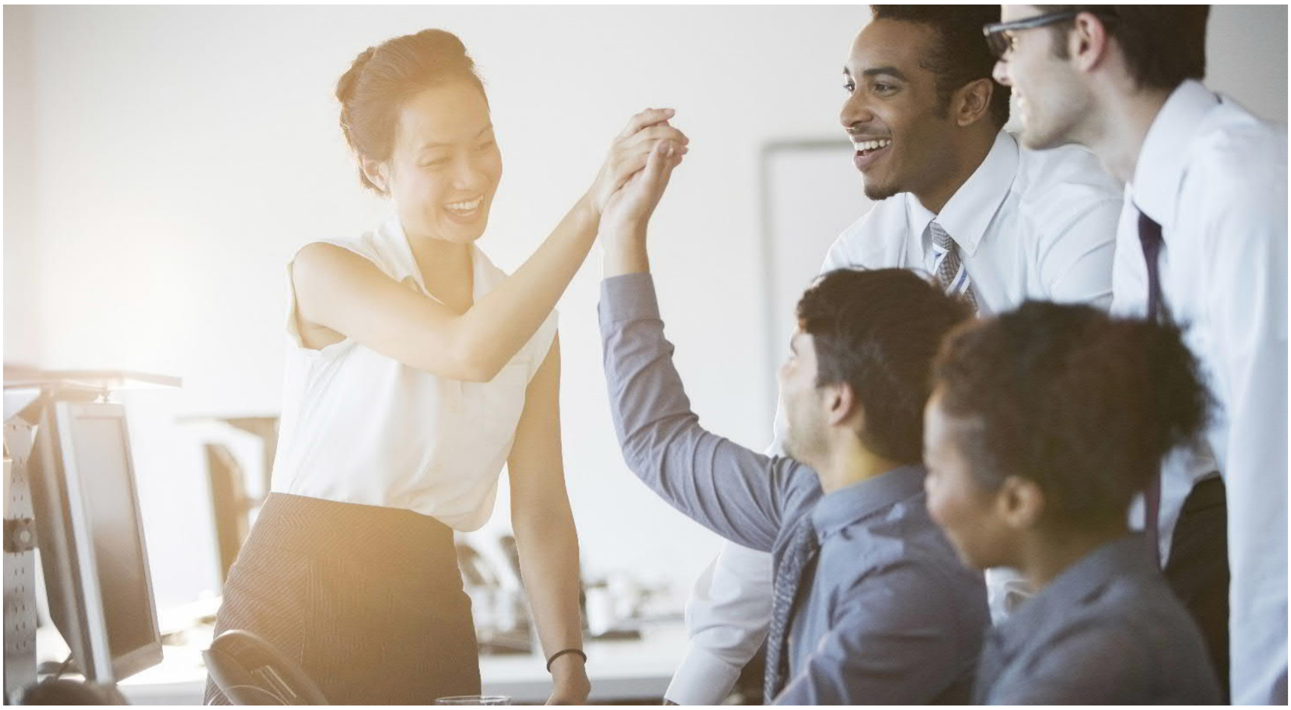


Exercise Solution: XML

Data Interface API

PUBLIC

INTRODUCTION

In this exercise, you will perform the following tasks:

1. Create new buttons in your project called "Working with XML"
2. Save the Invoice created in the Documents exercise as XML
3. Modify the XML data obtained before and add it to the SAP Business One database

PREREQUISITE:

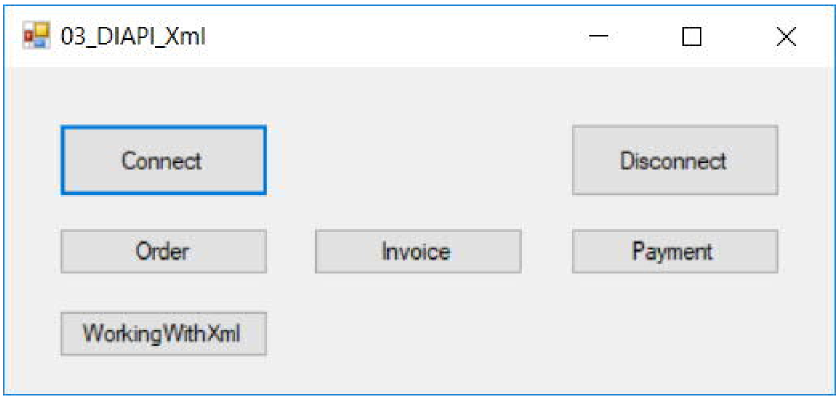
- This document is using the **C Sharp** (C#) language
- This document is using the Microsoft Visual Studio 2015
- Continue to work with the project finalized in previous exercise.
- Use the demo database for SAP Business One, version for SAP HANA or SAP Business One
- Credentials: User code: **manager**

GUIDELINES:

The screenshots provided here are for your reference only and may differ from the actual screenshots in your system.

1. TASK - CREATE NEW BUTTONS IN YOUR PROJECT CALLED "WORKING WITH XML"

1.1. On your Visual Studio project create new buttons called “Working with XML”



2. TASK - CREATE S SALES ORDER

2.1. Try all settings for *XmlExportType* property on the *Company* object and find the differences.

The 4 property types are

XML ExportType	Definition
oCompany.XmlExportType = SAPbobsCOM.BoXmlExportTypes.xet_AllNodes	Export to XML all fields (both read only and read/write fields) from the database.
oCompany.XmlExportType = SAPbobsCOM.BoXmlExportTypes.xet_ExportImportMode	Export to XML only valid fields that support XML import (read/write fields only) from the database.
oCompany.XmlExportType = SAPbobsCOM.BoXmlExportTypes.xet_NodesAsProperties	Export to XML all fields as properties from the database.
SAPbobsCOM.BoXmlExportTypes.xet_ValidNodesOnly	Export to XML only valid fields that support XML import and export (read/write fields only that do not contain null values) from the database.

2.2. Save the Invoice document created in the previous exercise in Xml format.

```
oCompany.XmlExportType = SAPbobsCOM.BoXmlExportTypes.xet_ExportImportMode;
SAPbobsCOM.Documents oInv;
oInv = oCompany.GetBusinessObject(SAPbobsCOM.BoObjectTypes.oInvoices);
if (oInv.GetByKey(Int32.Parse(MySalesInvoice))== true)
{
      oInv.SaveXML("c:\\temp\\invoice_" + MySalesInvoice + ".xml");
      MessageBox.Show("Invoice " + MySalesInvoice + " exported to XML");
}
Else
      MessageBox.Show("Get Invoice failed: " + MySalesInvoice + " - "
+oCompany.GetLastErrorDescription());
```

2.3. Test also the method *GetBusinessObjectXmlSchema* of the Company object. What kind of information does it save?

```
string schema;
schema = oCompany.GetBusinessObjectXmlSchema(SAPbobsCOM.BoObjectTypes.oInvoices);
MessageBox.Show(schema);
```

This method retrieves the XML schema used to define the structure and content of the object.

3. TASK - 3. MODIFY THE XML DATA OBTAINED BEFORE AND ADD IT TO THE SAP BUSINESS ONE DATABASE

3.1. Try all files generated above and check the errors (exceptions) for details

```
       oInv = oCompany.GetBusinessObjectFromXML("c:\\temp\\invoice_" + MySalesInvoice +
".xml", 0);
       oInv.Comments = "invoice loaded from xml";
       int ret = oInv.Add();
       if (ret == 0)
       {
          oCompany.GetNewObjectCode(out MySalesInvoice);
          MessageBox.Show("Invoice " + MySalesInvoice + " added from xml.");
        }
        Else
          MessageBox.Show("Adding invoice " + MySalesInvoice + " from XML failed: " +
   oCompany.GetLastErrorDescription());
```

Similar exercises can be found in the SDK samples (in the SDK Folder – see Appendix "SDK Installations" for more information), COM DI/7.SaveXML and COM DI/8.LoadFromXML www.sap.com