WebServices in Iseries

WebServices in Iseries

There are 2 types of webservices.

1. SOAP(This one is out of use.)
2. REST
3. There are 2 supporting data formats in REST.
4. JSON
5. XML

JSON and XML both are widely used to transfer data but JSON is more popular.

Steps to process an XML File.

1. CHGJOB CCSID(37)

This is extremely important to make it work. Run this in your CL Program

Dcl-S myLoginBase64 Varchar(200) Inz('') CCSID(1208);

This is how base64 variable should be declared.

1. URL is the http link from where your data will come.

Sample webservice.

**URL** = 'https://devwildfly.abc.com:8543/+

premium/policies/made?year=2020&month=01';

1. Setup your user passwords

myLogin = %trim(Username) + ':' + %trim(Password);

**Following command is to change the password to BASE64**

EXEC SQL SET :myLoginBase64 = SYSTOOLS.BASE64ENCODE(:myLoginBase64);

1. Setup myHeader to fetch the data in JSON/XML. Credentials are supplied here.

Header name Authorization is used to pass credentials to access webservice.

Header name Accept and Content is to ensure that webservice send back the XML file.

myHeader = '<httpHeader><header name="Authorization" value="Basic ' +

%trim(myLoginBase64) +'"/>+

<header name="Accept" value="application/xml"/>+

<header name="Content-Type" value="application/xml"/>+

</httpHeader>';

1. Following piece of code get the data from webservice in XML format

exec sql

drop table MYLIBL.XMLTable;

exec sql

Create table MYLIBL.XMLTable (MyXML XML);

monitor;

//replace :myheader with **null** if you don’t want to pass user password and if //you want to receive data back in DEFAULT format.

//From here on we are working only for XML. For JSON I would declare

//variable of **CLOB** type. But for XML we need data type XML.

//Following command HTTPGETCLOB will get the XML data into the table.

exec sql

insert into MYLIBL.XMLTable

SELECT \* FROM (Values(

SYSTOOLS.HTTPGETCLOB(:url,:myHeader)));

on-error;

return;

http\_crash();

endmon;

//The data in XML field does not look like XML. If you run select \* from //MYLIBL.XMLTABLE it won’t show you XML. It will show you pointer.

1. Parse the data.

This is the quickest way I found to parse the data.

begsr parse\_load;

exec sql

drop TABLE MYLIBL.XML\_tbl ;

//Create the structure of the data.

exec sql

CREATE TABLE MYLIBL.XML\_tbl (

STATECODE VARCHAR(02),

ASLOBCODE VARCHAR(05),

SYSTEMID VARCHAR(10),

UNERNPRM dec(11,2));

//Parse and load the data in MYLIBL.XML\_tbl

exec sql

insert into MYLIBL.XML\_tbl

SELECT X.\*

FROM MYLIBL.XMLTable A,

XMLTABLE ('$d/List/item' PASSING A.myXML

as "d" COLUMNS

STATECODE VARCHAR(02) PATH 'stateCode',

ASLOBCOD VARCHAR(05) PATH 'asLobCode',

SYSTEMID VARCHAR(10) PATH 'systemId',

UNERNPRM dec(11,2) PATH 'directUnearnedPremium') as X;

1. Results.

DATA WILL BE PARSED INTO THE MYLIBL.XML\_tbl TABLE

Some useful information

**Variable size and definitions**

H DFTACTGRP(\*NO) BNDDIR('HTTPAPI')

\* include libhttp2 in library list before compilation

/INCLUDE qrpglesrc,httpapi\_h

\*

F\*ESTFILE O E DISK

D url S 1000A varying

D ifs S 1006A varying

D Username s 80A

D Password s 1024A

d rc s 20I 0

D myHeader S 1000A varying

D myLogin S 200A varying

D myLoginBase64 S 200A varying

**Way to fetch the webservice JSON/XML file into IFS**

SetAuth is required for secured webservice which needs user & password.

rc=http\_setauth(HTTP\_AUTH\_BASIC: %Trim(Username): %Trim(Password));

ifs = '/home/SYNPRGPXB/resttest.xml';

RC = http\_get(%TRim(url): %Trim(ifs)) ;

if (rc<>1 and rc<>500);

http\_crash();

endif;

if rc=500;

errmsg#='path=error doc=file';

dsply errMsg#;

else;

Exsr parse\_load;

endif;

**Parsing of JSON DATA**

For JSON parsing there are following options

* **JSON\_TABLE** – I tried and spent lot of time but it did not work. Although it looked very promising.
* **JSON\_INFO** – Same as above.
* **JSON2BSON** – Hit the roadblock as following command needs to be run in QSHELL to make it work. This will update the SYSTOOLS library with new schemas and only DBAs can do that. I have high hopes that this will work.

/QIBM/ProdData/OS/SQLLIB/bin/db2nosql -setup enable