Using the OnDemand Web Enablement Kit (ODWEK) ODServer.xmlParse() method

Last updated: April 27, 2020

The ODWEK Java APIs are intended to provide end-user client functionality, most commonly used to allow users to query the Content Manager OnDemand archive and retrieve associated documents. However, business requirements might dictate a need to access information that is considered to be administrative in nature. Although the Java APIs were never intended for this purpose, there is an API method to satisfy that requirement.

The ODServer.xmlParse() method basically mirrors the functionality of the Content Manager OnDemand server command line program called ARSXML. It accepts XML as its input and returns XML as its result where applicable.

The caller of the function is required to provide the input XML and also to parse the output to extract the information that is sought. That might seem a bit daunting but it is actually not too difficult. The following example demonstrates one way to use the xmlParse() method. This example will create the XML needed to perform the equivalent of the ARSXML EXPORT function, to export data from a folder object, parse the output XML, and display the Content Manager OnDemand users and groups that have permissions to the specified Content Manager OnDemand folder.

Preparing to use the example - Update variables

Before using the sample code further down below, a few variables need to be changed to match the environment where the code will be executed. They are found at the beginning of the main function. These variables could easily be accepted as arguments from the command line but for this example, they are hard-coded into the sample code. Specifically, consider updating the following in the sample code down below before compiling it:

```
/* to create the xml used as input to xmlParse
 * In this example, a folder node will
 * be created with a name of 'Credit Card Statements' i.e.
 * <?xml version="1.0" encoding="UTF-8" standalone="no"?>
 *    <onDemand xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance>
 *       <folder name=" Credit Card Statements "/>
 *    </onDemand>
 * Change the ODNodeName and ODName values to match environment */
String ODNodeName = "folder";
String ODName = "Credit Card Statements";

/* server details */
String serverName = "odserver.yourcompany.com";
int port = 1445;
```

```
String userName = "admin";
String password = "secret";

/* xmlParse details for exporting
/* update to match your system's environment.
/* per the Javadocs 4 indicates an export */
int action = 4;
boolean continueOnError = false;
boolean updateExisting = false;
/* per the Javadocs, 2 indicates permissions will be included
 * in the output xml */
int exportFlags = 2;
int encoding = 1208;

/* Values to extract from the output of xmlParse for printing
 * In this example, for the <permission> node(s),
 * print the 'user' & 'group' attribute values.
 * Must match valid attributes for the specified node.
 * See the OnDemand.xsd for valid attributes */
String ODTargetNode = "permission";
String[] NodeAttrs = {"user", "group"};
```

As described in the comments, this sample code will create the XML required to export a folder called "Credit Card Statements". The generated XML will be submitted to the ODServer.xmlParse() function which will export the folder and its permission nodes. The test case will then target the user and group attributes of the permission nodes and print the values to the console.

See the ODWEK Javadocs for a listing of the properties that can be specified to xmlParse().

NOTE: The Content Manager OnDemand server and ODWEK must both be at the same version or the following error is issued:

"The client and server are incompatible. Reinstallation of the product is required."

This example represents only one way to access the XML data and is not intended to be a solution to every scenario. Feel free to modify it as is appropriate for your environment. It is provided AS-IS and without warranty.

The sample code – TcXML.java

```
import java.io.IOException;
import java.io.StringReader;
import java.io.StringWriter;
import java.util.Properties;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import javax.xml.transform.OutputKeys;
import javax.xml.transform.Transformer;
import javax.xml.transform.TransformerConfigurationException;
import javax.xml.transform.TransformerException;
import javax.xml.transform.TransformerFactory;
import javax.xml.transform.dom.DOMSource;
```

```
import javax.xml.transform.stream.StreamResult;
import org.xml.sax.InputSource;
import org.xml.sax.SAXException;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NodeList;

import com.ibm.edms.od.ODConfig;
import com.ibm.edms.od.ODServer;
import com.ibm.edms.od.ODConstant;
import com.ibm.edms.od.ODCallback;

public class TcXML {

  public static void main(String argv[]) {

  /* Create the xml used as input to xmlParse()
   * In this example, a folder node will
   * be created with a name of 'Credit Card Statements' i.e.
   *
   * <?xml version="1.0" encoding="UTF-8" standalone="no"?>
   *    <onDemand xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
   *       <folder name="Credit Card Statements"/>
   *    </onDemand>
   *
   *    You could also just read this from a file
   *
   * Change the ODNodeName and ODName values to match environment */
    String ODNodeName = "folder";
    String ODName = "Credit Card Statements";

  /* server details */
    String serverName = "odserver.yourcompany.com";
    int port = 1445;
    String userName = "admin";
    String password = "secret";

  /* xmlParse details for exporting */
  /* update to match your system's environment.
  /* per the Javadocs, 4 indicates export */
    int action = 4;
    boolean continueOnError = false;
    boolean updateExisting = false;
  /* per the Javadocs, 2 indicates permissions will be included in the output xml */
    int exportFlags = 2;
    int encoding = 1208;

  /* Values to extract from the output of xmlParse for printing
   * In this example, for the <permission> node(s),
   * print the 'user' & 'group' attribute values.
   * Must match valid attributes for the specified node */
    String ODTargetNode = "permission";
    String[] NodeAttrs = {"user", "group"};


  try {
      String xmlin = createXML( ODNodeName, ODName );
      System.out.println( xmlin );


      byte[] xmlout = submitXMLToOD(xmlin,
                                    serverName,
                                    port,
```

```
                                    userName,
                                    password,
                                    action,
                                    continueOnError,
                                    updateExisting,
                                    exportFlags,
                                    encoding);

    if (xmlout != null)
      {
       //System.out.println("Byte contents:");
       //System.out.println(new String(xmlout));
         printAttributesFromNode(new String(xmlout), ODTargetNode, NodeAttrs);
      }

  } catch (Exception e) {
      e.printStackTrace();
  }
}

public static String createXML( String nodeName, String objName )
{
  DocumentBuilderFactory docFactory;
  DocumentBuilder docBuilder;
  Element rootElement;

  try {
    docFactory = DocumentBuilderFactory.newInstance();
    docBuilder = docFactory.newDocumentBuilder();

   // root element
    Document doc = docBuilder.newDocument();
    rootElement = doc.createElement("onDemand");
    rootElement.setAttribute("xmlns:xsi",
                             "http://www.w3.org/2001/XMLSchema-instance");

    doc.appendChild(rootElement);

    // 1st level child element
    Element odObjChildElem = doc.createElement(nodeName);
    rootElement.appendChild(odObjChildElem);

    odObjChildElem.setAttribute("name", objName);

    // create XML
    TransformerFactory transformerFactory = TransformerFactory.newInstance();
    Transformer transformer = transformerFactory.newTransformer();
    transformer.setOutputProperty(OutputKeys.ENCODING, "UTF-8");
    // make it pretty
    transformer.setOutputProperty(OutputKeys.INDENT, "yes");
    transformer.setOutputProperty("{http://xml.apache.org/xslt}indent-amount", "2");
    DOMSource source = new DOMSource(doc);
    StringWriter writer = new StringWriter();
    StreamResult result = new StreamResult(writer);
    transformer.transform(source, result);

    return writer.toString();
  } catch (ParserConfigurationException pce) {
      pce.printStackTrace();
  } catch (TransformerConfigurationException tce) {
      tce.printStackTrace();
  } catch (TransformerException e) {
      e.printStackTrace();
```

```
    }
      catch (Exception e) {
         System.out.println("exception: " + e);
         e.printStackTrace();
    }
    return null;
    }

  public static byte[] submitXMLToOD(String inputXML,
                                     String serverName,
                                     int port,
                                     String userName,
                                     String password,
                                     int action,
                                     boolean continueOnError,
                                     boolean updateExisting,
                                     int exportFlags,
                                     int exportEncoding)
  {
    ODConfig odConfig;
    ODServer odServer;
    byte[] xmlout = null;

    try {
        odConfig = new ODConfig(ODConstant.PLUGIN, //AfpViewer
                    ODConstant.APPLET,             //LineViewer
                    null,                          //MetaViewer
                    200,                           //MaxHits
                    "/applets",                    //AppletDir
                    "ENU",                         //Language
                    "/tmp",                        //TempDir
                    "/tmp/trace",                  //TraceDir
                    1);                            //TraceLevel
        odServer = new ODServer(odConfig);
        odServer.initialize("TcXML.java");
        odServer.setPort(port);
        odServer.logon(serverName, userName, password);

        Properties xmlProperties = new Properties();

xmlProperties.setProperty("ContinueOnError",Boolean.toString(continueOnError));
        xmlProperties.setProperty("UpdateExisting", Boolean.toString(updateExisting));
        xmlProperties.setProperty("ExportFlags", Integer.toString(exportFlags));
        xmlProperties.setProperty("ExportEncoding", Integer.toString(exportEncoding));

        ODCallback odcb = new ODCallback();

        xmlout = odServer.xmlParse(inputXML,
                                   action,
                                   xmlProperties,
                                   odcb);
    } catch (Exception e) {
      e.printStackTrace();
    }
    return xmlout;
    }

  public static void printAttributesFromNode(String xml,
                                             String node,
                                             String[] targetAttrs)
  {
    int maxFieldSize = 30;
    int numSpaces;
```

```
    InputSource is = new InputSource();
    is.setCharacterStream(new StringReader(new String(xml)));

    DocumentBuilderFactory docfactory = DocumentBuilderFactory.newInstance();
    DocumentBuilder docbuilder;
    try {
         docbuilder = docfactory.newDocumentBuilder();
         Document doc = docbuilder.parse(is);
         NodeList topNL = doc.getChildNodes();

         for (int i = 0; i < topNL.getLength(); i++) {
             if (topNL.item(i) instanceof Element)
             {
                Element root = (Element) topNL.item(i);
                NodeList targetNL = root.getElementsByTagName(node);
                System.out.println("---Node Name: <" + node + ">\n");
                System.out.println("---Requested <" + node + "> Attributes:");

                /* print headers */
                for(String targetAttr : targetAttrs)
                {
                   numSpaces = maxFieldSize - targetAttr.length();
                   for(int s = 0; s < numSpaces;s++){ System.out.print(" "); }
                   System.out.print(targetAttr);
                }
                System.out.println("");

                for(int s = 0; s < maxFieldSize * targetAttrs.length; s++)
                   { System.out.print("-"); }
                System.out.println("");

               /* print attribute values */
                for (int j = 0; j < targetNL.getLength(); j++)
                {
                   if (targetNL.item(j) instanceof Element)
                   {
                      Element targetElem = (Element) targetNL.item(j);
                      for(String attr : targetAttrs)
                      {
                         String attrVal = targetElem.getAttribute(attr);
                         if(attrVal != null)
                         {
                            numSpaces = maxFieldSize - attrVal.length();
                            for(int s = 0; s < numSpaces;s++){ System.out.print("
"); }
                            System.out.print(attrVal);
                         }
                      }
                      System.out.println("");
                   }
                }
           }
        }
        } catch (ParserConfigurationException pce) {
          pce.printStackTrace();
        } catch (SAXException se) {
        se.printStackTrace();
        } catch (IOException e) {
        e.printStackTrace();
        }
  }
}
```