uniservice.asm.cs

-----------------

using System;

using System.Collections;

using System.ComponentModel;

using System.Data;

using System.Linq;

using System.Web;

using System.Web.Services;

using System.Web.Services.Protocols;

using System.Xml.Linq;

namespace UniWebService

{

/// <summary>

/// Summary description for UniWebService

/// </summary>

[WebService(Namespace = "http://tempuri.org/")]

[WebServiceBinding(ConformsTo = WsiProfiles.BasicProfile1\_1)]

[ToolboxItem(false)]

// To allow this Web Service to be called from script, using ASP.NET AJAX, uncomment the following line.

// [System.Web.Script.Services.ScriptService]

public class UniService : System.Web.Services.WebService

{

//sans XML

[WebMethod]

public UniServiceResult record\_Get(UserAuthentication user, string cultureCode, string recordId)

{

UniServiceResult result = new UniServiceResult();

try

{

AuthenticateUser(user);

//result.DataSetResult = DataAccessLayer.Factory.Default.record\_Get(cultureCode, userId, sessionId, recordId, viewStatus, viewMessage);

result.DataSetResult = DataAccessLayer.Factory.Default.record\_Get(cultureCode, recordId);

}

catch (Exception err)

{

result.Exception = new UniServiceException(err);

}

return result;

}

//avec xml

[WebMethod]

public UniServiceResult record\_Update(UserAuthentication user, string cultureCode, string xmlrecord,

string userName)

{

UniServiceResult result = new UniServiceResult();

try

{

AuthenticateUser(user);

result.DataSetResult = DataAccessLayer.Factory.Default.record\_Update(cultureCode, xmlrecord,

userName);

}

catch (Exception err)

{

result.Exception = new UniServiceException(err);

}

return result;

}

unidatalayer.cs

---------------

using System;

using System.Collections.Generic;

using System.Text;

using IBMU2.UODOTNET;

using DataAccessLayer.UniObjectData;

using System.Data;

using DataAccessLayer.Util;

namespace DataAccessLayer

{

public class UniDataLayer : BaseDataLayer

{

**exemple sans XML**

----------------

public override DataSet record\_Get(string cultureCode, string recordid)

{

UniSession session = UniCommunication.CreateConnection();

try

{

StringCollection args = new StringCollection();

args.Add(cultureCode);

args.Add(recordid);

System.Data.DataSet ds = UniCommunication.GetDataSetFromSubRoutine(session, "**UDT\_RECORD\_READ**", args);

return ds;

}

catch (Exception ex)

{

throw ex;

}

finally

{

UniObjects.CloseSession(session);

}

}

**exemple avec XML - (comme je disais, le XML est passée comme une string)**

---------------

public override DataSet record\_update(string cultureCode, string xmlInput, string userName)

{

UniSession session = UniCommunication.CreateConnection();

try

{

StringCollection args = new StringCollection();

args.Add(cultureCode);

args.Add(xmlInput);

args.Add(userName);

System.Data.DataSet ds = UniCommunication.GetDataSetFromSubRoutine(session, "**UDT\_RECORD\_UPDATE**", args);

return ds;

}

catch (Exception ex)

{

throw ex;

}

finally

{

UniObjects.CloseSession(session);

}

}

unicommunication.cs (méthodes appelés par unidatalayer.cs

-------------------

using System;

using System.Collections.Generic;

using System.Text;

using System.Data;

using System.Xml;

using IBMU2.UODOTNET;

using DataAccessLayer.Util;

public static UniSession CreateConnection(string Host, int Port, string User, string Password, string Account, string Type)

{

//Connection Pooling information should be loaded from Settings.

UniObjects.UOPooling = bool.Parse(Properties.Settings.Default.UniPooling);

UniObjects.MinPoolSize = Convert.ToInt32(Properties.Settings.Default.UniPoolMinSize);

UniObjects.MaxPoolSize = Convert.ToInt32(Properties.Settings.Default.UniPoolMaxSize);

UniObjects.PoolingOpenSessionTimeOut = Convert.ToInt32(Properties.Settings.Default.UniOpenSessionTimeOut);

UniObjects.IdleRemoveExecInterval = Convert.ToInt32(Properties.Settings.Default.UniIdleRemoveExecInterval);

UniObjects.IdleRemoveThreshold = Convert.ToInt32(Properties.Settings.Default.UniIdleRemoveThreshold);

return UniObjects.OpenSession(Host, 31438, User, Password, Account, "udcs");

}

public static DataSet GetDataSetFromSubRoutine(UniSession Session, string RoutineName, StringCollection Arguments)

{

string XmlString = GetStringFromSubRoutine(Session, RoutineName, Arguments);

return XmlFactory.DataSetFromXML(XmlString);

}

public static XmlDocument GetXmlDocumentFromSubRoutine(UniSession Session, string RoutineName, StringCollection Arguments)

{

XmlDocument XmlDoc = new XmlDocument();

string XmlString = GetStringFromSubRoutine(Session, RoutineName, Arguments);

XmlDoc.LoadXml(XmlString);

return XmlDoc;

}

public static XmlDocument GetXmlDocumentFromSubRoutine(string RoutineName, StringCollection Arguments)

{

XmlDocument XmlDoc = new XmlDocument();

string XmlString = GetStringFromSubRoutine(RoutineName, Arguments);

XmlDoc.LoadXml(XmlString);

return XmlDoc;

}

public static string GetStringFromSubRoutine(UniSession Session, string RoutineName, StringCollection Arguments)

{

int IntTotalArgs = Arguments.Count + 1;//Get the count of the arguments

//Create the UniSubroutine object.

UniSubroutine sub = Session.CreateUniSubroutine(RoutineName, IntTotalArgs);

sub.SetArg(0, "");//First argument of all the sub routines should be the return variable

int i = 1;

foreach (string argument in Arguments)

{

sub.SetArg(i++, argument);

}

sub.Call();

return sub.GetArg(0);

}

public static string GetStringFromSubRoutine(string RoutineName, StringCollection Arguments)

{

UniSession us1 = null;

string response = "";

try

{

us1 = UniCommunication.CreateConnection();

response = GetStringFromSubRoutine(us1, RoutineName, Arguments);

}

catch (Exception e)

{

if (us1 != null && us1.IsActive)

{

UniObjects.CloseSession(us1);

us1 = null;

}

throw e;

}

finally

{

if (us1 != null && us1.IsActive)

{

UniObjects.CloseSession(us1);

us1 = null;

}

}

return response;

}

xmlfactory.cs

-------------

using System;

using System.Collections.Generic;

using System.Text;

using System.Data;

using System.Xml;

using System.Xml.Serialization;

namespace DataAccessLayer.Util

{

public class XmlFactory

{

/// <summary>

/// Gets an XmlNodeReader from a string of XML

/// </summary>

/// <param name="xml"></param>

/// <returns></returns>

public static XmlNodeReader XmlReaderFromXML(string xml)

{

XmlDocument xmldoc = new XmlDocument();

xmldoc.LoadXml(xml);

return XmlReaderFromXML(xmldoc);

}

/// <summary>

/// Gets an XmlNodeReader from an XmlDocument Object

/// </summary>

/// <param name="xml"></param>

/// <returns></returns>

public static XmlNodeReader XmlReaderFromXML(XmlDocument xml)

{

XmlNode rootnode = xml.DocumentElement;

XmlNodeReader reader = new XmlNodeReader(rootnode);

return reader;

}

/// <summary>

/// Gets an XmlNodeReader from a string of XML

/// </summary>

/// <param name="filename"></param>

/// <returns></returns>

public static XmlNodeReader XmlReaderFromFile(string filename)

{

XmlDocument xmldoc = new XmlDocument();

xmldoc.Load(filename);

return XmlReaderFromXML(xmldoc);

}

/// <summary>

/// Loads a Generic DataSet from an XML string

/// </summary>

/// <param name="xml"></param>

/// <returns></returns>

public static DataSet DataSetFromXML(string xml)

{

XmlDocument xmldoc = new XmlDocument();

xmldoc.LoadXml(xml);

return DataSetFromXML(xmldoc);

}

/// <summary>

/// Loads a Generic DataSet from an XmlDocument Object

/// </summary>

/// <param name="xml"></param>

/// <returns></returns>

public static DataSet DataSetFromXML(XmlDocument xml)

{

XmlNodeReader reader = XmlReaderFromXML(xml);

DataSet ds = new DataSet();

ds.ReadXml(reader);

return ds;

}

/// <summary>

/// Replaces/Adds namespace to XML document

/// </summary>

/// <param name="oldDom"></param>

/// <param name="NewNamespaceURI"></param>

/// <returns></returns>

public static XmlDocument ReplaceDocumentNamespace(XmlDocument oldDom, string NewNamespaceURI)

{

oldDom.DocumentElement.SetAttribute("xmlns", NewNamespaceURI);

// must serialize and reload the DOM

// before this will actually take effect

XmlDocument newDom = new XmlDocument();

newDom.LoadXml(oldDom.OuterXml);

return newDom;

}

/// <summary>

/// Replaces/Adds namespace to XML document

/// </summary>

/// <param name="oldDom"></param>

/// <param name="NewNamespaceURI"></param>

/// <returns></returns>

public static string GetCleanDatasetXml(DataSet ds)

{

StringBuilder sb = new StringBuilder();

XmlWriter write = XmlWriter.Create(sb);

ds.WriteXml(write);

XmlDocument doc = new XmlDocument();

doc.LoadXml(sb.ToString());

doc.DocumentElement.RemoveAllAttributes();

return doc.OuterXml;

}

/// <summary>

/// Replaces/Adds namespace to XML document

/// </summary>

/// <param name="oldDom"></param>

/// <param name="NewNamespaceURI"></param>

/// <returns></returns>

public static string GetCleanDataTableXml(DataTable dt)

{

StringBuilder sb = new StringBuilder();

XmlWriter write = XmlWriter.Create(sb);

dt.WriteXml(write);

XmlDocument doc = new XmlDocument();

doc.LoadXml(sb.ToString());

doc.DocumentElement.RemoveAllAttributes();

return doc.OuterXml;

}

public static string GetXmlSerialization(object obj)

{

try

{

XmlSerializer ser = new XmlSerializer(obj.GetType());

StringBuilder build = new StringBuilder();

XmlWriter writer = XmlWriter.Create(build);

ser.Serialize(writer, obj);

return build.ToString();

}

catch (Exception err)

{

return err.Message;

}

}

/// <summary>

/// Checks XML document for errors thrown from XML.FATAL.ERROR pick program

/// </summary>

/// <param name="xml"></param>

public static void XmlErrorCheck(string xml)

{

if (xml.TrimStart().StartsWith("<error"))

{

XmlDocument doc = new XmlDocument();

doc.LoadXml(xml);

XmlErrorCheck(doc);

}

}/// <summary>

/// Checks XML document for errors thrown from XML.FATAL.ERROR pick program

/// </summary>

/// <param name="xml"></param>

public static void XmlErrorCheck(XmlDocument xml)

{

XmlNode rootNode = xml.DocumentElement;

string message = "";

string type = "0";

string program = "";

if (rootNode.Name.ToString().Equals("error"))

{

try

{

type = rootNode.Attributes["type"].Value.ToString();

}

catch { }

try

{

program = rootNode.Attributes["program"].Value.ToString();

}

catch { }

foreach (XmlNode n in rootNode.ChildNodes)

{

if (n.Name.ToString().Equals("msg"))

{

message += n.InnerText + "\n";

}

}

}

if (message.Trim().Length > 0)

{

throw new Exception(message);

}

}

}

}