



DB To Xml

Ultimate B2B tool

Version 1.4

1999-2002 SoftRus LLC.
All Rights Reserved.

Table Of Contents

Problem Statement	3
Solution	3
Advantages	3
System Requirements	4
Installation	4
Configuration	5
Template modification	6
Getting Started	6
Code Examples	8
W3C Xml schema example:	8
VB COM object example:	8
Java Code Example	18

Problem Statement

I believe there is no single person in the IT world who was not excited in some point of using Xml in the applications. Most of our system integration projects were heavily related on use of Xml. And eventually we find out that same or similar code is reproduced in different projects. Almost every project involve database access, whether you store your information in Oracle or your client stoked with flat files and excel spreadsheets, you always use same technology to extract and update data. Due to the Xml revolution everybody require from you to perform data exchange in Xml format. How many times you wrote code that generates these insert and update statements or functions that serialize your Recordsets to the xml back and forth. And finally you have little time left to actually implement business logic of your application. So we decide to implement solution that will solve all this problems in fast and simple way.

Solution

Our idea of DbToXml application is to create fast and reliable code-generator, that allow you to generate Xml schemas, test xml files and actual VB classes and java beans, that allow you to not worry about all database access code and Xml parsing procedures. DbToXml does just that. You show it your existing ODBC connection, and it will do the rest for you. Sounds to good to be true... Try it for your self. Our developers enjoy using it in the endless data integration projects. It cut time and cost of the project from two to three times. And most of all you are not wasting your time on database calls debugging you are implementing the business classes, which are actually different depending of the client requirements.

Advantages

Advantages of using DbToXml are obvious. You are getting up to speed with your project in hours. The client doesn't need to know how much time you actually spend on the development couple of weeks or couple hours. Here what you get without any efforts:

- o Biztalk schemas for your Integration needs.
- o Ready to use VB COM components.
- o Ready to be compiled and packaged Java beans.
- o Easily adjustable templates and configuration to satisfy your coding standards.
- o XML data for testing.
- o SQL scripts for data manipulations.

So decide for your self.

System Requirements

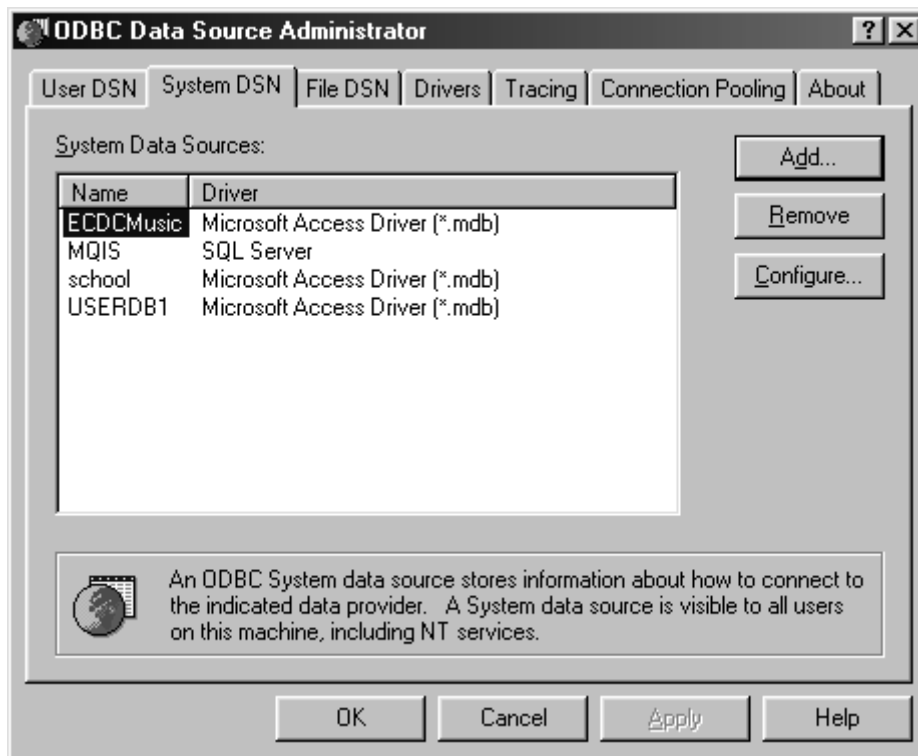
DbToXml requires a minimum of 3.9 MB of hard disk space, processor and memory requirements as shown below.

Operating System	Minimum RAM	Processor
Windows 95/98	32 MB	486 or compatible
Windows	Me 32 MB	Pentium/150 MHz or compatible
Windows NT 4.0	32 MB	486/33 or compatible
Workstation with SP4 applied		
Windows 2000 Professional	64 MB	Pentium/133 MHz or compatible

Installation

You should have MSXML 3.0 installed on your machine PRIOR to installing Db To Xml. You can download MSXML 3.0 from Microsoft's XML page (<http://msdn.microsoft.com/code/default.asp?url=/code/sample.asp?url=/msdn-files/027/001/591/msdncompositedoc.xml>). If you're not sure whether MSXML 3.0 is installed correctly on your machine, the easiest way to tell is to open VB6, click **Project** \square **References** from the menu, and search for 'Microsoft XML, v3.0'. If it's there, please proceed to the next step below. Otherwise, you must download MSXML 3.0 prior to proceeding. Once you download zipped archive, unzip it to a temporary folder and run setup.exe. Follow simple installation instructions to complete the process. If you don't have any ODBC connections configured, you will need to create at least one for your testing.

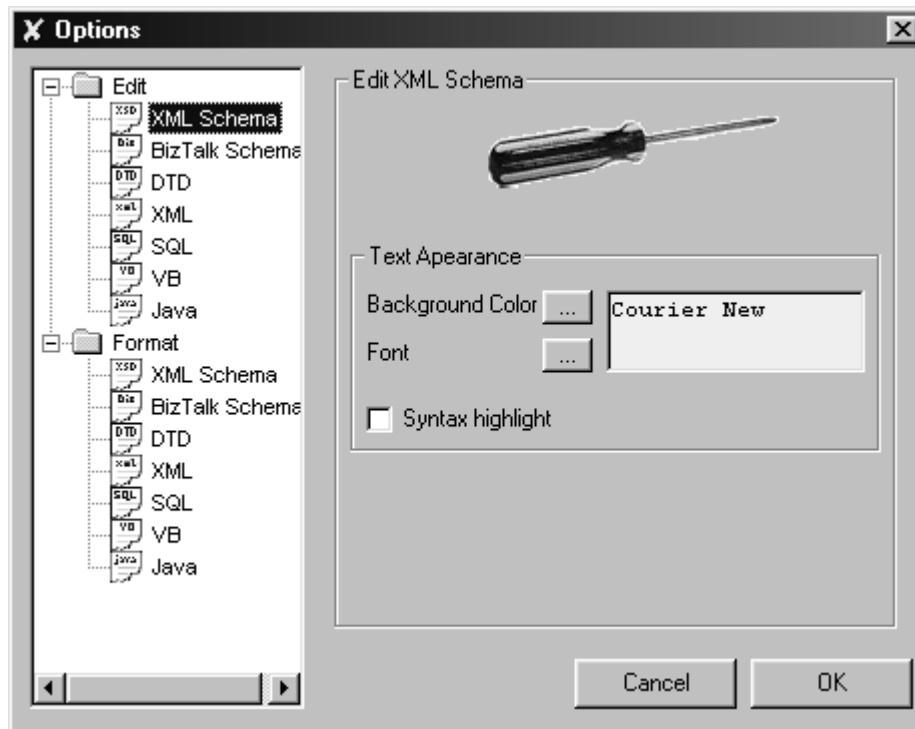
Go to the Administrative tools and start ODBC Connection manager, or to the control panel on Windows 95 and 98.



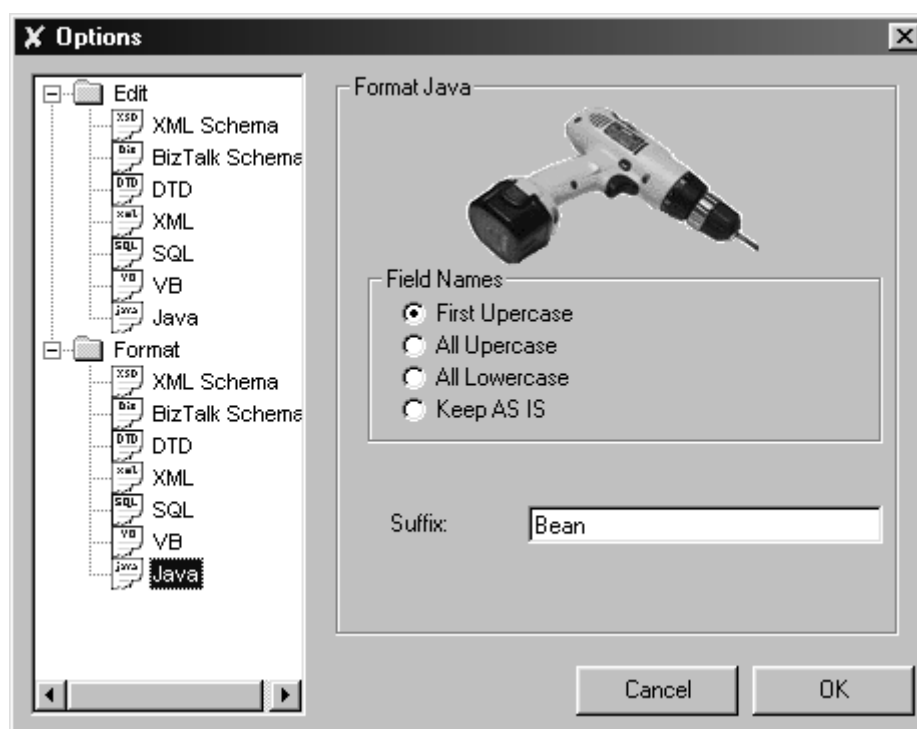
Click add button and select valid driver for your database, Name your connection, configure corresponding user logins and passwords if required and you are ready to use DbToXml

Configuration

Several configuration options are available to customize generated code. To adjust this options, select **Tools** menu item and then **Options**.



For every type of generated code you can specify **Edit** options and **Format** options. **Edit** options include selection of font, background color and optional syntax highlighting. **Format** options allow you to specify the way your database field names are used in application. You can use them the way they defined in database or you probably would rather have all VB variables in lowercase and have your XML schema tags to be in uppercase (some mainframe applications require this). In addition it allow you to add your own style to code and schema naming conventions by adding application specific suffixes.



If this is not clear, try to generate some code from your database, and it became pretty obvious. You can highlight code syntax by selecting **Highlight** menu-item or clicking on corresponding toolbar button.

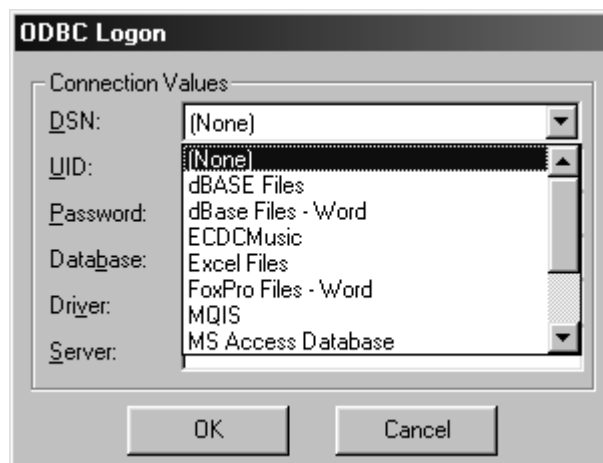
Template modification

The DbToXml comes with default templates for VB and Java code. Templates are stored in Templates subfolder of application installation folder. Templates define main methods of your classes, and allow you to modify these methods, add application specific comments or add some other useful methods to your classes. Template processing engine of the DbToXml defined several variables that will be replaced in with information from database. Here is the list of most common template variables:

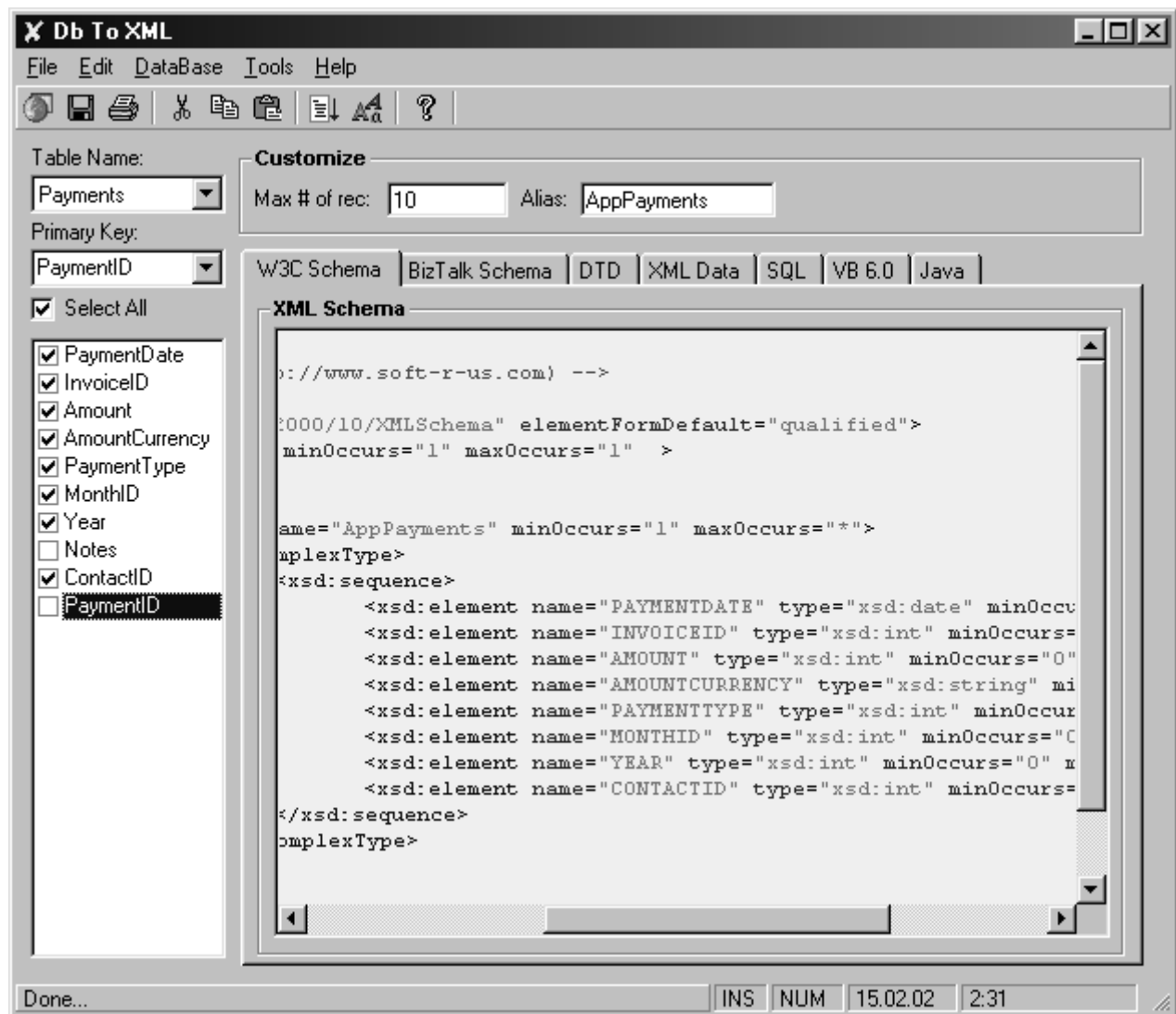
Variable	Description	Example of substitution
%TABLE_NAME%	Current Table or alias name	TUser
%PK_NAME%	Selected Primary Key name	UserId
%CONNECT_STRING%	Database Connection string	PROVIDER=MSDASQL;DSN=tst;UID=;PWD=;
%GET_XML%	Xml extraction lines	appendNode root, "Author", vAuthor
%SET_XML%	Xml parsing lines	vAuthor = getNodeData(node, "Author")
%FIELD_NAMES%	Field names for the insert statement	(Au_ID, Author, Year_Born)
%FIELD_VALUES%	Field values for the insert statement	(" + SqlString(vAu_ID, adDecimal) + ", " + SqlString(vAuthor, adVarChar) + ") "
%FIELD_ASSIGN%	Field assign for the update and select statements	Au_ID = " + SqlString(vAu_ID, adDecimal) + "
%FIELD_NAME%	Field name for the Property	UserId

Getting Started

You install DbToXml and configured the ODBC connections. What is next? The rest is very simple. Start DbToXml , select **Database** then **Connect To Database** menu item. Select your DSN from drop-down list, provide login and password information if required. And you will see all available tables displayed in the Tables drop-down listbox.



Select Table from the listbox and then select primary key for the table. Only single field primary keys are supported in this version.



Generated code can be easily updated if your table has complex primary key. After you made your selection, you can update Alias field to specify the alias name to be used for this table in your code. To accelerate code generation process you can limit number of records to be processed by populating 'Max # of Rec' field. Select **Tools** ▢ **Generate** menu item or corresponding button on the tool bar to generate code for you. You can save, print, or just copy and paste code to your application.

Code Examples

Here is a few examples of the code generated by using DbToXml:

W3C Xml schema example:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- XSD Generated by DbToXML v1.5 (http://www.soft-r-us.com). Please send your comments to:
support@soft-r-us.com ->

<xsd:schema xmlns:xsd="http://www.w3.org/2000/10/XMLSchema" elementFormDefault="qualified">
  <xsd:element name="CUSTOMERSDOC" minOccurs="1" maxOccurs="1" >
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="CUSTOMERS" minOccurs="1" maxOccurs="*">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="CUSTOMERID" type="xsd:string" minOccurs="1" maxOccurs="1"/>
              <xsd:element name="COMPANYNAME" type="xsd:string" minOccurs="1" maxOccurs="1"/>
              <xsd:element name="CONTACTNAME" type="xsd:string" minOccurs="0" maxOccurs="1"/>
              <xsd:element name="CONTACTTITLE" type="xsd:string" minOccurs="0" maxOccurs="1"/>
              <xsd:element name="ADDRESS" type="xsd:string" minOccurs="0" maxOccurs="1"/>
              <xsd:element name="CITY" type="xsd:string" minOccurs="0" maxOccurs="1"/>
              <xsd:element name="REGION" type="xsd:string" minOccurs="0" maxOccurs="1"/>
              <xsd:element name="POSTALCODE" type="xsd:string" minOccurs="0" maxOccurs="1"/>
              <xsd:element name="COUNTRY" type="xsd:string" minOccurs="0" maxOccurs="1"/>
              <xsd:element name="PHONE" type="xsd:string" minOccurs="0" maxOccurs="1"/>
              <xsd:element name="FAX" type="xsd:string" minOccurs="0" maxOccurs="1"/>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

VB COM object example:

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1 'True
  Persistable = 0 'NotPersistable
  DataBindingBehavior = 0 'vbNone
  DataSourceBehavior = 0 'vbNone
  MTSTransactionMode = 1 'NoTransaction
End
Attribute VB_Name = "CUSTOMERSClass"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False

Implements ObjectControl

'This class module references following COM objects:
' * Microsoft XML v3.0
' * Microsoft ActiveX Data Objects 2.0 Library
' * COM+ Services Type Library for W2K.
'Please update your VB project file when you add this Class Module

Private objConn As ADODB.Connection ' Shared DbConnection object
Private objRS As ADODB.Recordset
Private objCtx AsObjectContext

Private m_Customerid As String
Private m_Companyname As String
Private m_Contactname As String
Private m_Contacttitle As String
```

```

Private m_Address As String
Private m_City As String
Private m_Region As String
Private m_Postalcode As String
Private m_Country As String
Private m_Phone As String
Private m_Fax As String

'*****
'* Property: Customerid
'* Based on the field Customerid of dbo.Customers table
'* Let and Get methods
'* Purpose: access Customerid property
'*****
Public Property Let Customerid(ByVal val As String)
On Error GoTo Customerid_Err
    m_Customerid = val
    Exit Property
Customerid_Err:
    Err.Raise Err.Number, Err.Source & " (CUSTOMERSClass.Customerid.Let)",
Err.Description
End Property

Public Property Get Customerid() As String
On Error GoTo Customerid_Err
    Customerid = m_Customerid
    Exit Property
Customerid_Err:
    Err.Raise Err.Number, Err.Source & " (CUSTOMERSClass.Customerid.Get)",
Err.Description
End Property

'*****
'* Property: Companyname
'* Based on the field Companyname of dbo.Customers table
'* Let and Get methods
'* Purpose: access Companyname property
'*****
Public Property Let Companyname(ByVal val As String)
On Error GoTo Companyname_Err
    m_Companyname = val
    Exit Property
Companyname_Err:
    Err.Raise Err.Number, Err.Source & " (CUSTOMERSClass.Companyname.Let)",
Err.Description
End Property

Public Property Get Companyname() As String
On Error GoTo Companyname_Err
    Companyname = m_Companyname
    Exit Property
Companyname_Err:
    Err.Raise Err.Number, Err.Source & " (CUSTOMERSClass.Companyname.Get)",
Err.Description
End Property

'*****
'* Property: Contactname
'* Based on the field Contactname of dbo.Customers table
'* Let and Get methods
'* Purpose: access Contactname property
'*****
Public Property Let Contactname(ByVal val As String)
On Error GoTo Contactname_Err
    m_Contactname = val
    Exit Property
Contactname_Err:
    Err.Raise Err.Number, Err.Source & " (CUSTOMERSClass.Contactname.Let)",
Err.Description
End Property

```

```

Public Property Get Contactname() As String
On Error GoTo Contactname_Err
    Contactname = m_Contactname
    Exit Property
Contactname_Err:
    Err.Raise Err.Number, Err.Source & " (CUSTOMERSClass.Contactname.Get)",
Err.Description
End Property

'*****
'* Property: Contacttitle
'* Based on the field Contacttitle of dbo.Customers table
'* Let and Get methods
'* Purpose: access Contacttitle property
'*****
Public Property Let Contacttitle(ByVal val As String)
On Error GoTo Contacttitle_Err
    m_Contacttitle = val
    Exit Property
Contacttitle_Err:
    Err.Raise Err.Number, Err.Source & " (CUSTOMERSClass.Contacttitle.Let)",
Err.Description
End Property

Public Property Get Contacttitle() As String
On Error GoTo Contacttitle_Err
    Contacttitle = m_Contacttitle
    Exit Property
Contacttitle_Err:
    Err.Raise Err.Number, Err.Source & " (CUSTOMERSClass.Contacttitle.Get)",
Err.Description
End Property

'*****
'* Property: Address
'* Based on the field Address of dbo.Customers table
'* Let and Get methods
'* Purpose: access Address property
'*****
Public Property Let Address(ByVal val As String)
On Error GoTo Address_Err
    m_Address = val
    Exit Property
Address_Err:
    Err.Raise Err.Number, Err.Source & " (CUSTOMERSClass.Address.Let)", Err.Description
End Property

Public Property Get Address() As String
On Error GoTo Address_Err
    Address = m_Address
    Exit Property
Address_Err:
    Err.Raise Err.Number, Err.Source & " (CUSTOMERSClass.Address.Get)", Err.Description
End Property

'*****
'* Property: City
'* Based on the field City of dbo.Customers table
'* Let and Get methods
'* Purpose: access City property
'*****
Public Property Let City(ByVal val As String)
On Error GoTo City_Err
    m_City = val
    Exit Property
City_Err:
    Err.Raise Err.Number, Err.Source & " (CUSTOMERSClass.City.Let)", Err.Description
End Property

Public Property Get City() As String
On Error GoTo City_Err

```

```

        City = m_City
        Exit Property
City_Err:
    Err.Raise Err.Number, Err.Source & " (CUSTOMERSClass.City.Get)", Err.Description
End Property

'*****
'* Property: Region
'* Based on the field Region of dbo.Customers table
'* Let and Get methods
'* Purpose: access Region property
'*****
Public Property Let Region(ByVal val As String)
On Error GoTo Region_Err
    m_Region = val
    Exit Property
Region_Err:
    Err.Raise Err.Number, Err.Source & " (CUSTOMERSClass.Region.Let)", Err.Description
End Property

Public Property Get Region() As String
On Error GoTo Region_Err
    Region = m_Region
    Exit Property
Region_Err:
    Err.Raise Err.Number, Err.Source & " (CUSTOMERSClass.Region.Get)", Err.Description
End Property

'*****
'* Property: Postalcode
'* Based on the field Postalcode of dbo.Customers table
'* Let and Get methods
'* Purpose: access Postalcode property
'*****
Public Property Let Postalcode(ByVal val As String)
On Error GoTo Postalcode_Err
    m_Postalcode = val
    Exit Property
Postalcode_Err:
    Err.Raise Err.Number, Err.Source & " (CUSTOMERSClass.Postalcode.Let)",
Err.Description
End Property

Public Property Get Postalcode() As String
On Error GoTo Postalcode_Err
    Postalcode = m_Postalcode
    Exit Property
Postalcode_Err:
    Err.Raise Err.Number, Err.Source & " (CUSTOMERSClass.Postalcode.Get)",
Err.Description
End Property

'*****
'* Property: Country
'* Based on the field Country of dbo.Customers table
'* Let and Get methods
'* Purpose: access Country property
'*****
Public Property Let Country(ByVal val As String)
On Error GoTo Country_Err
    m_Country = val
    Exit Property
Country_Err:
    Err.Raise Err.Number, Err.Source & " (CUSTOMERSClass.Country.Let)", Err.Description
End Property

Public Property Get Country() As String
On Error GoTo Country_Err
    Country = m_Country
    Exit Property
Country_Err:
    Err.Raise Err.Number, Err.Source & " (CUSTOMERSClass.Country.Get)", Err.Description

```

```

End Property

'*****
'* Property: Phone
'* Based on the field Phone of dbo.Customers table
'* Let and Get methods
'* Purpose: access Phone property
'*****
Public Property Let Phone(ByVal val As String)
On Error GoTo Phone_Err
    m_Phone = val
    Exit Property
Phone_Err:
    Err.Raise Err.Number, Err.Source & " (CUSTOMERSClass.Phone.Let)", Err.Description
End Property

Public Property Get Phone() As String
On Error GoTo Phone_Err
    Phone = m_Phone
    Exit Property
Phone_Err:
    Err.Raise Err.Number, Err.Source & " (CUSTOMERSClass.Phone.Get)", Err.Description
End Property

'*****
'* Property: Fax
'* Based on the field Fax of dbo.Customers table
'* Let and Get methods
'* Purpose: access Fax property
'*****
Public Property Let Fax(ByVal val As String)
On Error GoTo Fax_Err
    m_Fax = val
    Exit Property
Fax_Err:
    Err.Raise Err.Number, Err.Source & " (CUSTOMERSClass.Fax.Let)", Err.Description
End Property

Public Property Get Fax() As String
On Error GoTo Fax_Err
    Fax = m_Fax
    Exit Property
Fax_Err:
    Err.Raise Err.Number, Err.Source & " (CUSTOMERSClass.Fax.Get)", Err.Description
End Property

Public Property Get RS() As ADODB.Recordset
    Set RS = objRS
End Property

'*****
'* Function Insert will insert new record into table dbo.Customers
'*****
Public Function Insert()
Dim sSQL As String
    On Error GoTo Insert_Error
    getDbConnection
    sSQL="insert into dbo.Customers " + _
        "
(CUSTOMERID,COMPANYNAME,CONTACTNAME,CONTACTTITLE,ADDRESS,CITY,REGION,POSTALCODE,COUNTRY,PHONE,FAX)
" + _
        "values (" + SqlString(m_Customerid,adVarChar ) + _
        " ,"+ SqlString(m_Companyname,adVarChar ) + _
        " ,"+ SqlString(m_Contactname,adVarChar ) + _
        " ,"+ SqlString(m_Contacttitle,adVarChar ) + _
        " ,"+ SqlString(m_Address,adVarChar ) + _
        " ,"+ SqlString(m_City,adVarChar ) + _
        " ,"+ SqlString(m_Region,adVarChar ) + _
        " ,"+ SqlString(m_Postalcode,adVarChar ) + _

```

```

        " ," + SqlString(m_Country,adVarChar ) + _
        " ," + SqlString(m_Phone,adVarChar ) + _
        " ," + SqlString(m_Fax,adVarChar ) + _
        ")"
objConn.Execute sSql, , adExecuteNoRecords

If Not objCtx Is Nothing Then objCtx.SetComplete
Exit Function
Insert_Error:
If Not objCtx Is Nothing Then objCtx.SetAbort
Err.Raise Err.Number, Err.Source & " (CUSTOMERSClass.Insert)", Err.Description
End Function

'*****
'* Function Update will update corresponding record in table dbo.Customers
'*****
Public Function Update(ByVal sCustomerid As String)
Dim sSQL As String
On Error GoTo Update_Error
getDbConnection
sSQL="update dbo.Customers set " + _
    " Customerid = " + SqlString(m_Customerid,adVarChar ) + _
    " ,Companyname = " + SqlString(m_Companyname,adVarChar ) + _
    " ,Contactname = " + SqlString(m_Contactname,adVarChar ) + _
    " ,Contacttitle = " + SqlString(m_Contacttitle,adVarChar ) + _
    " ,Address = " + SqlString(m_Address,adVarChar ) + _
    " ,City = " + SqlString(m_City,adVarChar ) + _
    " ,Region = " + SqlString(m_Region,adVarChar ) + _
    " ,Postalcode = " + SqlString(m_Postalcode,adVarChar ) + _
    " ,Country = " + SqlString(m_Country,adVarChar ) + _
    " ,Phone = " + SqlString(m_Phone,adVarChar ) + _
    " ,Fax = " + SqlString(m_Fax,adVarChar ) + _
    " where CustomerID = " + SqlString(sCustomerid,adVarChar )
objConn.Execute sSql, , adExecuteNoRecords

If Not objCtx Is Nothing Then objCtx.SetComplete
Exit Function
Update_Error:
If Not objCtx Is Nothing Then objCtx.SetAbort
Err.Raise Err.Number, Err.Source & " (CUSTOMERSClass.Update)", Err.Description
End Function

'*****
'* Function Delete will delete corresponding record from table dbo.Customers
'*****
Public Function Delete(ByVal sCustomerid As String)
Dim sSQL As String
On Error GoTo Delete_Error
getDbConnection
sSQL="delete from dbo.Customers where CustomerID = " + SqlString(sCustomerid,adVarChar
)
objConn.Execute sSql, , adExecuteNoRecords

If Not objCtx Is Nothing Then objCtx.SetComplete
Exit Function
Delete_Error:
If Not objCtx Is Nothing Then objCtx.SetAbort
Err.Raise Err.Number, Err.Source & " (CUSTOMERSClass.Delete)", Err.Description
End Function

'*****
'* Function Load will load corresponding record into object from table dbo.Customers
'*****
Public Function Load(ByVal sCustomerid As String)

```

```

Dim sSQL As String
On Error GoTo Load_Error
getDbConnection
sSQL="select * from dbo.Customers where CustomerID = " +
SqlString(sCustomerid,adVarChar )
Set objRS = objConn.Execute(sSql)
If Not objRS.EOF Then
    populate
    Load = True
Else
    Load = False
End If

Exit Function
Load_Error:
Set objRS = Nothing
If Not objCtx Is Nothing Then objCtx.SetAbort
Err.Raise Err.Number, Err.Source & " (CUSTOMERSClass.Load)", Err.Description

End Function

'*****
'* Function populate will populate properties from corresponding fields of the
'* Recordset.
'*****
Private Function populate()
    clean
    If Not objRS Is Nothing Then
        If Not objRS.EOF Then
            if(Not isNull(objRS!Customerid)) Then m_Customerid=objRS!Customerid
            if(Not isNull(objRS!Companyname)) Then m_Companyname=objRS!Companyname
            if(Not isNull(objRS!Contactname)) Then m_Contactname=objRS!Contactname
            if(Not isNull(objRS!Contacttitle)) Then m_Contacttitle=objRS!Contacttitle
            if(Not isNull(objRS!Address)) Then m_Address=objRS!Address
            if(Not isNull(objRS!City)) Then m_City=objRS!City
            if(Not isNull(objRS!Region)) Then m_Region=objRS!Region
            if(Not isNull(objRS!Postalcode)) Then m_Postalcode=objRS!Postalcode
            if(Not isNull(objRS!Country)) Then m_Country=objRS!Country
            if(Not isNull(objRS!Phone)) Then m_Phone=objRS!Phone
            if(Not isNull(objRS!Fax)) Then m_Fax=objRS!Fax
        End If
    End If
End Function

'*****
'* Function clean will clean current properties of the object.
'*****
Private Function clean()
    m_Customerid=""
    m_Companyname=""
    m_Contactname=""
    m_Contacttitle=""
    m_Address=""
    m_City=""
    m_Region=""
    m_Postalcode=""
    m_Country=""
    m_Phone=""
    m_Fax=""
End Function

'*****
'* Function MoveNext will move recordset position to the next record and populate
'* properties. Will return False on EOF of the Recordset.
'*****
Public Function MoveNext() As Boolean
    MoveNext = False
    If Not objRS Is Nothing Then
        If Not objRS.EOF Then
            objRS.MoveNext
            populate
        End If
    End If
End Function

```

```

        MoveNext = True
    End If
End If
End Function

'* Function Name:   getDbConnection
'* Function Description: First time create new Connection,
'*                  on other invocations reuse existing one.
'* Input Parameters:
'* Return Value:   ADODB.Connection
*****
Private Function getDbConnection() As ADODB.Connection
'* TODO: rewrite getConnection method if you want to use
'*       other than ODBC methods of database connection.
    On Error GoTo Connection_Error
    If objConn Is Nothing Then
        Set objConn = New ADODB.Connection
        objConn.Open "PROVIDER=MSDASQL;DSN=northwind;UID=sa;PWD="
    End If
    Set getDbConnection = objConn

    Exit Function
Connection_Error:
    Set objConn=Nothing
    Err.Raise Err.Number, Err.Source & " (CUSTOMERSClass.getDbConnection)",
Err.Description

End Function

*****
'* Function Name:   setXML
'* Function Description: Set object properties from XML string
'* Input Parameters:ByVal xml As String
'* Return Value:
*****
Public Function setXML(ByVal sXml As String)
Dim objDom As DOMDocument30
Dim objNode As IXMLDOMNode

    On Error GoTo setXML_Error

    Set objDom = CreateObject("MSXML2.DOMDocument.3.0")
    objDom.loadXML sXml
    Set objNode = objDom.selectSingleNode("CUSTOMERSDOC/CUSTOMERS")

    m_Customerid = getNodeData(objNode, "CUSTOMERID")
    m_Companyname = getNodeData(objNode, "COMPANYNAME")
    m_Contactname = getNodeData(objNode, "CONTACTNAME")
    m_Contacttitle = getNodeData(objNode, "CONTACTTITLE")
    m_Address = getNodeData(objNode, "ADDRESS")
    m_City = getNodeData(objNode, "CITY")
    m_Region = getNodeData(objNode, "REGION")
    m_Postalcode = getNodeData(objNode, "POSTALCODE")
    m_Country = getNodeData(objNode, "COUNTRY")
    m_Phone = getNodeData(objNode, "PHONE")
    m_Fax = getNodeData(objNode, "FAX")
    Set objDom = Nothing
    Set objNode = Nothing
    Exit Function

setXML_Error:
    Err.Raise Err.Number, Err.Source & " (CUSTOMERSClass.setXML)", Err.Description
End Function

*****
'* Function Name:   getXML
'* Function Description: Generates XML string from object properties
'* Input Parameters:
'* Return Value:String
'* Author: Max Chernyshov
*****

```

```

Public Function getXML() As String
Dim objDom As DOMDocument30
Dim objNode As IXMLDOMNode
Dim objRoot As IXMLDOMELEMENT
    On Error GoTo getXML_error

    Set objDom = CreateObject("MSXML2.DOMDocument.3.0")
    Set objRoot = objDom.createElement("CUSTOMERSDOC")
    Set objDom.documentElement = objRoot
    Set objRoot = appendNode(objRoot, "CUSTOMERS", "")

    appendNode objRoot, "CUSTOMERID", m_Customerid
    appendNode objRoot, "COMPANYNAME", m_Companyname
    appendNode objRoot, "CONTACTNAME", m_Contactname
    appendNode objRoot, "CONTACTTITLE", m_Contacttitle
    appendNode objRoot, "ADDRESS", m_Address
    appendNode objRoot, "CITY", m_City
    appendNode objRoot, "REGION", m_Region
    appendNode objRoot, "POSTALCODE", m_Postalcode
    appendNode objRoot, "COUNTRY", m_Country
    appendNode objRoot, "PHONE", m_Phone
    appendNode objRoot, "FAX", m_Fax
    getXML = objDom.Xml
    Set objDom = Nothing
    Set objRoot = Nothing
    Exit Function

getXML_Error:
    Err.Raise Err.Number, Err.Source & " (CUSTOMERSClass.getXML)", Err.Description

End Function

'*****
'* Function Name: getNodeData
'* Function Description: get XMLDOM node data
'* Input Parameters:parent As IXMLDOMNode, ByVal nodeName As String
'* Return Value:String
'*****
Private Function getNodeData(objParent As IXMLDOMNode, ByVal sNodeName As String) As String
Dim objNode As IXMLDOMNode
    Set objNode = objParent.selectSingleNode(sNodeName)
    if Not objNode is Nothing Then
        getNodeData = objNode.Text
    Else
        getNodeData = ""
    End If

    Set objNode = Nothing
End Function

'*****
'* Function Name: appendNode
'* Function Description: appends XMLDOM node
'* Input Parameters:parent - parent node, Name - name of new node, Value - value of new node
'* Return Value:String
'*****
Private Function appendNode(ByRef objParent As IXMLDOMELEMENT, ByVal sName As String, ByVal sValue As String) As IXMLDOMNode
Dim objNode As IXMLDOMNode
    Set objNode = objParent.ownerDocument.createElement(NODE_ELEMENT, sName, objParent.ownerDocument.documentElement.namespaceURI)
    objNode.Text = sValue
    objParent.appendChild objNode
    Set appendNode = objNode
End Function

```

```

'*****
'* Function Name: SqlString
'* Function Description: fix string parameter for SQL query
'* Input Parameters:sValue - SQL value, dbType - SQL type
'* Return Value:String
'*****
Private Function SqlString(ByVal sValue As Variant, ByVal dbType As DataTypeEnum) As
String
    On Error GoTo SqlString_Error
        If dbType = adBigInt Or dbType = adDecimal Or dbType = adInteger Or dbType =
adDouble Or dbType = adNumeric Then
            If IsNull(sValue) Or sValue = "" Then
                SqlString = "0"
            Else
                SqlString = CStr(sValue)
            End If
        ElseIf dbType = adDate Or dbType = adDBDate Or dbType = adDBTimeStamp Then
            If IsNull(sValue) Or sValue = "" Then
                SqlString = "NULL"
            Else
                SqlString = "{d'" + Format(sValue,"yyyy-mm-dd") + "'}"
            End If
        Else 'Assume String
            If IsNull(sValue) Or sValue = "" Then
                SqlString = ""
            Else
                'Fix the single Quotes, if any
                sValue = Replace(sValue, "'", "'")
                SqlString = "'" + sValue + "'"
            End If
        End If
    Exit Function

SqlString_Error:
    Err.Raise Err.Number, Err.Source & " (CUSTOMERSClass.SqlString)", Err.Description

End Function

Private Sub Class_Terminate()
    Set objConn = Nothing
    Set objRS = Nothing
    Set objCtx = Nothing
End Sub

'*****
' The purpose of this ObjectControl code is to provide a slight
' simplification to the MTS component programmer. In most cases,
' All you will need to do in your methods is to decide whether to
' call ObjCtx.SetComplete or ObjCtx.SetAbort before exiting a method.
'
' This method is not used by MTS. Future versions must not pass true.
Private Function ObjectControl_CanBePooled() As Boolean
    ObjectControl_CanBePooled = False
End Function

Private Sub ObjectControl_Activate()
    Set objCtx = GetObjectContext()
End Sub

Private Sub ObjectControl_Deactivate()
    Set objCtx = Nothing
End Sub

```

Java Code Example

```

//*****
// The CUSTOMERS class is generated to handle database operations
// with the dbo.Customers table. It also able to serialize itself to Xml.
// This class uses Xerces2 Java Xml parser,
// which can be downloaded from http://xml.apache.org/xerces2-j/index.html
//*****
// Following classes are used for Xml parsing:
import org.apache.xerces.parsers.DOMParser;
import org.apache.xerces.dom.DocumentImpl;
import org.w3c.dom.Document;
import org.w3c.dom.Node;
import org.w3c.dom.Element;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;
import org.xml.sax.InputSource;
import org.apache.xml.serialize.OutputFormat;
import org.apache.xml.serialize.Serializer;
import org.apache.xml.serialize.SerializerFactory;
import org.apache.xml.serialize.XMLSerializer;

//These Packages are used for data manipulation and Database access
import java.util.Date;
import java.sql.*;
import java.text.*;
import java.io.*;

public class CUSTOMERS{
    private Connection dbconn; // Reusable db connection object (Singleton pattern)

    private String      s_Customerid;
    private String      s_Companyname;
    private String      s_Contactname;
    private String      s_Contacttitle;
    private String      s_Address;
    private String      s_City;
    private String      s_Region;
    private String      s_Postalcode;
    private String      s_Country;
    private String      s_Phone;
    private String      s_Fax;
//*****
* Access methods for Customerid field of dbo.Customers table
*****/
    public void setCustomerid(String val){
        this.s_Customerid = val;
    }

    public String getCustomerid(){
        return this.s_Customerid;
    }
//*****
* Access methods for Companyname field of dbo.Customers table
*****/
    public void setCompanyname(String val){
        this.s_Companyname = val;
    }

    public String getCompanyname(){
        return this.s_Companyname;
    }
//*****
* Access methods for Contactname field of dbo.Customers table
*****/
    public void setContactname(String val){
        this.s_Contactname = val;
    }

    public String getContactname(){

```

```

        return this.s_Contactname;
    }
}
/*****
 * Access methods for Contacttitle field of dbo.Customers table
 *****/
public void setContacttitle(String val){
    this.s_Contacttitle = val;
}

public String getContacttitle(){
    return this.s_Contacttitle;
}
/*****
 * Access methods for Address field of dbo.Customers table
 *****/
public void setAddress(String val){
    this.s_Address = val;
}

public String getAddress(){
    return this.s_Address;
}
/*****
 * Access methods for City field of dbo.Customers table
 *****/
public void setCity(String val){
    this.s_City = val;
}

public String getCity(){
    return this.s_City;
}
/*****
 * Access methods for Region field of dbo.Customers table
 *****/
public void setRegion(String val){
    this.s_Region = val;
}

public String getRegion(){
    return this.s_Region;
}
/*****
 * Access methods for Postalcode field of dbo.Customers table
 *****/
public void setPostalcode(String val){
    this.s_Postalcode = val;
}

public String getPostalcode(){
    return this.s_Postalcode;
}
/*****
 * Access methods for Country field of dbo.Customers table
 *****/
public void setCountry(String val){
    this.s_Country = val;
}

public String getCountry(){
    return this.s_Country;
}
/*****
 * Access methods for Phone field of dbo.Customers table
 *****/
public void setPhone(String val){
    this.s_Phone = val;
}

public String getPhone(){
    return this.s_Phone;
}

```

```

    }
    /*****
    * Access methods for Fax field of dbo.Customers table
    *****/
    public void setFax(String val){
        this.s_Fax = val;
    }

    public String getFax(){
        return this.s_Fax;
    }

    /*****
    * Function Insert will insert new record into table dbo.Customers
    *****/
    public void insert()
    {
        String sSQL;
        Statement stmt;
        try{
            sSQL="insert into dbo.Customers " +
            "(Customerid,Companyname,Contactname,Contacttitle,Address,City,Region,Postalcode,Country,Phone,Fax)"+
            "values (" + SqlString(s_Customerid,Types.VARCHAR ) +
            " ,"+ SqlString(s_Companyname,Types.VARCHAR ) +
            " ,"+ SqlString(s_Contactname,Types.VARCHAR ) +
            " ,"+ SqlString(s_Contacttitle,Types.VARCHAR ) +
            " ,"+ SqlString(s_Address,Types.VARCHAR ) +
            " ,"+ SqlString(s_City,Types.VARCHAR ) +
            " ,"+ SqlString(s_Region,Types.VARCHAR ) +
            " ,"+ SqlString(s_Postalcode,Types.VARCHAR ) +
            " ,"+ SqlString(s_Country,Types.VARCHAR ) +
            " ,"+ SqlString(s_Phone,Types.VARCHAR ) +
            " ,"+ SqlString(s_Fax,Types.VARCHAR ) +
            ")";

            stmt = getConnection().createStatement();
            stmt.executeQuery(sSQL);
            stmt = null;
        }catch( SQLException e ){
            stmt = null;
            handleError(e);
            return;
        }
        return;
    }

    /*****
    * Function Update will update corresponding record in table dbo.Customers
    *****/
    public void update(String s_Customerid)
    {
        String sSQL;
        Statement stmt;
        try{
            sSQL="update dbo.Customers set " +
            " Customerid = " + SqlString(s_Customerid,Types.VARCHAR ) +
            " ,Companyname = " + SqlString(s_Companyname,Types.VARCHAR ) +
            " ,Contactname = " + SqlString(s_Contactname,Types.VARCHAR ) +
            " ,Contacttitle = " + SqlString(s_Contacttitle,Types.VARCHAR ) +
            " ,Address = " + SqlString(s_Address,Types.VARCHAR ) +
            " ,City = " + SqlString(s_City,Types.VARCHAR ) +
            " ,Region = " + SqlString(s_Region,Types.VARCHAR ) +
            " ,Postalcode = " + SqlString(s_Postalcode,Types.VARCHAR ) +
            " ,Country = " + SqlString(s_Country,Types.VARCHAR ) +
            " ,Phone = " + SqlString(s_Phone,Types.VARCHAR ) +
            " ,Fax = " + SqlString(s_Fax,Types.VARCHAR ) +
            " where CustomerID = " + SqlString(s_Customerid,Types.VARCHAR ) ;

            stmt = getConnection().createStatement();
            stmt.executeQuery(sSQL);

```

```

        stmt = null;
    }catch(SQLException e){
        stmt = null;
        handleError(e);
        return;
    }
    return;
}

/*****
 * Function Delete will delete corresponding record from table dbo.Customers
 *****/
public void delete(String s_Customerid)
{
    String sSQL;
    Statement stmt;
    try{
        sSQL="delete from dbo.Customers where CustomerID = " +
        SqlString(s_Customerid,Types.VARCHAR ) ;

        stmt = getConnection().createStatement();
        stmt.executeQuery(sSQL);
        stmt = null;
    }catch( SQLException e ){
        stmt = null;
        handleError(e);
    }
    return;
}

/*****
/* Function Load will load corresponding record into object from table dbo.Customers
 *****/
public void load(String s_Customerid)
{
    String sSQL;
    ResultSet rs;
    Statement stmt;
    try{
        sSQL="select * from dbo.Customers where CustomerID = " +
        SqlString(s_Customerid,Types.VARCHAR ) ;
        stmt = getConnection().createStatement();
        rs = stmt.executeQuery(sSQL);
        if(rs.next())
        {

            s_Customerid=rs.getString("Customerid");
            s_Companyname=rs.getString("Companyname");
            s_Contactname=rs.getString("Contactname");
            s_Contacttitle=rs.getString("Contacttitle");
            s_Address=rs.getString("Address");
            s_City=rs.getString("City");
            s_Region=rs.getString("Region");
            s_Postalcode=rs.getString("Postalcode");
            s_Country=rs.getString("Country");
            s_Phone=rs.getString("Phone");
            s_Fax=rs.getString("Fax");

        }

        stmt = null;
        rs=null;

    }catch( SQLException e ){
        stmt = null;
        rs=null;
        handleError(e);
        return;
    }
    return;
}

```

```

/*****
* Function setXml will set bean properties from passed xml string
*****/
public void setXml(String sXml)
{
    DOMParser parser;
    Element docElement;
    try{
        parser = new DOMParser();
        parser.parse(new InputSource(new StringReader(sXml)));
        docElement = parser.getDocument().getDocumentElement();
    }catch (SAXException e){
        handleError(e);
        return;
    }catch (IOException e){
        handleError(e);
        return;
    }
    Element firstElement=(Element)docElement.getElementsByTagName("CUSTOMERS").item(0);

    s_Customerid = (String)getFirstNodeValue(firstElement,"CUSTOMERID",Types.VARCHAR);
    s_Companyname = (String)getFirstNodeValue(firstElement,"COMPANYNAME",Types.VARCHAR);
    s_Contactname = (String)getFirstNodeValue(firstElement,"CONTACTNAME",Types.VARCHAR);
    s_Contacttitle = (String)getFirstNodeValue(firstElement,"CONTACTTITLE",Types.VARCHAR);
    s_Address = (String)getFirstNodeValue(firstElement,"ADDRESS",Types.VARCHAR);
    s_City = (String)getFirstNodeValue(firstElement,"CITY",Types.VARCHAR);
    s_Region = (String)getFirstNodeValue(firstElement,"REGION",Types.VARCHAR);
    s_Postalcode = (String)getFirstNodeValue(firstElement,"POSTALCODE",Types.VARCHAR);
    s_Country = (String)getFirstNodeValue(firstElement,"COUNTRY",Types.VARCHAR);
    s_Phone = (String)getFirstNodeValue(firstElement,"PHONE",Types.VARCHAR);
    s_Fax = (String)getFirstNodeValue(firstElement,"FAX",Types.VARCHAR);

}

/*****
* Function getXml will serialize bean to the xml string
*****/
public String getXml()
{
    try{
        Document doc= new DocumentImpl();
        Element root = doc.createElement("CUSTOMERSDOC");
        Element first = doc.createElement("CUSTOMERS");

        addNode(doc,first,"CUSTOMERID",s_Customerid);
        addNode(doc,first,"COMPANYNAME",s_Companyname);
        addNode(doc,first,"CONTACTNAME",s_Contactname);
        addNode(doc,first,"CONTACTTITLE",s_Contacttitle);
        addNode(doc,first,"ADDRESS",s_Address);
        addNode(doc,first,"CITY",s_City);
        addNode(doc,first,"REGION",s_Region);
        addNode(doc,first,"POSTALCODE",s_Postalcode);
        addNode(doc,first,"COUNTRY",s_Country);
        addNode(doc,first,"PHONE",s_Phone);
        addNode(doc,first,"FAX",s_Fax);

        root.appendChild(first);
        doc.appendChild(root);

        StringWriter      stringOut = new StringWriter();
        XMLSerializer      serial = new XMLSerializer( stringOut, (new OutputFormat(doc)));
        serial.asDOMSerializer();
        serial.serialize( doc.getDocumentElement() );

        return stringOut.toString();

    }catch ( Exception e ){
        handleError(e);
        return "";
    }
}
}

```

```

/*****
* Here are helper functions that can be customized to your coding standards.
*****/

private void addNode(Document doc, Element parent, String name, Object value)
{
    Element item = doc.createElement(name);
    if(value==null)
        item.appendChild( doc.createTextNode("") );
    else
        item.appendChild( doc.createTextNode(value.toString()) );
    parent.appendChild( item );
}

private String date2Str(Date value)
{
    SimpleDateFormat df;
    try{
        df = new SimpleDateFormat("MM/dd/yyyy");
        return df.format((Date)value);
    }catch(Exception e){
        handleError(e);
        return null;
    }
}

private Date str2Date(String value)
{
    SimpleDateFormat df;
    try{
        df = new SimpleDateFormat("MM/dd/yyyy");
        return df.parse(value);
    }catch(Exception e){
        handleError(e);
        return null;
    }
}

/*****
* Function getFirstNodeText will return string value of the first found Node
* with corresponding name
*****/
private Object getFirstNodeValue(Element parent, String name, int type)
{
    String val;

    try{
        val=parent.getElementsByTagName(name).item(0).getNodeValue();
        switch(type)
        {
            case Types.BIGINT :
            case Types.DECIMAL :
            case Types.NUMERIC :
            case Types.REAL :
            case Types.TINYINT :
            case Types.SMALLINT :
            case Types.BIT :
            case Types.DOUBLE :
            case Types.FLOAT :
            case Types.INTEGER :
                return new Double(val);
            case Types.DATE :
            case Types.TIMESTAMP :
                return str2Date(val);
            default:
                return val ;
        }
    }
}

```

```

    }catch(Exception e){
        handleError(e);
        return null;
    }
}

/*****
 * Function handleError needs to be updated with regular for your project
 * error-handling procedure.
 *****/
private void handleError(Exception e)
{
    //TODO: Please update the error handling procedure.
    e.printStackTrace();
}

/*****
 * Function getConnection will create Connection object or return existing one
 *****/
private Connection getConnection()
{
    Driver driver;
    if(dbconn == null){
        try{
            driver = (Driver)Class.forName("sun.jdbc.odbc.JdbcOdbcDriver").newInstance();
        }catch( Exception e ){
            handleError(e);
            return null;
        }

        try{
            dbconn = DriverManager.getConnection("jdbc:odbc:northwind","sa","");
        }catch( SQLException e ){
            handleError(e);
            return null;
        }
    }
    return dbconn;
}

/*****
 * Function SqlString will generate valid SQL parameter
 *****/
private String SqlString(Object val, int type)
{
    SimpleDateFormat df;

    if (val == null) return "NULL";

    switch(type)
    {
        case Types.BIGINT :
        case Types.DECIMAL :
        case Types.NUMERIC :
        case Types.REAL :
        case Types.TINYINT :
        case Types.SMALLINT :
        case Types.BIT :
        case Types.DOUBLE :
        case Types.FLOAT :
        case Types.INTEGER :
            return val.toString();
        case Types.LONGVARCHAR :
        case Types.VARCHAR :
        case Types.CHAR :
            return "'" +val.toString() + "'";
        case Types.DATE :
            df = new SimpleDateFormat ("yyyy-MM-dd");
            return "{d'" + df.format((Date)val) + "'}";
    }
}

```

```

    case Types.TIME :
        df = new SimpleDateFormat ("hh:mm:ss");
        return "{t'" + df.format((Date)val) + "'}";
    case Types.TIMESTAMP :
        df = new SimpleDateFormat ("yyyy-MM-dd hh:mm:ss");
        return "{ts'" + df.format((Date)val) + "'}";
    case Types.BINARY :
    case Types.BLOB :
    case Types.CLOB :
    case Types.LONGVARBINARY :
    case Types.VARBINARY :
        return "'MEMO'";
        //TODO: Custom LOB implementation required;
    }
    return val.toString();
}

//TODO: add other helper functions here.

/*
public static void main(String argv[])
{
    CUSTOMERS obj = new CUSTOMERS();
    Double val= new Double(1);
    obj.load(val);
    System.out.println(obj.getXml());
}
*/
}

```