```
/*-----------------------------------------------------------------------------*
 * File Name:                                                                  *
 * Creation:                                                                   *
 * Purpose: OriginC Source C file                                              *
 * Copyright (c) Originlab Corp.    2002                                       *
 * All Rights Reserved                                                         *
 *                                                                             *
 * Modification Log:                                                           *
 *-----------------------------------------------------------------------------*/

////////////////////////////////////////////////////////////////////////////////
// you can include just this typical header file for most Origin built-in functions and classes
// and it takes a reasonable amount of time to compile,
#include <origin.h>
// this file include most of the other header files except the NAG header, which takes longer t
// NAG routines
//#include <OC_nag.h> // this contains all the NAG headers,

////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////
//This sample demonstrates how to use COM object through OriginC's COM support and
//how to access worksheet properties from OriginC.
//
//  In this sample, the MSXML parser being used is version 3. You can switch to other
//version by changing the XML_PARSER_APP_ID definition
//
//  Function WKSToXML will export an Origin worksheet into an XML document based on
//the schema definition (that schema file is located under OriginC\Samples\COM\
//folder, worksheet.xsd).
//
//  Function XMLToWKS will import a formatted XML document into worksheet
//
//
//  To export a Worksheet into an XML document:
//      1, make sure there is a worksheet exist (such as Data1) and all columns
//         (or columns need to be exported) are numeric format
//      2, load WksToXML.c into workspace and compile it
//      3, execute following command from labtalk window
//          WKSToXML Data1 c:\myData1.xml
//          This command will export data from worksheet Data1 into XML document,
//          c:\myData1.xml
//
//  To import a formatted XML document into Worksheet
//      1, load WksToXML.c into workspace and compile it if that has not been done
//      2, execute following command from labtalk window
//          XMLToWKS c:\myData1.xml
//          That will create a new worksheet and import data from that XML document
//          into this worksheet
//
//Please Note:
//  1, Currently, only numeric columns can be exported, all other types will be skipped.
//     Support for other column types will be added in the future.




////////////////////////////////////////////////////////////////////////////////
//use MSXML version 3 feature
#define XML_PARSER_APP_ID           "msxml2.domdocument.3.0"

//use MSXML version 2 feature
//#define XML_PARSER_APP_ID             "msxml2.domdocument"

//use MSXML version 1 feature
//#define XML_PARSER_APP_ID             "msxml.domdocument"

#define XML_INSTRUCTION_TAG         "xml"
#define XML_INSTRACTION_CONTENTS    "version=\"1.0\""
#define WORKSHEET_TAG               "Worksheet"
#define COLUMN_TAG                  "Column"
#define DATASET_TAG                 "Dataset"


#define NAME_ATTRIBUTE              "Name"
#define LABEL_ATTRIBUTE             "Label"
```

```
#define NUMOFCOLS_ATTRIBUTE         "NumCols"
#define FORMAT_ATTRIBUTE            "Format"
#define SUBFORMAT_ATTRIBUTE         "SubFormat"
#define TYPE_ATTRIBUTE              "Type"
#define NUMROWS_ATTRIBUTE           "NumRows"
#define INTERNAL_DATATYPE_ATTRIBUTE "InternalDataType"

#define BIN_BASE64_DATATYPE         "bin.base64"
#define SCHEMA_DEFINITION           "worksheet.xsd"
#define RELATIVE_PATH_TO_SCHEMA     "OriginC\\Samples\\COM\\"



////////////////////////////////////////////////////////////////////////////////
// start your functions here
static Object g_XMLParser;

static BOOL getXMLParser(Object& parser)
{
    //this will remove previous object
    parser = CreateObject(XML_PARSER_APP_ID);

    if( !parser )
    {
        printf("Fail to create the XML parser object\n");

        return FALSE;
    }

    return TRUE;
}




//-----------------------------------------------------------------------------------------
//XMLToWKS(string strXMLFile = "c:\\Data1.xml")
//  The function to import an XML document into worksheet
//Arguments:
//  strXMLFile: the full path to the XML file
//-----------------------------------------------------------------------------------------
void XMLToWKS(string strXMLFile = "c:\\Data1.xml")
{
    if( !getXMLParser(g_XMLParser) )
        return;

    //disable asynchronous download for embedded documents
    g_XMLParser.async = FALSE;
    if( !loadXMLFile(strXMLFile) )
        return;

    Object rootElement;
    if( !getRootElement(rootElement) )
        return;
    if( !rootElement.hasChildNodes() )
    {
        printf("No child nodes to read\n");
        return;
    }

    Worksheet wks;
    wks.Create();
    if(!wks)
    {
        printf("Fail to create a new worksheet\n");
        return;
    }

    //delete all columns from worksheet
    while( wks.DeleteCol(0) )
        ;

    //the node list will get all <Column></Column> records
    Object colNodeList;
    colNodeList = rootElement.childNodes;
```

```
    if( !colNodeList )
    {
        printf("Fail to get node list\n");
        return;
    }


    //loop through all nodes and construct the worksheet
    Object colNode;
    colNode = colNodeList.nextNode();

    while( colNode )
    {
        string strName;
        strName = colNode.nodeName;
        if( strName.Compare(COLUMN_TAG) != 0 )
        {
            printf("Error to read information: it is not a Column node\n");
            continue;
        }

        //get the first child from column node
        // that node is the Dataset node with all data included
        Object datasetNode;
        datasetNode = colNode.firstChild;
        if( !datasetNode )
            continue;   //this <Column> node has no Dataset child

        strName = datasetNode.nodeName;
        if( strName.Compare(DATASET_TAG) != 0 )
        {
            printf("Error to read information: it is not a Dataset node\n");
            continue;
        }

        //add a new column, then use information from XML
        //document to initialize column
        int nIndex = wks.AddCol();
        Column colObj = wks.Columns(nIndex);
        setColumnValue(colNode, datasetNode, colObj);

        colNode = colNodeList.nextNode();
    }

    out_str("Done!");
}


//-------------------------------------------------------------------------------------------
//loadXMLFile(string strFile)
//  Parse the XML document in DOM mode
//Arguments:
//  strFile: full path to the XML file location
//Return:
//  TRUE/FALSE
//-------------------------------------------------------------------------------------------
static BOOL loadXMLFile(string strFile)
{
    g_XMLParser.async = FALSE;
    int nRet = g_XMLParser.load(strFile);

    if( nRet != 0 )
    {
        Object Error = g_XMLParser.parseError;

        int nErrCode;
        nErrCode = Error.errorCode;

        if( nErrCode == 0 )
            return TRUE;

        string strReason;
        strReason = Error.reason;
        printf("Error code is %d\nThe reason is : %s\n", nErrCode, strReason);

        return FALSE;
```

```
    }

    return TRUE;
}


//-----------------------------------------------------------------------------------------
//getRootElement(Object& rootElement)
//  Get the root element from the parsed document. The root node is the <Worksheet> node
//Arguments:
//  rootElement: return the object represent the root element in that XML document
//Return:
//  TRUE/FALSE
//-----------------------------------------------------------------------------------------
static BOOL getRootElement(Object& rootElement)
{
    rootElement = g_XMLParser.documentElement;

    if( !rootElement )
    {
        printf("Fail to get root element\n");
        return FALSE;
    }

    return TRUE;
}


//-----------------------------------------------------------------------------------------
//getAttributeByName(Object& node, string strAttribute, string& strValue)
//  Get attribute's value from its name
//Arguments:
//  node:           the node to retrieve attribute
//  strAttribute:   the attribute name
//  strValue:       value of the attribute, in string format
//Return:
//  TRUE/FALSE
//-----------------------------------------------------------------------------------------
static BOOL getAttributeByName(Object& node, string strAttribute, string& strValue)
{

    Object attributeMap = node.attributes;
    Object attr;

    if( attributeMap )
    {
        attr = attributeMap.nextNode();
        while( attr )
        {
            string strName;
            strName = attr.nodeName;

            if( strName.Compare(strAttribute) == 0 )
            {
                strValue = attr.nodeValue;
                return TRUE;
            }

            attr = attributeMap.nextNode();
        }

    }

    return FALSE;
}


//-----------------------------------------------------------------------------------------
//setColumnValue(Object& colNode, Object& datasetNode, Column& colObj)
//  populate an Origin column from XML document's Column node object
//Arguments:
//  colNode:        the Column node
//  datasetNode:    the Dataset node, child node of the Column node
//  colObj:         Origin column object in worksheet
//-----------------------------------------------------------------------------------------
static void setColumnValue(Object& colNode, Object& datasetNode, Column& colObj)
```

```
{
    string strName;
    if( getAttributeByName(colNode, NAME_ATTRIBUTE, strName) )
        colObj.SetName(strName);

    string strLabel;
    if( getAttributeByName(colNode, LABEL_ATTRIBUTE, strLabel) )
        colObj.SetLabel(strLabel);

    string strColFormat;
    if( getAttributeByName(colNode, FORMAT_ATTRIBUTE, strColFormat) )
        colObj.SetFormat(atoi(strColFormat));
    else
        colObj.SetFormat(OKCOLTYPE_NUMERIC); //numeric as the default

    string strColType;
    if( getAttributeByName(colNode, TYPE_ATTRIBUTE, strColType) )
        colObj.SetType(atoi(strColType));
    else
        colObj.SetType(OKDATAOBJ_DESIGNATION_Y); //Y as the default

    string strInternalType;
    if( getAttributeByName(colNode, INTERNAL_DATATYPE_ATTRIBUTE, strInternalType) )
    {
        colObj.SetInternalData(atoi(strInternalType));
    }
    else
    {
        colObj.SetInternalData(FSI_DOUBLE);
    }


    //get byte stream from XML document
    vector<byte> vv;
    vv = datasetNode.nodeTypedValue;

    //restore dataset by calling SetBytes()
    Dataset dataset(colObj);
    dataset.SetBytes(vv);

}




//-------------------------------------------------------------------------------------------
//WKSToXML(string strWks = "Data1", string strXMLFile ="c:\\Data1.xml")
//  The function to export a worksheet into XML document
//Arguments:
//  strXMLFile: the full path to the XML file
//  strWks:     the worksheet name, the worksheet should exist to run this sample
//-------------------------------------------------------------------------------------------
void WKSToXML(string strWks = "Data1", string strXMLFile ="c:\\Data1.xml")
{
    if( !getXMLParser(g_XMLParser) )
        return;

    Object pInstruction;
    pInstruction = g_XMLParser.createProcessingInstruction(XML_INSTRUCTION_TAG, XML INSTRACTION

    if( !pInstruction)
    {
        printf("Fail to create process instruction node\n");
        return;
    }

    g_XMLParser.appendChild(pInstruction);


    Worksheet wks(strWks);

    if(!wks)
    {
        printf("Worksheet %s not found\n", strWks);
```

```
        return;
    }


    Object root = g_XMLParser.createElement(WORKSHEET_TAG);
    if( !root )
    {
        printf("Fail to create root element\n");
        return;
    }

    //set attributs of root element, worksheet properties
    int nNumOfColumns  = addWorksheetInformation(root, wks);
    g_XMLParser.appendChild(root);

    for( int ii=0; ii<nNumOfColumns; ii++ )
    {
        //first, create the node for column
        Object colNode = g_XMLParser.createElement(COLUMN_TAG);
        if( !colNode )
        {
            printf("Fail to create Column element\n");
            continue;
        }

        //Get the column
        Column colObj = wks.Columns(ii);

        //retrieve information from this column and save into XML node
        //if the column can not be converted, skip to next column
        if( !addColumnInformation(colNode, colObj) )
            continue;

        //append column node to the document
        root.appendChild(colNode);
    }


    //save document to file
    g_XMLParser.save(strXMLFile);
    printf("XML document is created as %s\n", strXMLFile);
}



//-----------------------------------------------------------------------------------------
//addWorksheetInformation(Object& wksNode, Worksheet& wksObj)
//  Add worksheet name and number of columns to root node's attribute list while setting
//other attributes of <Worksheet> node
//Arguments:
//  wksNode:    the Worksheet node in XML document
//  wksObj:     the worksheet object from Origin
//Return:
//  an integer number to indicate number of columns in the worksheet
//-----------------------------------------------------------------------------------------
static int addWorksheetInformation(Object& wksNode, Worksheet& wksObj)
{
    string strName = wksObj.GetPage().GetName();
    int nNumCols = wksObj.GetNumCols();

    string strAppPath = GetAppPath();
    string strSchemaLocation;
    strSchemaLocation.Format("%s%s%s", strAppPath, RELATIVE_PATH_TO_SCHEMA, SCHEMA_DEFINITION);

    //set attributs of root element, worksheet properties
    wksNode.setAttribute("xmlns:xsi", "http://www.w3.org/2001/XMLSchema");
    wksNode.setAttribute("xsi:schemaLocation", strSchemaLocation);
    wksNode.setAttribute("xmlns:dt", "urn:schemas-microsoft-com:datatypes");
    wksNode.setAttribute(NAME_ATTRIBUTE, strName);
    wksNode.setAttribute(NUMOFCOLS_ATTRIBUTE, nNumCols);

    return nNumCols;
}


//-----------------------------------------------------------------------------------------
```

```
//addColumnInformation(Object& colNode, Column& colObj)
//  Export an Origin column into an XML Column node including column data and information
//Arguments:
//  colNode:    the Column node in XML document
//  colObj:     the column object from Origin
//Return:
//  TRUE/FALSE
//-----------------------------------------------------------------------------------------
static BOOL addColumnInformation(Object& colNode, Column& colObj)
{
    if( !colObj )
    {
        printf("Invalid column object\n");
        return FALSE;
    }

    int nDataType = colObj.GetInternalData();
    //get all information to be saved in the column node
    string strColName;
    colObj.GetName(strColName);

    //the converting from worksheet to XML document can not handle
    //the following type of columns right now
    //  FSI_CHAR,
    //  FSI_TEXT,
    //  FSI_MIXED
    if( nDataType == FSI_CHAR || nDataType == FSI_TEXT || nDataType == FSI_MIXED )
    {
        printf("Fail to export coulmn \"%s\"!!! Please make sure it is Numeric type\n", strColN
        return FALSE;
    }

    //cloumn Name
    colNode.setAttribute(NAME_ATTRIBUTE, strColName);

    //column Label
    string strLabel;
    colObj.GetLabel(strLabel);
    colNode.setAttribute(LABEL_ATTRIBUTE, strLabel);

    //column Format
    int nFormat = colObj.GetFormat();
    colNode.setAttribute(FORMAT_ATTRIBUTE, nFormat);

    //column Type
    int nType = colObj.GetType();
    colNode.setAttribute(TYPE_ATTRIBUTE, nType);

    //construct a dataset object from column object
    Dataset datSet(colObj);

    //column NumRows
    int nSize =datSet.GetSize();
    colNode.setAttribute(NUMROWS_ATTRIBUTE, nSize);

    colNode.setAttribute(INTERNAL_DATATYPE_ATTRIBUTE, nDataType);

    Object datasetNode;
    datasetNode = g_XMLParser.createElement(DATASET_TAG);
    if( !datasetNode )
    {
        printf("Fail to create Dataset element\n");
        return FALSE;
    }
    else
    {
        //set the dataType to be bin.base64
        string strType = BIN_BASE64_DATATYPE;
        datasetNode.dataType = strType;

        //for bin.base64 data type, XML requires data to be in
        //a variant with byte array. the variant needs to
        //VT_UI1 | VT_ARRAY type
        //To get proper data, first, get a byte vector from the dataset,
        //then, get the variant data from the byte vector
        vector<byte> vv;
```

```
        datSet.GetBytes(vv);
        _VARIANT varData = vv.GetDataAsOneDimensionalArray();
        datasetNode.nodeTypedValue = varData;

        //append datasetNode to column node
        colNode.appendChild(datasetNode);
    }

    return TRUE;
}
```