

# Dököll Solutions, Inc.

## Application Development

<b>Project Name:</b>	Oracle JDeveloper, IBM Domino Designer, DB2, XML Data Insert	<b>Start Date:</b>	2015.11.09.12.01.AM
<b>Purpose:</b>	Combine Java Code, Read HTTP XML	<b>End Date:</b>	2015.11.09.12.01.AM
<b>Language:</b>	Java, XML	<b>Environment:</b>	IBM Domino Designer, DB2, Oracle JDeveloper
<b>Employee Name:</b>	Dököll Solutions	<b>Employee ID:</b>	Dököll Solutions
<b>Task(s):</b>	Transmit XML via URL Connection	<b>Document:</b>	Journal Entries

## **IBM Domino Designer, Oracle JDeveloper Domino URL Connection, DB2 Insert**

### **System Requirements**

<b>Software/Environment</b>	<b>Language/Technology</b>	<b>Protocol/Framework/Platform</b>
Microsoft Windows 7, 8, 10	VBScript, Batch	Active Directory, Operating System
Microsoft Internet Explorer	N/A	TCPIP, HTTP, Browser
Google Chrome	N/A	TCPIP, HTTP, Browser
Mozilla FireFox	N/A	TCPIP, HTTP, Browser
Oracle JDeveloper 12.xx	Java	HTTP, TCPIP, IDE
Oracle Integrated WebLogic 12.x	Console	HTTP, TCPIP, Server
IBM DB2 Express-C	SQL	SQL Database Server
IBM Data Studio	SQL	SQL Database Management, IDE
IBM Notes Domino Designer 8.5.3, 9, 10	JavaAgent, XML	HTTP, TCPIP, IDE

## **Disclaimer:**

Information contained in the following is presented as is. This tutorial assumes you have basic programming and software configuration knowledge. All tutorials are based on IBM Notes Domino or Oracle Fusion Middleware, including and not limited to items stated in the System Requirements. Should you need to familiarize yourself with IBM Domino Designer or Oracle JDeveloper environments, prior to continuing, stop now and see our Journal Entries page on our website: [www.dokollsolutionsinc.com](http://www.dokollsolutionsinc.com) for additional support...

## **Foreword:**

Samples included in this Journal Entries document are part of a series, be sure to consult other versions of the documentation to benefit in full.

## **Introduction:**

In this Journal Entries episode, we will piggy-back on code we have worked on to create an XML document to a URL using IBM Domino Designer JavaAgent. The goal is to make the data available to IBM DB2, via Oracle JDeveloper Java program... We will do this Live, and we will document our steps as we find information, post errors, and so on- We suspect existing code samples within our IBM Domino Designer environment are present to do the Insert Statement, or we should be able to salvage a CSV to DB2 option to do the trick. As we continue our research, we are going to ensure that all areas of interest are highlighted for your convenience. Do stay tuned!

+++++>>

2015.11.08.11.24.PM

We've been out of it for a while, prior to typing up this Journal Entries document, we realize how consumed we have been programming other projects, in HTML5, jQuery, PHP 'Cloud' environments. We did collect some data for all to see, you will be able to cue into what we've been up to, when we are done with our ADF, Xpages, Java Journal Entries documentation. For now, we're taking a quick look to see where we are with the existing IBM Domino Designer and Oracle JDeveloper XML data project for IBM DB2.

Let's take a moment to run through each environment to explore our options.

+++++>>

2015.11.09.12.01.AM

Here is an accidental find, a piece of code found in our existing IBM Domino Designer App that can also be used to grab XML via URL. Will keep it here since the other samples read CSV into XML for directory submissions, our readers can also find great use for this...

```
package com.dokoll.solutions.inc.search;

import java.net.*;
import java.io.*;


public class CreateXmlOnDir {
    public void doGetXMLFromURL() {
```

```

try {

    //TO DO: Send ip and other info to back-end if page firing
    //this button code

    //Get XML file from URL.
    URL xmlURL = new URL("http", "localhost", 80,
                          "/docucontent.nsf/dashboardchartdata.xml");
    // Set up Connection to URL, grab file
    URLConnection conn3ctXML = xmlURL.openConnection();
    // connect to URL. grab file
    conn3ctXML.connect();

    // Build URL file into new XML file
    // ...
    PrintWriter printerWriter = new PrintWriter(new FileWriter(
        "c:\\temp\\XML_DATA\\documentbycategoryfromurl.xml"));

;

    // Read URL data into new XML file
    BufferedReader bufferRead = new BufferedReader(
        new InputStreamReader(conn3ctXML.getInputStream()));
    // ...
    String LineRead = bufferRead.readLine();

    // ...
    while (LineRead != null) {
        printerWriter.println(LineRead);
        LineRead = bufferRead.readLine();
    }

    printerWriter.close();

} catch (Exception e) {
    System.out.println("Error: " + e);
}
}

```

If you need to see how this is done as a CSV to XML, grab the previous version of these Journal Entries, search by date to make it easier. We will not be posting the original CSV-specific code here, trying to keep the current document light.

We are going to give it a go with the above to test it and see what we can do with it. We can pluck the CSV to DB2-specific chunk to do the insert. For now, let's put below sample on the back burner. But before we do that, we are going to compare the two.

Below seems like it will do the work, no problem, we will need to spit the values from the XML that can later be programmed into CSV strings, to be split/inserted in an Array, then be submitted to DB2. The above sample grabs the XML, splits it into available strings for the Array to submit. Should be a fun trial and error type of thing to see which one will work the best for us...

The screenshot shows the IBM Domino Designer interface with two code editors open. The left editor contains the code for `UserDefinedURLAccessBean.java`, and the right editor contains the code for `VersionDownloadFeedJavaBean.java`. Both files are in Java syntax highlighting mode.

```
File Edit Source Refactor Create Design Navigate Tools Search Project Run Window Help
UserDefinedURLAccessBean.java  VersionDownloadFeedJavaBean.java
public String getFetchItem() {
    try {
        //grab XML file @ once...
        DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
        DocumentBuilder db = dbf.newDocumentBuilder();
        Document doc = db
            .parse("http://localhost/docucontent.nsf/javaagenthttpversionchgreader.xml");
        doc.getDocumentElement().normalize();
        NodeList nodeLst = doc.getElementsByTagName("allusers");
        //loop through element list and fish out unique results
        for (int s = 0; s < nodeLst.getLength(); s++) {
            Node fstNode = nodeLst.item(s);
            if (fstNode.getNodeType() == Node.ELEMENT_NODE) {
                Element fstElmnt = (Element) fstNode;
                NodeList fstNmElmntLst = fstElmnt
                    .getElementsByTagName("itemsearch");
                Element fstNmElmnt = (Element) fstNmElmntLst.item(0);
                NodeList fstNm = fstNmElmnt.getChildNodes();
                //System.out.println("itemsearch : "
                //    + ((Node) fstNm.item(0)).getNodeValue());
                FetchItem = ((Node) fstNm.item(0)).getNodeValue().toString();
                System.out.println("getFetchItem() XML DATA, Found in VersionDownloadFeedJavaBean.java");
            }
        }
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

if you recall, this particular sample, seen in the screenshot, was taken from IBM Domino Designer, and was tested in Oracle JDeveloper environment, but the version advertised below has been modified to include the Insert Statement to submit to IBM DB2...

```
package com.dokoll.view.backing;

import com.dokoll.solutions.inc.oracle_src.DBConnector;

import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

import java.io.*;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.MalformedURLException;
import java.net.URL;
import java.net.*;

import javax.faces.context.FacesContext;
import javax.servlet.http.HttpServletResponse;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
```

```
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.Node;
import org.w3c.dom.NodeList;

import org.xml.sax.SAXException;
import org.xml.sax.SAXParseException;

import oracle.adf.view.rich.component.rich.RichDocument;
import oracle.adf.view.rich.component.rich.RichForm;
import oracle.adf.view.rich.component.rich.data.RichTable;
import oracle.adf.view.rich.component.rich.input.RichInputNumberSpinbox;
import oracle.adf.view.rich.component.rich.input.RichInputText;
import oracle.adf.view.rich.component.rich.input.RichTextEditor;
import oracle.adf.view.rich.component.rich.nav.RichButton;
import oracle.adf.view.rich.component.rich.output.RichOutputText;

//...
//Start of App
public class Jdlotusfeed {
    private RichForm f1;
    private RichDocument d1;
    private RichButton b1;
    private String ShowData;
    private String UserID;
    private String PageID;
    private String Priority;
    private String Issues;
    private RichInputText it1;
    private RichInputNumberSpinbox ins1;
    private RichInputNumberSpinbox ins2;
    private RichInputText it2;
    private RichInputText it3;
    private RichInputText it4;
    private RichTable t1;
    private RichOutputText ot2;
    private RichOutputText ot3;
    private RichOutputText ot4;
    private RichOutputText ot5;
    private RichTextEditor rte1;

    public String getShowData() {
        return ShowData;
    }

    public void setShowData(String showData) {
        ;
    }
}
```

```
        this.ShowData = showData;  
    }  
  
    public String getUserId() {  
        return UserID;  
    }  
  
    public void setUserId(String userID) {  
        ;  
        this.UserID = userID;  
    }  
  
    public String getPageID() {  
        return PageID;  
    }  
  
    public void setPageID(String pageID) {  
        ;  
        this.PageID = pageID;  
    }  
  
    public String getPriority() {  
        return Priority;  
    }  
  
    public void setPriority(String priority) {  
        ;  
        this.Priority = priority;  
    }  
  
    public String getIssues() {  
        return Issues;  
    }  
  
    public void setIssues(String issues) {  
        ;  
        this.Issues = issues;  
    }  
}
```

```

public void setF1(RichForm f1) {
    this.f1 = f1;
}

public RichForm getF1() {
    return f1;
}

public void setD1(RichDocument d1) {
    this.d1 = d1;
}

public RichDocument getD1() {
    return d1;
}

public void setB1(RichButton b1) {
    this.b1 = b1;
}

public RichButton getB1() {
    return b1;
}

//Button code
@SuppressWarnings("oracle.jdeveloper.java.nested-assignment")
public String b1_action() throws MalformedURLException, IOException {
    // Add event code here...

//...
//TESTING PURPOSES INTO ORACLE WEBLOGIC SERVER CONSOLE
//...
try {
    DocumentBuilderFactory docBuilderFactory = DocumentBuilderFactory.newInstance();
    DocumentBuilder docBuilder = docBuilderFactory.newDocumentBuilder();

    //Get XML file from URL and submit to Oracle WebLogic Console
    //...
    Document doc =
    docBuilder.parse("http://localhost/docucontent.nsf/javaagentdataforwebnewshttp.xml");

    // ...
    doc.getDocumentElement().normalize();
}

```

```

System.out.println("Root element " + doc.getDocumentElement().getNodeName());

NodeList listOfIssues = doc.getElementsByTagName("person");
int totalSheets = listOfIssues.getLength();
System.out.println("Total Issues : " + totalSheets);

for (int s = 0; s < listOfIssues.getLength(); s++) {

    Node currPersonNode = listOfIssues.item(s);
    if (currPersonNode.getNodeType() == Node.ELEMENT_NODE) {

        Element currPersonElement = (Element) currPersonNode;

        // ...
        // firstname value...
        NodeList userNameList = currPersonElement.getElementsByTagName("username");
        Element userNameElement = (Element) userNameList.item(0);

        NodeList textUserNameList = userNameElement.getChildNodes();
        System.out.println("UserName : " + ((Node)
textUserNameList.item(0)).getNodeValue().trim());

        // ...
        // lastname value...
        NodeList pageIDList = currPersonElement.getElementsByTagName("pageid");
        Element pageIDNameElement = (Element) pageIDList.item(0);

        NodeList textPageIDList = pageIDNameElement.getChildNodes();
        System.out.println("PageID : " + ((Node) textPageIDList.item(0)).getNodeValue().trim());
        // ...
        // age value...
        NodeList priorityList = currPersonElement.getElementsByTagName("priority");
        Element ageElement = (Element) priorityList.item(0);

        NodeList priorityInfoList = ageElement.getChildNodes();
        System.out.println("Priority : " + ((Node) priorityInfoList.item(0)).getNodeValue().trim());
        // ...
        // height value...
        NodeList issuesFoundList = currPersonElement.getElementsByTagName("usernotes");
        Element heightElement = (Element) issuesFoundList.item(0);

        NodeList textHeightList = heightElement.getChildNodes();
        System.out.println("issues Found : " + ((Node)
textHeightList.item(0)).getNodeValue().trim());

        //ShowData= ((Node) textUserNameList.item(0)).getNodeValue().trim();

        //...
    }
}

```

```
//TESTING PURPOSES INTO DB2ADMIN.NOTES DOMINODB
```

```
//...
```

```
//Rather than using the above, which requires additional modification to code, we are going to test  
//the following code in JDeveloper to send data to IBM DB2ADMIN.NOTES_DOCS_RPT table.
```

```
//...
```

```
//TO DO: Split programs into two so that button runs for the specific process,  
//results in console or split into Strings for DB2
```

```
//Get XML file from URL, Now add same results found above to DB2
```

```
URL xmlURL = new URL("http", "localhost", 80,  
"docucontent.nsf/javaagentdataforwebnewshttp.xml");  
// Set up Connection to URL, grab file  
URLConnection conn3ctXML = xmlURL.openConnection();  
// connect to URL. grab file  
conn3ctXML.connect();
```

```
// Read URL data into new XML file
```

```
BufferedReader bufferRead = new BufferedReader(new  
InputStreamReader(conn3ctXML.getInputStream()));
```

```
//...
```

```
//establish a Connection
```

```
Connection connection = DBConnector.getConnection();
```

```
//TO DO: Attempt using Prepared Statement here...
```

```
Statement statement = connection.createStatement();
```

```
String line;
```

```
while ((line = bufferRead.readLine()) != null) {
```

```
String[] XMLValues = line.split(","); //your seperator
```

```
//Convert String to right type. Integer, double, date etc.
```

```
statement.executeUpdate("INSERT INTO DB2ADMIN.NOTES_DOCS_RPT  
VALUES(" + XMLValues[0] + ",'" +  
+ XMLValues[1] + "','" +  
+ XMLValues[2] + "','" +  
+ XMLValues[3] + "')");
```

```
//Use a PeparedStatement to insert, it's easier and safer
```

```
System.out.println("Insert into DB2: " + XMLValues);
```

```
}
```

```
// clean up
```

```
bufferRead.close();  
statement.close();
```

```
        connection.close();

    }

}

} catch (SAXParseException err) {

    System.out.println("Error" + ", errLine " + err.getLineNumber() + ", uri " + err.getSystemId());
    System.out.println(" " + err.getMessage());

} catch (SAXException e) {

    Exception x = e.getException();
    ((x == null) ? e : x).printStackTrace();

} catch (Throwable t) {
    t.printStackTrace();
}

return null;

}

public void setIt1(RichInputText it1) {
    this.it1 = it1;
}

public RichInputText getIt1() {
    return it1;
}

public void setIns1(RichInputNumberSpinbox ins1) {
    this.ins1 = ins1;
}

public RichInputNumberSpinbox getIns1() {
    return ins1;
}

public void setIns2(RichInputNumberSpinbox ins2) {
    this.ins2 = ins2;
}

public RichInputNumberSpinbox getIns2() {
    return ins2;
}
```

```
}

public void setIt2(RichInputText it2) {
    this.it2 = it2;
}

public RichInputText getIt2() {
    return it2;
}

public void setIt3(RichInputText it3) {
    this.it3 = it3;
}

public RichInputText getIt3() {
    return it3;
}

public void setIt4(RichInputText it4) {
    this.it4 = it4;
}

public RichInputText getIt4() {
    return it4;
}

public void setT1(RichTable t1) {
    this.t1 = t1;
}

public RichTable getT1() {
    return t1;
}

public void setOt2(RichOutputText ot2) {
    this.ot2 = ot2;
}

public RichOutputText getOt2() {
    return ot2;
}

public void setOt3(RichOutputText ot3) {
    this.ot3 = ot3;
}

public RichOutputText getOt3() {
    return ot3;
}
```

```

public void setOt4(RichOutputText ot4) {
    this.ot4 = ot4;
}

public RichOutputText getOt4() {
    return ot4;
}

public void setOt5(RichOutputText ot5) {
    this.ot5 = ot5;
}

public RichOutputText getOt5() {
    return ot5;
}

public void setRte1(RichTextEditor rte1) {
    this.rte1 = rte1;
}

public RichTextEditor getRte1() {
    return rte1;
}

}

```

Below is the WebLogic console results from the previous code tested in JDeveloper. The button also fired the subsequent code found in Domino Designer to again grab the XML, but this time the process loads data to DB2...

```

Target URL -- http://localhost:7101/MyWebNews-ViewController-context-root/faces/jdlotusfeed.jsf
<Nov 9, 2015 9:22:17 PM EST> <Warning> <Socket> <BEA-000449> <Closing the socket, as no data read from it on 127.0.0.1:<4852><4852:2676:1109/212226:ERROR:ssl_client_socket_openssl.cc(1077)] handshake failed; returned -1, SSL error code 1, net_error=0<Nov 9, 2015 9:22:17 PM EST> <Warning> <Socket> <BEA-000449> <Closing the socket, as no data read from it on 127.0.0.1:<4852><4852:2676:1109/212226:ERROR:ssl_client_socket_openssl.cc(1077)] handshake failed; returned -1, SSL error code 1, net_error=0<Nov 9, 2015 9:24:11 PM EST> <Warning> <Socket> <BEA-000449> <Closing the socket, as no data read from it on 127.0.0.1:<org.apache.myfaces.trinidadinternal.application.ViewHandlerImpl> <ViewHandlerImpl> <_checkTimestamp> <TIMESTAMP_CHECKIN><Root element sitesissues>
Total Issues : 2
UserName : AlienHouse
PageID : http://localhost/docucontent.nsf/xpissuesformnewuser.xsp
Priority : Low
issues Found : Testing
UserName : username107
PageID : http://localhost/docucontent.nsf/xpusersubmit.xsp
Priority : Low
issues Found : I need more Cow Bells
<Nov 9, 2015 9:35:26 PM EST> <Warning> <Socket> <BEA-000449> <Closing the socket, as no data read from it on 127.0.0.1:<
```

IBM Domino Designer records from the XML file were gathered and were read into strings for Oracle JDeveloper Java program, which then loaded all values from the XML file into our Oracle WebLogic console. We were hoping to also forward data to IBM DB2's DB2ADMIN.NOTES\_DOCS\_RPT database table, however, we are seeing a log error in the system. Not worried at all, it could be something really simple. Let's have a look next.

## **Conclusion:**

Our existing IBM Domino Designer code samples were combined to arrive at a process that helped submit to Oracle WebLogic server console using an Oracle JDeveloper ADF project. We merged yet other samples to complete a working version that reads XML from the same URL, not only pushed the data to WebLogic console but also performed a submit to DB2. Will be checking the back-end to investigate. Still, we are very optimistic about the project thus far.

**See Next Journal Entries document...**

Version: 2015.11.09.12.01.AM

### **Credits**

**eBook Content**

Dököll Solutions, Inc.

**eBook Cover Logo**

IBM, Oracle