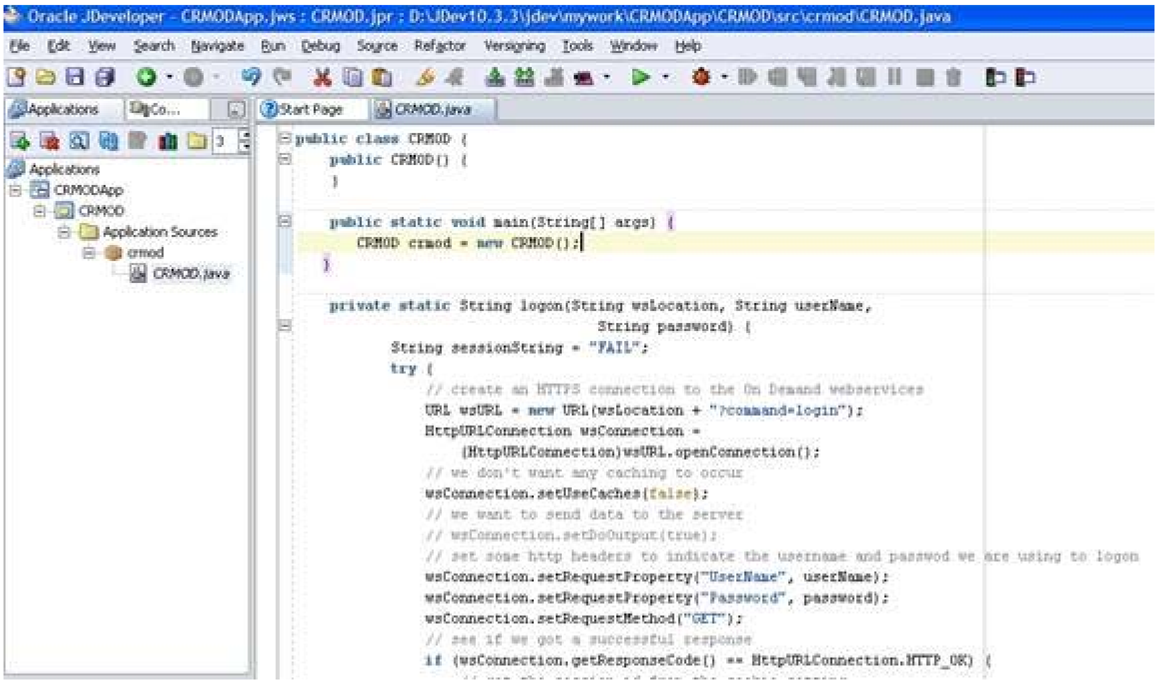
5) Import code to logon and logoff

Paste the following code at the bottom of your code, before the closing “}”. Thanks to Charles McGuinness who provided this revised version of our sample code. *If you use the standard sample code provided via CRM On demand Helpdesk, the logoff command won’t work and your sessions stay open for 10 minutes. Not a big deal for your demo, but raised a few issues with production customers.*

```
private static String logon(String wsLocation, String userName,
                              String password) {
    String sessionString = "FAIL";
    try {
        // create an HTTPS connection to the On Demand webservices
        URL wsURL = new URL(wsLocation + "?command=login");
        HttpURLConnection wsConnection =
            (HttpURLConnection)wsURL.openConnection();
        // we don't want any caching to occur
        wsConnection.setUseCaches(false);
        // we want to send data to the server
        // wsConnection.setDoOutput(true);
        // set some http headers to indicate the username and passwod we are using to logon
        wsConnection.setRequestProperty("UserName", userName);
        wsConnection.setRequestProperty("Password", password);
        wsConnection.setRequestMethod("GET");
        // see if we got a successful response
        if (wsConnection.getResponseCode() == HttpURLConnection.HTTP_OK) {
            // get the session id from the cookie setting
            sessionString = getCookieFromHeaders(wsConnection);
        }
    } catch (Exception e) {
        System.out.println("Logon Exception generated :: " + e);
    }
    return sessionString;
}
/*
* log off an existing web services session, using the sessionCookie information
* to indicate to the server which session we are logging off of
* @param wsLocation - location of web services provider
* @param sessCookie - cookie string that indicates our sessionId with the WS provider
*
*/

private static void logoff(String wsLocation, String sessionCookie) {
    try {
        // create an HTTPS connection to the On Demand webservices
        URL wsURL = new URL(wsLocation + "?command=logoff");
        HttpURLConnection wsConnection =
            (HttpURLConnection)wsURL.openConnection();
        // we don't want any caching to occur
        wsConnection.setUseCaches(false);
        // let it know which session we're logging off of
        wsConnection.setRequestProperty("Cookie", sessionCookie);
        wsConnection.setRequestMethod("GET");
        // see if we got a successful response
        if (wsConnection.getResponseCode() == HttpURLConnection.HTTP_OK) {
            // if you care that a logoff was successful, do that code here
            // showResponseHttpHeaders(wsConnection);
        }
    } catch (Exception e) {
        System.out.println("Logoff Exception generated :: " + e);
    }
}
```

```
/*
* given a successful logon response, extract the session cookie information
* from the response HTTP headers
*
* @param wsConnection successfully connected connection to On Demand web services
* @return the session cookie string from the On Demand WS session or FAIL if not
found*
*/

private static String getCookieFromHeaders(HttpURLConnection wsConnection) {
  // debug code - display all the returned headers
  String headerName;
  String headerValue = "FAIL";
  for (int i = 0; ; i++) {
    headerName = wsConnection.getHeaderFieldKey(i);
    if (headerName != null && headerName.equals("Set-Cookie")) {
      // found the Set-Cookie header (code assumes only one cookie is being set)
      headerValue = wsConnection.getHeaderField(i);
      break;
    }
  }
  // return the header value (FAIL string for not found)
  return headerValue;
}
private static String getSessionId(String cookie) {
    StringTokenizer st = new StringTokenizer(cookie, ";");
    String jsessionid = st.nextToken();
    st = new StringTokenizer(jsessionid, "=");
    st.nextToken();

    return st.nextToken();
  }
```

Your code now should look as follows:


Oracle JDeveloper - CRMODApp.jws : CRMOD.jpr : D:\JDev10.3.3\jdev\mywork\CRMODApp\CRMOD\src\crmod\CRMOD.java
File Edit View Search Navigate Run Debug Source Refactor Versioning Tools Window Help
Applications
Start Page
CRMOD.java
Applications
CRMODApp
CRMOD
Application Sources
crmod
CRMOD.java

```
public class CRMOD {
    public CRMOD() {
    }

    public static void main(String[] args) {
        CRMOD crmod = new CRMOD();
    }

    private static String logon(String wsLocation, String userName,
                                String password) {
            String sessionString = "FAIL";
            try {
                // create an HTTPS connection to the On Demand webservices
                URL wsURL = new URL(wsLocation + "?command=login");
                HttpURLConnection wsConnection =
                    (HttpURLConnection)wsURL.openConnection();
                // we don't want any caching to occur
                wsConnection.setUseCaches(false);
                // we want to send data to the server
                // wsConnection.setDoOutput(true);
                // set some http headers to indicate the username and passwod we are using to logon
                wsConnection.setRequestProperty("UserName", userName);
                wsConnection.setRequestProperty("Password", password);
                wsConnection.setRequestMethod("GET");
                // see if we got a successful response
                if (wsConnection.getResponseCode() == HttpURLConnection.HTTP_OK) {
```

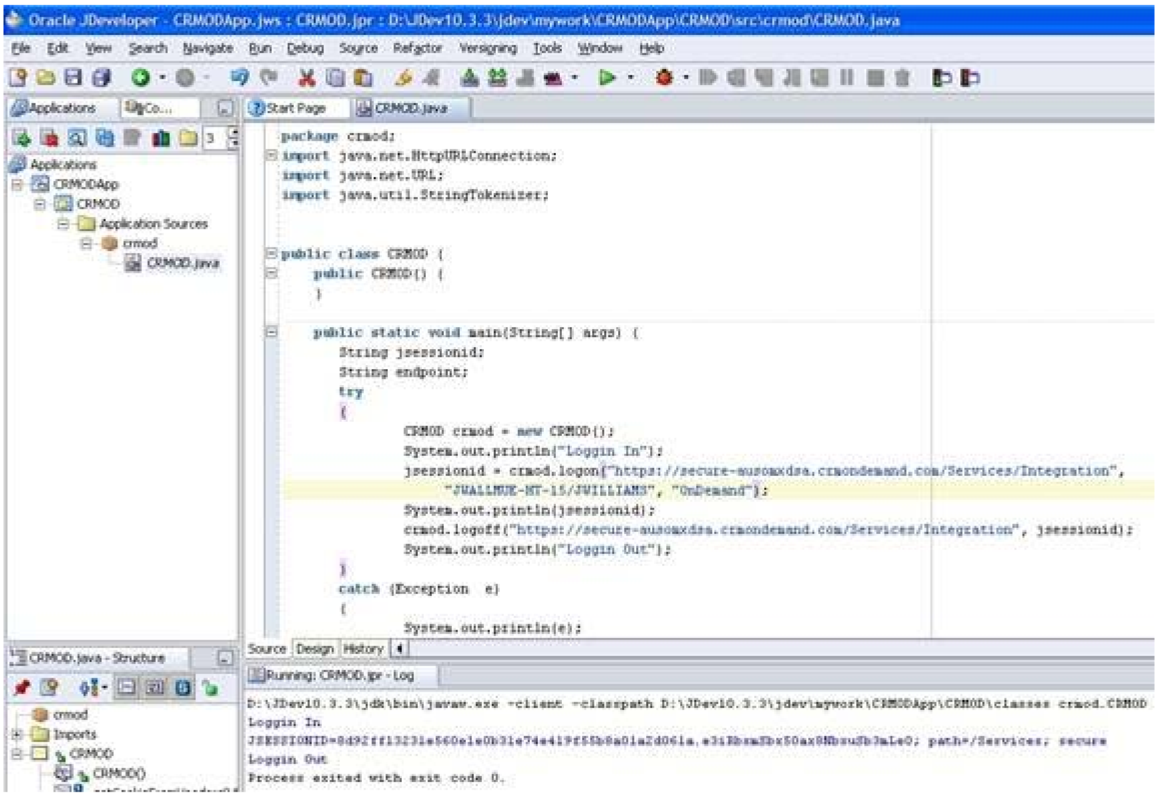
6) Invoke the login and logoff code

Time to do use our imported code! Go to the “public static void main” method, and **replace** the existing line “new CRMOD();” with the following code (but please change the username to one of your demo instances):

```
String jsessionid, jsessionid_full;
    String endpoint;
    try
    {
                CRMOD crmod = new CRMOD();
                System.out.println("Loggin In");
                jsessionid_full = crmod.logon("https://secure-ausomxdsa.crmondemand.com/Services/Integration", "JWALLMUE-HT-15/JWILLIAMS", "OnDemand");
                jsessionid = getSessionId(jsessionid_full);
                System.out.println(jsessionid);
                crmod.logoff("https://secure-ausomxdsa.crmondemand.com/Services/Integration", jsessionid_full);
                System.out.println("Loggin Out");
    }
    catch (Exception  e)
    {
        System.out.println(e);
    }
```

7) Test our code

New let’s test our code: In the application navigator select the “CRMOD.java” class and select from the context menu “Run”. You will see in the console a message which includes the CRM On Demand session id returned.



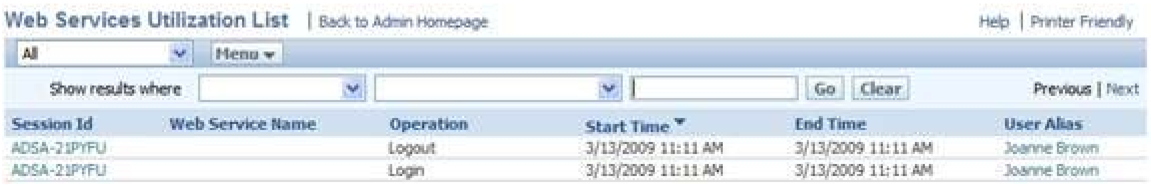
```
package crmod;
import java.net.HttpURLConnection;
import java.net.URL;
import java.util.StringTokenizer;


public class CRMOD {
    public CRMOD() {
    }

    public static void main(String[] args) {
        String jsessionid;
        String endpoint;
        try
        {
                CRMOD crmod = new CRMOD();
                System.out.println("Loggin In");
                jsessionid = crmod.logon("https://secure-ausomxdsa.crmondemand.com/Services/Integration",
                    "JWALLMUE-HT-15/JWILLIAMS", "OnDemand");
                System.out.println(jsessionid);
                crmod.logoff("https://secure-ausomxdsa.crmondemand.com/Services/Integration", jsessionid);
                System.out.println("Loggin Out");
        }
        catch (Exception  e)
        {
                System.out.println(e);
```

```
D:\JDev10.3.3\jdk\bin\javaw.exe -client -classpath D:\JDev10.3.3\jdev\mywork\CRMODApp\CRMOD\classes crmod.CRMOD
Loggin In
JSESSIONID=8d92ff13231e560e1e0b31e74e419f55b8a01a2d061a.e3iRbxmSbx50ax8NbxuSb3mLe0; path=/Services; secure
Loggin Out
Process exited with exit code 0.
```

8) Validate success of login/logout in CRM On Demand

As our Java program already performed a login, we will be able to validate that in CRM On Demand. Open the instance against which you tested, and navigate to “Admin → Web Services Utilization”.

You will see at the top of the list the session details generated from our Java program:


Web Services Utilization List | Back to Admin Homepage

Session Id	Web Service Name	Operation	Start Time	End Time	User Alias
ADSA-21PYFU		Logout	3/13/2009 11:11 AM	3/13/2009 11:11 AM	Joanne Brown
ADSA-21PYFU		Login	3/13/2009 11:11 AM	3/13/2009 11:11 AM	Joanne Brown

Step 2: Download a WSDL file

So far our Java program simply performs a login and logoff. But to do something really useful as for example creating a record or querying for data, we need to use the provided WSDL files.

In the CRM On Demand application, navigate to "Admin → Web Services Administration". In the drop down list "Select Service" select the "Web Services 1.0" and press the "Go" button.

There are some differences between "Web Services 1.0" and "Web Services 2.0", and the documentation will give you more details about those. Some of the basic differences consist in data type casting, the support of attachments and a couple of new methods. I will use Web Services 1.0", as those give access to the child entities as well. But the usage of "Web Services 2.0" won't be too much different.

For this tutorial we will use the Contact Web Service. Select "Contact" in the "WSDL Object" box and press "Download Custom WSDL".

Using the Custom WSDLs ensures that all your custom fields are contained in the WSDL file AND that the correct SOAP address location is contained. If you use "Generic WSDL", you won't be able to use that without some manual tweaking – that is a common source of errors!

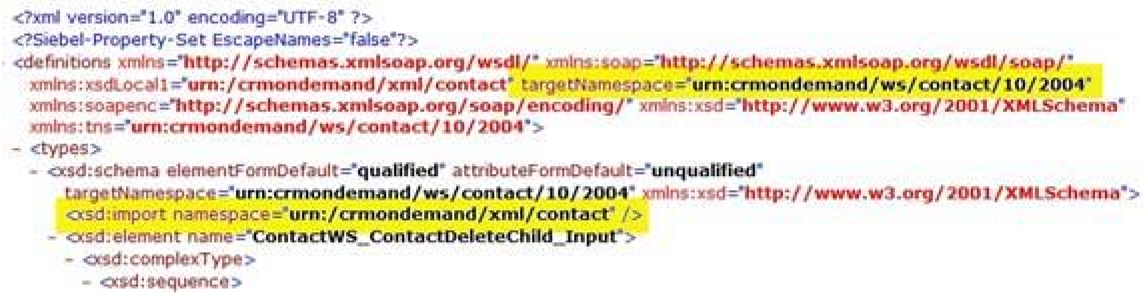
The WSDL file will open in your web browser. Save it as "Contact.wsdl" to your "D:\Axis" directory (which we created in the beginning of the tutorial).

Step 3: Generate Java Proxies

Java code can't work with a WSDL file directly, so we need a mechanism to generate Java classes from our WSDL file. An easy mechanism to perform that comes as part of the Apache AXIS wsdl2java utility; wsdl2java is part of the AXIS libraries which you downloaded at the beginning. We will generate the Java code using a small command line program. In your "D:\Axis" directory create a new file named "wsdl_contact.bat". Edit it using notepad and paste the following text:

```
java -cp .;d:\axis\lib\wsdl4j-1.5.1.jar;d:\axis\lib\saaj.jar;d:\axis\lib\jaxrpc.jar;d:\axis\lib\axis-ant.jar;d:\axis\lib\log4j-
1.2.8.jar;d:\axis\lib\commons-discovery-0.2.jar;d:\axis\lib\commons-logging-
1.0.4.jar;d:\axis\lib\axis.jar;d:\axis\lib\activation.jar;d:\axis\lib\mailapi.jar org.apache.axis.wsdl.WSDL2Java -
N"urn:crmondemand/ws/contact/10/2004"="crmondemand.ws.contact" -
N"urn:/crmondemand/xml/contact"="crmondemand.xml.contact" contact.wsdl
```

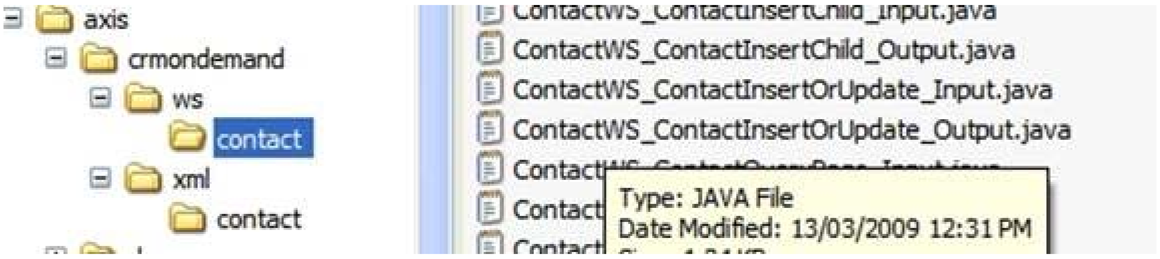
If you are using e.g. the opportunity WSDL file, you would replace "contact" with "opportunity". The exact values you need to choose ("urn:crmondemand/ws/contact/10/2004" and "urn:/crmondemand/xml/contact) are part of the WSDL file:



```
<?xml version="1.0" encoding="UTF-8" ?>
<?Siebel-Property-Set EscapeNames="false"?>
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:xsdLocal1="urn:/crmondemand/xml/contact" targetNamespace="urn:crmondemand/ws/contact/10/2004"
  xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:tns="urn:crmondemand/ws/contact/10/2004">
- <types>
  - <xsd:schema elementFormDefault="qualified" attributeFormDefault="unqualified"
      targetNamespace="urn:crmondemand/ws/contact/10/2004" xmlns:xsd="http://www.w3.org/2001/XMLSchema">
      <xsd:import namespace="urn:/crmondemand/xml/contact" />
    - <xsd:element name="ContactWS_ContactDeleteChild_Input">
      - <xsd:complexType>
        - <xsd:sequence>
```

If you are using "Web Services 2.0", you need to look up those values in the WSDL file, as namespaces are a bit different in that case!

Now execute the batch file wsdl_contact.bat. This will generate the following folder structure in your "D:\Axis" directory. If it doesn't, you probably typed the namedspaces in the wsdl wrong – check those again in that case.



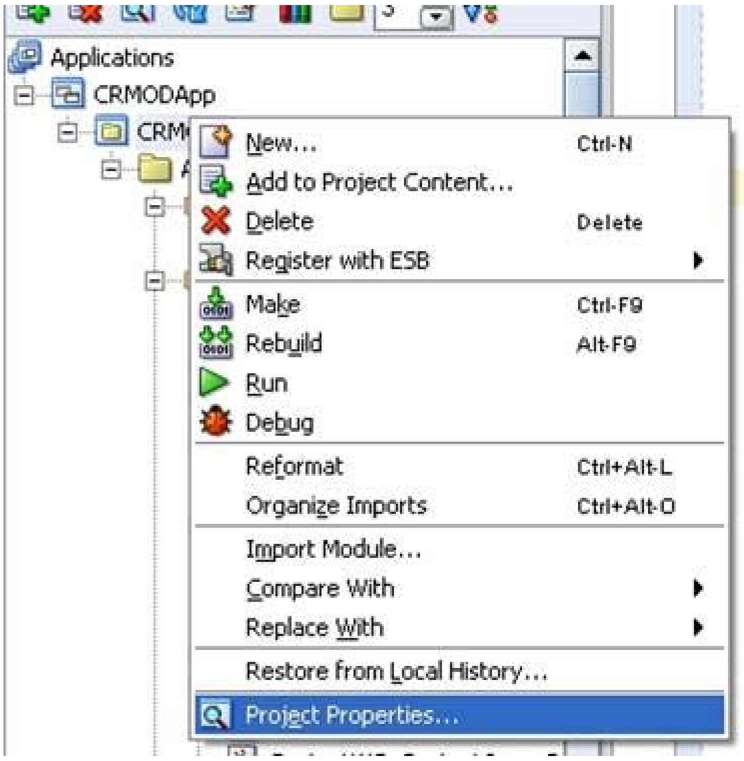
axis
crmondemand
ws
contact
xml
contact
ContactWS_ContactInsertChild_Input.java
ContactWS_ContactInsertChild_Output.java
ContactWS_ContactInsertOrUpdate_Input.java
ContactWS_ContactInsertOrUpdate_Output.java
Contact
Contact
Contact
Type: JAVA File
Date Modified: 13/03/2009 12:31 PM

The folders contain Java source files, which we will use in the next steps to talk to CRM On Demand.

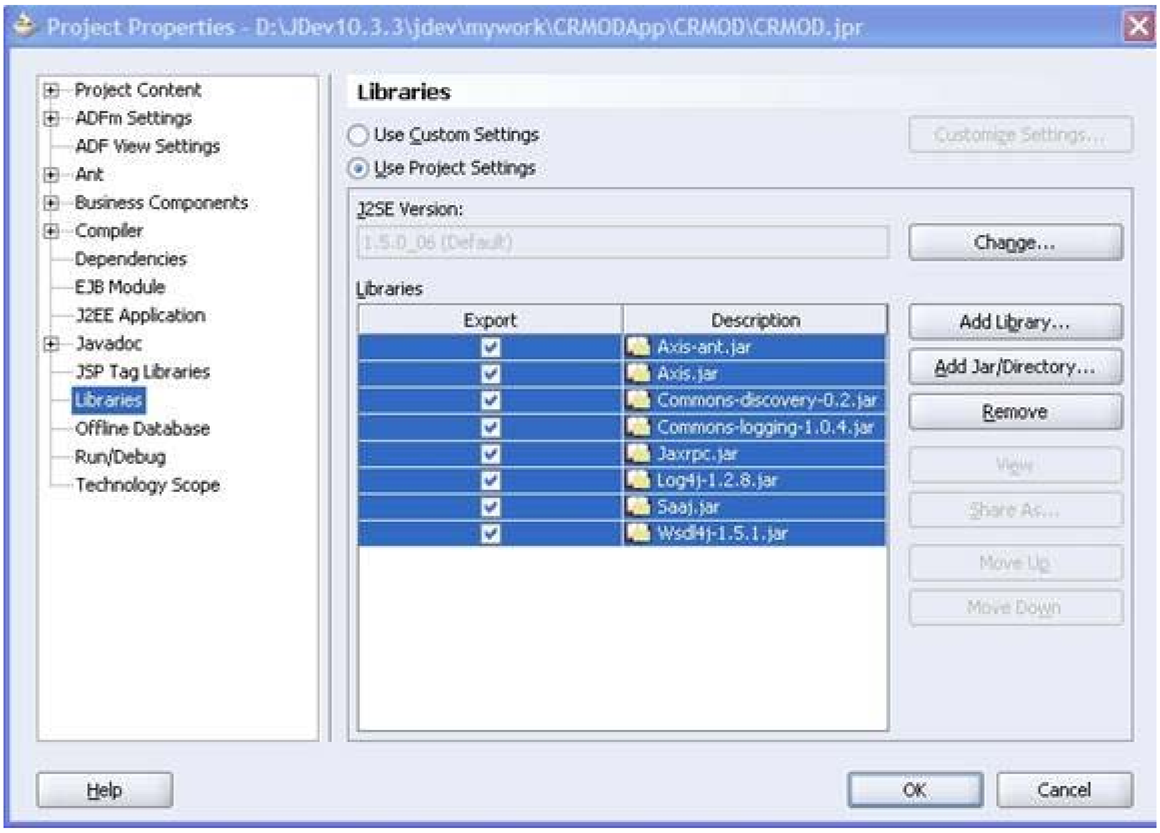
Step 4: Import Java Proxies

1) Add required Axis libraries to the project

Now we are going to import the generated Java proxy files into JDeveloper. To do so, JDeveloper should have access to the Axis libraries. Select your "CRMOD" project in the Application Navigator and from the context menu select "Project Properties".



Select "Libraries" on the left, and press the "Add Jar/Directory" button. In the following dialog, select all the files in the "D:\Axis\lib" directory. Press "OK" to leave the dialog.



Libraries

( ) Use Custom Settings
(●) Use Project Settings

J2SE Version: 1.5.0_06 (Default)

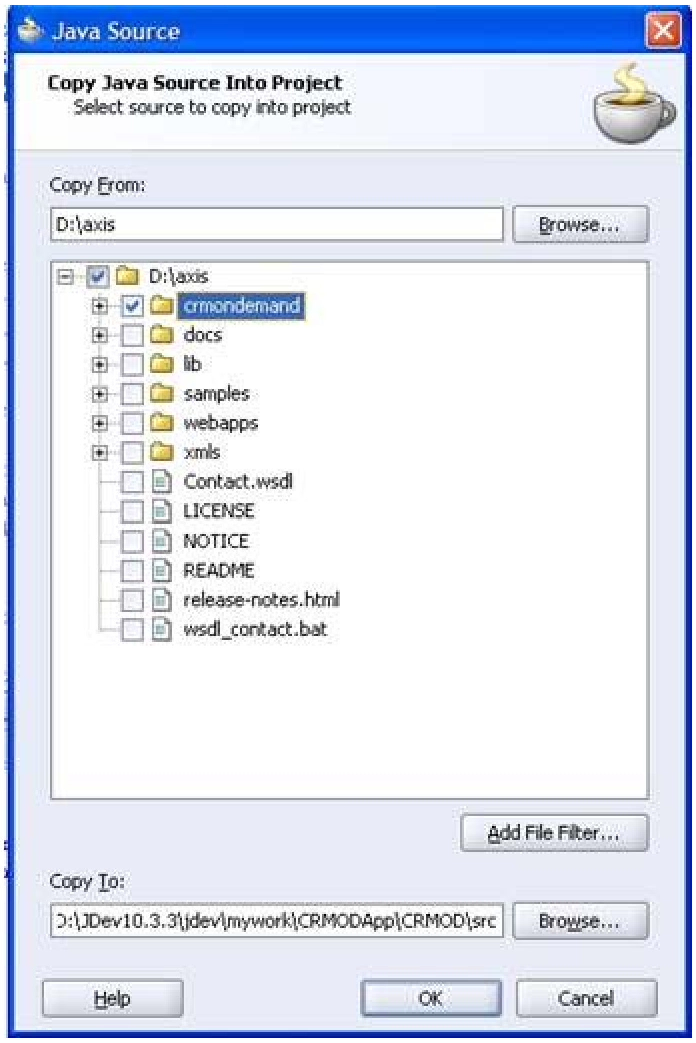
Libraries

Export	Description
✓	Axis-ant.jar
✓	Axis.jar
✓	Commons-discovery-0.2.jar
✓	Commons-logging-1.0.4.jar
✓	Jaxrpc.jar
✓	Log4j-1.2.8.jar
✓	Saaj.jar
✓	Wsdl4j-1.5.1.jar

2) Add Java Classes

Select your "CRMOD" project in the Application Navigator, and select "File / Import" from the JDeveloper menu. In the following dialog, select "Java Source" and press "OK".

Select in the following dialog to your "D:\Axis" directory and in the next dialog box choose only the "crmondemand" subfolder (as shown on the next screenshot). ***It is important that you first select the "D:\Axis" folder and not directly the "D:\Axis\crmondemand" folder. Otherwise you namespaces would miss the "crmondemand" prefix, and would you need to modify them manually!***

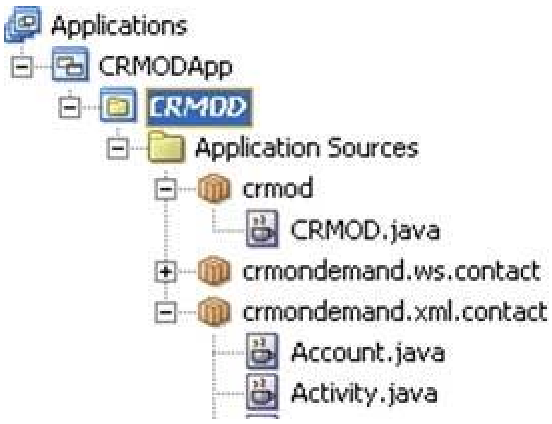

Java Source
Copy Java Source Into Project
Select source to copy into project

Copy From: D:\axis

- D:\axis [✓]
  - crmondemand [✓]
  - docs
  - lib
  - samples
  - webapps
  - xmls
  - Contact.wsdl
  - LICENSE
  - NOTICE
  - README
  - release-notes.html
  - wsdl_contact.bat

Copy To: D:\JDev10.3.3\jdev\mywork\CRMODApp\CRMOD\src

If you followed the instructions, you will result in the following folder structure:


- Applications
  - CRMODApp
    - CRMOD
      - Application Sources
        - crmod
          - CRMOD.java
        - crmondemand.ws.contact
        - crmondemand.xml.contact
          - Account.java
          - Activity.java

Step 5: Create a new contact using Java

1) Now we add the code to actually create a new contact:

Enter a couple of empty lines after the line "System.out.println(jsessionid);" in the main method.

Paste here the following code:

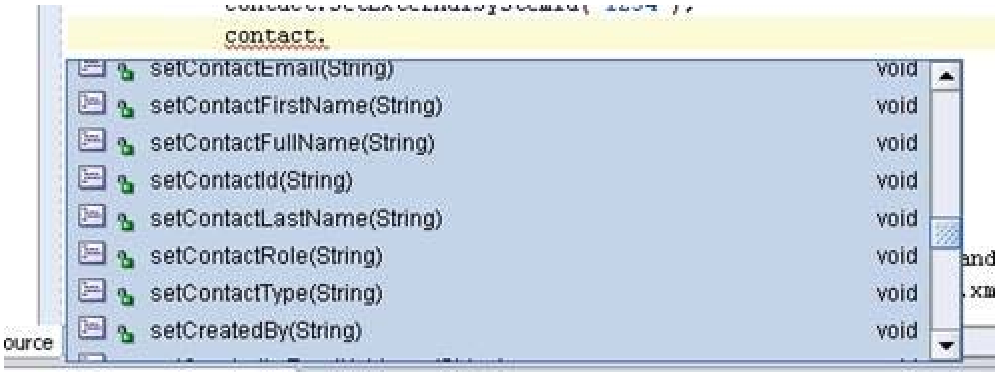
```
endpoint =  "https://secure-ausomxdsa.crmondemand.com/Services/Integration" + ";jsessionid=" + jsessionid;
URL urlAddr =  new java.net.URL( endpoint);


crmondemand.ws.contact.Contact service = new crmondemand.ws.contact.ContactLocator();
crmondemand.ws.contact.Default_Binding_Contact stub = service.getDefault(urlAddr);
crmondemand.ws.contact.ContactWS_ContactInsert_Input contactlist = new crmondemand.ws.contact.ContactWS_ContactInsert_Input();
crmondemand.ws.contact.ContactWS_ContactInsert_Output outlist = new crmondemand.ws.contact.ContactWS_ContactInsert_Output();
crmondemand.xml.contact.Contact[] contacts = new crmondemand.xml.contact.Contact[1];
crmondemand.xml.contact.Contact contact = new crmondemand.xml.contact.Contact();

contact.setContactFirstName("Joerg");
contact.setContactLastName("Wallmueller");
contact.setExternalSystemId("1234");

contacts[0] = contact;
contactlist.setListOfContact(contacts);
contactlist.setEcho("off");  // Don't forget this line, as the Web service call would fail!
outlist = stub.contactInsert(contactlist);

crmondemand.xml.contact.Contact[] results = new crmondemand.xml.contact.Contact[1];
crmondemand.xml.contact.Contact result = new crmondemand.xml.contact.Contact();
results = outlist.getListOfContact();
result = results[0];
System.out.println(result.getContactId());
```

You will see that I only specified minimal information for the contact to be created. The Field "ExternalSystemId" needs to be unique. Of course you can automate the generation of a unique id using e.g. the current milliseconds or the random method.

If you want to set more fields for the contact, do as follows:

Create a new line after the line "contact.setExternalSystemId("1234");".

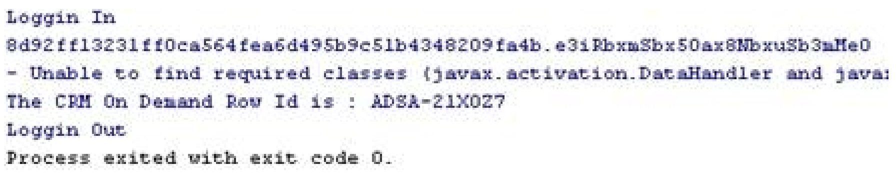
Type "contact." and now stop typing, wait one second: you will see a list of all the following methods, from which you can choose from:



```
contact.
setContactEmail(String)          void
setContactFirstName(String)      void
setContactFullName(String)       void
setContactId(String)             void
setContactLastName(String)       void
setContactRole(String)           void
setContactType(String)           void
setCreatedBy(String)             void
```

2) Execute your code

If you execute your program by selecting "Run" from the project's context menu, a new contact will be created in CRM On Demand. The CRM On Demand internal RowId will be returned in the debug window:



```
Loggin In
8d92ff13231ff0ca564fea6d495b9c51b4348209fa4b.e3iRbxmSbx50ax8MbxuSb3mMe0
- Unable to find required classes (javax.activation.DataHandler and java:
The CRM On Demand Row Id is : ADSA-21X0Z7
Loggin Out
Process exited with exit code 0.
```

Step 6: Query for an existing contact

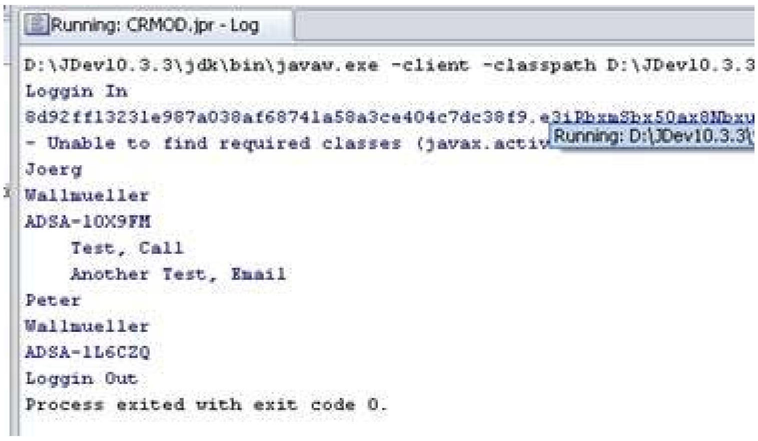
In the previous example we created a new contact. Let's see now what you need to do if you want to query for an existing contact and related child entities (e.g. the list of activities).

1) Replace the code from Step 5) with the following one:

```
endpoint =  "https://secure-ausomxdsa.crmondemand.com/Services/Integration" + ";jsessionid=" + jsessionid;
URL urlAddr =  new java.net.URL( endpoint);

crmondemand.ws.contact.Contact service = new crmondemand.ws.contact.ContactLocator();
crmondemand.ws.contact.Default_Binding_Contact stub = service.getDefault(urlAddr);
crmondemand.ws.contact.ContactWS_ContactQueryPage_Input contactlist = new crmondemand.ws.contact.ContactWS_ContactQueryPage_Input();
crmondemand.ws.contact.ContactWS_ContactQueryPage_Output outlist = new crmondemand.ws.contact.ContactWS_ContactQueryPage_Output();
crmondemand.xml.contact.Contact[] contacts = new crmondemand.xml.contact.Contact[1];
crmondemand.xml.contact.Contact contact = new crmondemand.xml.contact.Contact();
crmondemand.xml.contact.Activity[] activities = new crmondemand.xml.contact.Activity[1];
crmondemand.xml.contact.Activity activity = new crmondemand.xml.contact.Activity();

activity.setSubject("");
activity.setType("");
activity.setRowStatusOld("");
activities[0] = activity;

contact.setContactLastName("='Wallmueller'");
contact.setContactFirstName("");
contact.setContactId("");
contact.setListOfActivity(activities);

contacts[0] = contact;
contactlist.setPageSize("10");
contactlist.setUseChildAnd("false");
contactlist.setStartRowNum("0");
contactlist.setListOfContact(contacts);
outlist = stub.contactQueryPage(contactlist);

crmondemand.xml.contact.Contact[] results =
   new crmondemand.xml.contact.Contact[1];
results = outlist.getListOfContact();

crmondemand.xml.contact.Activity[] activitiesout =
   new crmondemand.xml.contact.Activity[1];

int lenC = results.length;
            if (lenC > 0) {
                for (int i = 0; i < lenC; i++) {
                  System.out.println(results[i].getContactFirstName());
                  System.out.println(results[i].getContactLastName());
                  System.out.println(results[i].getContactId());

                  int lenA = results[i].getListOfActivity().length;
                      if (lenA > 0) {
                          for (int j = 0; j < lenA; j++) {
                                activitiesout = results[i].getListOfActivity();
                                System.out.println("    " + activitiesout[j].getSubject() + ", " + activitiesout[j].getType());
                          }
                        }
```

```
                    }

            }
```

If you create a couple of Activities for your contact and execute it, something similar as in the screenshot below will be returned:


Running: CRMOD.jpr - Log

```
D:\JDev10.3.3\jdk\bin\javaw.exe -client -classpath D:\JDev10.3.3
Loggin In
8d92ff13231e987a038af68741a58a3ce404c7dc38f9.e3iRbxmSbx50ax8Nbxu
- Unable to find required classes (javax.activ Running: D:\JDev10.3.3\
Joerg
Wallmueller
ADSA-10X9FM
    Test, Call
    Another Test, Email
Peter
Wallmueller
ADSA-1L6CZQ
Loggin Out
Process exited with exit code 0.
```

Conclusion

Hopefully you saw that it is not too hard to invoke CRM On Demand Web Services from Java code. Based on the techniques outlined here, you can start build more sophisticated Java programs, Web applications or portals which access CRM On Demand.