make.tol de Xlsx.Reader

Librería que permite construir lectores de ficheros Excel Xlx, de extensión xlsx, de Office de Microsoft. El objetivo de este fichero make.tol es sólo probar las diferentes funcionalidades de esta librería que se implementan en el fichero xlx.tol y que puede ser incluida desde otros programas Tol. Los actuales ficheros Xlsx de Microsoft Excel son ficheros comprimidos y se pueden abrir con WinZip o 7Zip. En especial, este código, usa 7Zip, desde la línea de mandatos, para extraer la hoja de datos requerida. Las hojas dentro de este fichero comprimido se numeran como 1, 2, 3,... independiéntemente de cual sea el nombre de la hoja.

Xls.Reader utiliza un mandato 7Xip para extraer el contenido, por ejemplo, para extraer la hoja 1 el mandato es: 7z.exe e excel.xlsx xl\worksheets\sheet1.xml -so > _temporal_.xml. Este mandato extrae la primera hoja (sheet1.xml) del fichero llamado excel.xlsx y la guarda por volcado de la salida estándar con el nombre de fichero _temporal_.xml. La hoja que se extrae de esta forma es un fichero Xml con los tags <sheetData>, <row> y <v> (value) que determinan: a) el primero <sheetData> todos los datos del fichero Excel, b) el segundo <row> cada una de las filas, que son un conjunto horizontal de celdas, y c) el tercero <v> cada uno de los valores de cada celda. Por lo que este fichero Xml puede convertirse en un conjunto de conjunto de textos, esto es, en una tabla Tol.

El código de esta librería Xls.Reader, por razones de simplicidad, no realiza especiales comprobaciones: a) ni de mayúsculas ni minúsculas en las etiquetas Xml, b) ni de celdas Excel combinadas, c) omite las celdas vacías que no se guarden, por lo que el resultado puede no ser perféctamente rectangular, d) tampoco extrae fórmulas, ni formatos y e) todos los campos los retorna en formato de texto, asi, por ejemplo, las fechas se retornan como un número (identificador de la fecha) en formato de texto y, f) dependiendo de la variable de control XlxCpm, retorna todos los textos compactados o no, de forma independiente a las directrices internas de Excel-Xml del tipo xml:space = preserve.

A partir de las funciones de la librería Xls.Reader se pueden programar lectores de de Excel Xlx más avanzados en lenguaje Tol, pudiendo ser una buena base de partida. Esta libraría utiliza un único fichero temporal, que usa y cuando deja de ser útil no se borra pues su contenido puede facilitar la comprobación de los resultados. Por esta razón no puede emplearse en extracciones en paralelo, para ello, habría que generar nombres aleatorios de ficheros y borrarlos después. El funcionamiento de este librería ha sido probado con las versiones de Tol 1.1.1, 1.1.5, 1.1.6 y 2.0.1.

Dentro de esta librería Xls.Reader hay 2 tipos de funciones: a) la que retorna directamente los contenidos de las celdas y b) la que si el contenido de la celda no es un valor sino es el índice a la tabla de shared strings realiza la búsqueda en esa tabla y retorna los contenidos reales. Para ocupar menos espacio en Excel Xlx los datos de tipo texto, en vez de guardarse junto con el resto, se guardan en un fichero externo, de forma que si un texto aparece en más de una celda todas lo comparten. De esta forma en la celda de Excel hay números, fechas,... pero cuando se trata de un texto lo que hay es un apuntador a la posición a una lista de textos compartidos (shared strings). Para entender el uso de la tabla de sharedStrings en Excel Xlsx puede consultarse el sitio web: http://www.sadev.co.za/content en reading-and-writing-excel-2007-or-excel-2010-c-series-index. La lectura de un Excel con shared strings es,

naturalmente debido a este doble acceso, más lenta que si no los tiene.

Árbol de ficheros

Xlsx.Reader funciones de lectura desde Tol de hojas en Excel Xlsx de Microsoft

- ← make.tol programa de test de las funciones de lectura de hojas Xlsx
- ← make.bat mandato de ejecución del programa de test de lectura Excel
- tol directorios de funciones desarrolladas en Time Oriented Languaje
  - cmm directorio con las funciones comunes
    - ← xlx.tol librería de funciones para descomprimir y leer Excel Xlx
  - ← inc.tol fichero con la orden de inclusión del fichero Tol xlx.tol
- → excel.xlsx.descompresor.png Excel abierto y marcados los Xml de datos y strings compartidos
- → xlsx_reader.pdf documento de funciones de descompresión y lectura Xml de Excel

Declaraciones

Inclusiones

- `Set allInc`
  Inclusion de las funciones comunes.

Pruebas

- `Real timIni`
  Para controlar tiempos.
- `Real basTst`
  Leer un Excel Xlsx sencillo, con celdas vacias, lo que implica que el resultado no es rectangular.
- `Real intTst`
  Pruebas de integridad de ficheros Excel en extension y contenido.
- `Real tru001`
  Leer de un Excel Xlsx una hoja interna con mas datos que los visibles.
- `Real gmeTst`
  Leer de un Excel Xlsx con shared strings.
- `Real tru002`
  Leer uno de los Excel Xlsx anteriores con shared strings.

Set allInc

```
////////////////////////////////////////////////////////////////////////////
Set  allInc = Include("tol/inc.tol");
////////////////////////////////////////////////////////////////////////////
PutDescription("Inclusion de las funciones comunes.", allInc);
////////////////////////////////////////////////////////////////////////////
```

Real timIni

```
////////////////////////////////////////////////////////////////////////////
Real timIni = Copy(Time);
////////////////////////////////////////////////////////////////////////////
PutDescription("Para controlar tiempos.", timIni);
////////////////////////////////////////////////////////////////////////////
```

Real basTst

```
////////////////////////////////////////////////////////////////////////////
Real basTst = If(FALSE, // Poner true para realizar este test
{
  Text WriteLn(XlxFromLblToEnd("0000sheetData1 2 3 4", "sheetData"));
  Text WriteLn(XlxFromTagToEnd("0000<sheetData>1 2 3", "sheetData"));
  Set  View(XlxSplitByTag("a<v>1</v>b<v>2</v>c<v>3</v>d", "v"), "");

  Text WriteLn("\nExecution\n");
  Real exe001 = Xlx7ZipExtract("dat/test01.xlsx", "1", "dat/test01.xml");

  Text WriteLn("\nReading as text\n");
  Text txtXml = XlxReadText("dat/test01.xlsx", "1");

  Text WriteLn("\nReading as table\n");
  Set  setTab = XlxReadTable("dat/test01.xlsx", "1");
  Set  View(setTab, "");

  TRUE
},FALSE);
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Leer un Excel Xlsx sencillo, con celdas vacias, lo que implica que el
resultado no es rectangular.",
basTst);
////////////////////////////////////////////////////////////////////////////
```

Real intTst

```
////////////////////////////////////////////////////////////////////////////
Real intTst = If(FALSE, // Poner true para realizar este test
{
  Set  View(SetOfReal
  (
    Xlx7ZipTest(   "dat/test01.xlsx"),          // Debe ser cierto
    XlxIsExcelXlsx("dat/test01.xlsx"),          // Debe ser cierto
    XlxIsExcelXlsx("dat/test01.xml"),           // Debe ser falso
    XlxIsExcelXlsx("dat/corrupto.xlsx")         // Debe ser falso
  ), "");

  TRUE
},FALSE);
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Pruebas de integridad de ficheros Excel en extension y contenido.",
intTst);
////////////////////////////////////////////////////////////////////////////
```

Real tru001

```
////////////////////////////////////////////////////////////////////////////
Real tru001 = If(FALSE, // Poner true para realizar este test
```

```
{
  Text WriteLn("\nReading as table\n");
  Set  setTab = XlxReadTable("dat/tru.input.xlsx",
                             "xl/externalLinks/externalLink1.xml");
  Set  setExt = Extract(setTab, 2, 3, 6, 7, 9, 15, 18, 24);
  Set  setSel = Select(setExt, Real(Set rowSet)
  { Text venIni = Sub(rowSet[3],1,3); venIni <: [["KOC","SEG","THQ"]] });

  Text filNam = "dat/tru.output.csv";
  Text filHea = "PRODUCTO;CODIGO;VENDOR;EAN;COSTE;VENTA;ACUMULADO;STOCK\n";
  Text WriteFile(filNam,filHea);
  Set  wriCic = EvalSet(setSel, Real(Set rowSet)
  {
    Text linTxt = Replace(rowSet[1],"&amp;","&") + ";" + rowSet[2] + ";" +
                  Replace(rowSet[3],"&amp;","&") + ";" + rowSet[4] + ";" +
                  Replace(rowSet[5],".",    ",") + ";" + rowSet[6] + ";" +
                        rowSet[7]                + ";" + rowSet[8] + "\n";
    Text AppendFile(filNam, linTxt);
    TRUE
  });
  SetSum(wriCic)
},FALSE);
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Leer de un Excel Xlsx una hoja interna con mas datos que los visibles.",
tru001);
////////////////////////////////////////////////////////////////////////////
```

Real gmeTst

```
////////////////////////////////////////////////////////////////////////////
Real gmeTst = If(FALSE, // Poner true para realizar este test
{
  Text WriteLn("\nReading with shared strings\n");

  Set  setTab = XlxReadTable("dat/gmeW16.xlsx", "1"); // Muchos son shared
  Set  View(setTab, "");

  Set  shaStr = XlxReadShared("dat/gmeW16.xlsx"); // Leer la tabla de shared
  Set  View(shaStr, "");

  Set  xlsTab = XlxReadTableShared("dat/gmeW16.xlsx", "1"); // Contenido real
  Set  View(xlsTab, ""); // Los valores directos y los shared localizados

  Card(xlsTab)
},FALSE);
////////////////////////////////////////////////////////////////////////////
PutDescription("Leer de un Excel Xlsx con shared strings.", gmeTst);
////////////////////////////////////////////////////////////////////////////
```

Real tru002

```
////////////////////////////////////////////////////////////////////////////
Real tru002 = If(FALSE, // Poner true para realizar este test
{
  Text WriteLn("\nReading with shared strings\n");
  Set  xlsTab = XlxReadTableShared("dat/tru.input.xlsx", "1");
  Set  View(xlsTab, "");

  Card(xlsTab)
},FALSE);
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Leer uno de los Excel Xlsx anteriores con shared strings.",
tru002);
////////////////////////////////////////////////////////////////////////////
```

xlx.tol de Xlsx.Reader

Lectura basica de ficheros Excel Xlsx. Los Xlsx son ficheros comprimidos y se pueden abrir con WinZip o 7Zip. Este codigo usa 7Zip para extraer la hoja de datos requerida 1, 2, 3,... Por ejemplo, se puede extraer el contenido de la hoja 1 con el mandato: 7z.exe e excel.xlsx xl\worksheets\sheet1.xml -so > _temporal_.xml La hoja es un fichero Xml con los tags <sheetData> <row> y <v> (value). Por lo que puede convertirse en un conjunto de conjunto de textos. Este codigo no realiza especiales comprobaciones: a) ni de mayusculas ni minusculas en las etiquetas Xml, b) ni de celdas combinadas, c) omite las celdas vacias que no se guarden, por lo que el resultado puede no ser rectangular, d) tampoco extrae formulas, ni formatos y e) todos los campos los retorna en formato de texto, asi, por ejemplo, las fechas se retornan como un numero en formato de texto y f) dependiendo de XlxCpm retorna todos los textos compactados o no, de forma independiente a las directrices del tipo xml:space = preserve. A partir de este codigo pueden programarse lectores de Xlsx mas avanzados. El unico fichero temporal que usa no se borra para facilitar la validacion. Hay 2 tipos de funciones: a) la que retorna directamente los contenidos y b) la que si el contenido de la celda no es un valor sino es el indice a la tabla de shared strings realiza la busqueda en esa tabla y retorna los contenidos reales. Para entender el uso de la tabla de sharedStrings en Excel Xlsx puede consultarse el sitio web: http://www.sadev.co.za/content en reading-and-writing-excel-2007-or-excel-2010-c-series-index

Declaraciones

Variables de control

- `Real XlxCmp`
  Si cierto compacta todos los textos siempre.
- `Real XlxTra`
  Si cierto traduce de Xml a Ascii los & < y >.

Constantes

- `Text XlxChr`
  Caracter separador que se espera unico.
- `Text XlxTmp`
  Fichero temporal que se espera unico.
- `Text Xlx7zp`
  Path al progama ejecutable 7 Zip.
- `Text XlxSha`
  Atributo shared string con dobles comillas.
- `Set XlxRep`
  Tabla de reemplazamientos Xml a Ascii.

Funciones

- `Set XlxSetCtr(Real newCmp, Real newTra)`

Memoriza los nuevos valores para los controles XlxCmp y XlxTra sobre la compactacion y traduccion de los textos leidos de ficheros Excel Xlsx, que se emplearan a partir de la llamada a esta funcion, esto es, actualiza los parametros de control. Retorna los valores anteriores por si se desean restaurar posteriormente.

- `Text XlxFromLblToEnd(Text txtXml, Text lblIni)`
  Retorna todo el texto a la derecha de la primera ocurrencia de la etiqueta lblIni en txtXml. Si la etiqueta lblIni no aparece retorna el texto vacio.

- `Text XlxFromTagToEnd(Text txtXml, Text tagIni)`
  Retorna todo el texto a la derecha de la primera ocurrencia del tag tagIni. Tiene en cuenta la forma de los tags Xml <tagIni . . .>. Es adecuada cuando no interesan los atributos del tag. Si tagIni no aparece o esta mal formado retorna el texto vacio.

- `Set XlxFromTagToAtrVal(Text txtXml, Text tagIni)`
  Retorna un par con el contenido de los atributos del tag y el valor hasta el final, por ejemplo, <c a=1>valor a la derecha -> [a=1, valor a la derecha]. Es adecuada cuando si interesan los atributos del tag. Si tagIni no aparece o esta mal formado retorna textos vacios.

- `Set XlxSplitByTag(Text txtXml, Text tagBrk)`
  Retorna el conjunto de todos los textos entre <tagBrk> y </tagBrk>. Es adecuada cuando no interesan los atributos del tag. XlxSplitByTag(a<v>1</v>b<v>2</v>c<v>3</v>d, v)-> [1,2,3] pero no los textos a b c d.

- `Set XlxSplitByTagAtrVal(Text txtXml, Text tagBrk)`
  Retorna el conjunto de pares formados por (todos los atributos, valor del contenido) de un tag XML. Siendo: a) los atributos del tag <tagBrk...a1=1 a2=2...aN=n> y b) el texto entre <tagBrk...> y </tagBrk> el valor de su contenido. Es adecuada cuando si interesan los atributos del tag.

- `Real Xlx7ZipExtract(Text inpPth, Text sheIde, Text outPth)`
  Retorna cierto si puede extraer la hoja numero sheIde del fichero Excel Xlsx de camino inpPth y guardarla en el fichero de salida de camino outPth. Para los casos sencillos sheIde puede ser un numero en forma de texto, por ejemplo si es un 2 se asume xl/worksheets/sheet2.xml. Si sheIde, no es un numero y empieza por xl, entonces se asume que se esta proporcionando el path completo de la hoja, por ejemplo, xl/externalLinks/externalLink1.xml, que en Excels dinamicos pueden contener la informacion que no contienen las hojas visibles.

- `Real Xlx7ZipTest(Text inpPth)`
  Retorna cierto si el fichero inpPth tiene una estructura PK correcta.

- `Real XlxIsExcelXlsx(Text inpPth)`
  Retorna cierto si inpPth tiene la extension y estructura PK correctas.

- `Text XlxReadText(Text inpPth, Text sheIde)`
  Retorna el texto Xml del Excel Xlsx de camino inpPth de su hoja sheIde.

- `Set XlxReadTable(Text inpPth, Text sheIde)`

Retorna una tabla, como conjunto de filas de conjuntos de celdas, con el contenido en texto del Excel Xlsx de camino inpPth de su hoja sheIde. Se trata de una version simple que puede retornar tablas no extrictamente rectangulares si hay celdas sin contenido o celdas combinadas. En caso de error retorna el conjunto vacio. Usa un metodo simple retornando el contenido de las celdas sin tener en cuenta la posible existencia de shared strings.

- `Set XlxReadShared(Text inpPth)`
  Retorna una lista como un conjunto lineal de los shared strings de un fichero Excel Xlsx. Esta lista de textos compartido se guarda en el Xml xl/sharedStrings.xml con el formato: <sst...> <si><t>Estos son los strings compartidos y</t></si> <si><t>enumerados del 0 al n</t></si> <si><t>aunque en tol del 1 al n+1</t></si> </sst> En caso de error esta funcion retorna el conjunto vacio.

- `Set XlxReadTableShared(Text inpPth, Text sheIde)`
  Retorna una tabla, como conjunto de filas de conjuntos de celdas, con el contenido en texto del Excel Xlsx de camino inpPth de su hoja sheIde. Se trata de una version simple que puede retornar tablas no extrictamente rectangulares si hay celdas sin contenido o celdas combinadas. A diferencia de XlxReadTable() lee previamente la tabla de shared string y comprueba en cada celda si el contenido es el original o si es un indice a la tabla de textos compartidos. Estos Xml tienen el formato: <c r='A1'><v>Soy un contenido original pero la celda C1 no lo tiene</v></c> <c r='B1'><v>Lo marca el atributo t=s y hay que usar el indice 652</v></c> <c r='C1' t='s'><v>652</v></c> En caso de error retorna el conjunto vacio.

Variables de control

Real XlxCmp

```
////////////////////////////////////////////////////////////////////////
Real XlxCmp = TRUE;
////////////////////////////////////////////////////////////////////////
PutDescription("Si cierto compacta todos los textos siempre.",
XlxCmp);
////////////////////////////////////////////////////////////////////////
```

Real XlxTra

```
////////////////////////////////////////////////////////////////////////
Real XlxTra = TRUE;
////////////////////////////////////////////////////////////////////////
PutDescription("Si cierto traduce de Xml a Ascii los & < y >.",
XlxTra);
////////////////////////////////////////////////////////////////////////
```

Constantes

Text XlxChr

```
////////////////////////////////////////////////////////////////////////
Text XlxChr = Char(7);
////////////////////////////////////////////////////////////////////////
PutDescription("Caracter separador que se espera unico.",
XlxChr);
```

```
////////////////////////////////////////////////////////////////////////
```

Text XlxTmp

```
////////////////////////////////////////////////////////////////////////
Text XlxTmp = "tmp/_xlsx_.tmp";
////////////////////////////////////////////////////////////////////////
PutDescription("Fichero temporal que se espera unico.",
XlxTmp);
////////////////////////////////////////////////////////////////////////
```

Text Xlx7zp

```
////////////////////////////////////////////////////////////////////////
Text Xlx7zp = "bin/7zip/7z.exe";
////////////////////////////////////////////////////////////////////////
PutDescription("Path al progama ejecutable 7 Zip.",
Xlx7zp);
////////////////////////////////////////////////////////////////////////
```

Text XlxSha

```
////////////////////////////////////////////////////////////////////////
Text XlxSha = "t=\"s\"";
////////////////////////////////////////////////////////////////////////
PutDescription("Atributo shared string con dobles comillas.",
XlxSha);
////////////////////////////////////////////////////////////////////////
```

Set XlxRep

```
////////////////////////////////////////////////////////////////////////
Set  XlxRep = [[ [["&amp;","&"]],
                 [["&lt",  "<"]],
                 [["&gt;", ">"]] ]];
////////////////////////////////////////////////////////////////////////
PutDescription("Tabla de reemplazamientos Xml a Ascii.",
XlxRep);
////////////////////////////////////////////////////////////////////////
```

Set XlxSetCtr()

```
////////////////////////////////////////////////////////////////////////
Set  XlxSetCtr(Real newCmp, // Nuevo valor para el control de compactacion
               Real newTra) // Nuevo valor para el control de traduccion
////////////////////////////////////////////////////////////////////////
{
  Set  oldVal = SetOfText(Copy(XlxCmp), Copy(XlxTra)); // Valores actuales
  Text(XlxCmp:=Copy(newCmp)); // Memoriza la nueva compactacion
  Text(XlxTra:=Copy(newTra)); // Memoriza la nueva traduccion
  oldVal // Retorna los antiguos valores
};
////////////////////////////////////////////////////////////////////////
PutDescription(
"Memoriza los nuevos valores para los controles XlxCmp y XlxTra sobre
la compactacion y traduccion de los textos leidos de ficheros Excel Xlsx,
que se emplearan a partir de la llamada a esta funcion, esto es, actualiza
los parametros de control.
Retorna los valores anteriores por si se desean restaurar posteriormente.",
```

```
XlxSetCtr);
////////////////////////////////////////////////////////////////////////////
```

Text XlxFromLblToEnd()

```
////////////////////////////////////////////////////////////////////////////
Text XlxFromLblToEnd(Text txtXml, // Original Xml text
                     Text lblIni) // Initial label
////////////////////////////////////////////////////////////////////////////
{
  Text xml2Asc(Text xmlTxt) { ReplaceTable(xmlTxt, XlxRep) }; // Cambia & < >

  Real posIni = TextFind(txtXml, lblIni);
  If(LE(posIni,0), "",
  {
    Real posSub = posIni + TextLength(lblIni);
    Real posEnd = TextLength(txtXml);
    Text subTxt = Sub(txtXml,posSub,posEnd);
    Text subCmp = If(XlxCmp, Compact(subTxt), subTxt); // Compacta si procede
    Text subTra = If(XlxTra, xml2Asc(subCmp), subCmp); // Traduce si procede
    subTra
  })
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna todo el texto a la derecha de la primera ocurrencia de la etiqueta
lblIni en txtXml.
Si la etiqueta lblIni no aparece retorna el texto vacio.",
XlxFromLblToEnd);
////////////////////////////////////////////////////////////////////////////
```

Text XlxFromTagToEnd()

```
////////////////////////////////////////////////////////////////////////////
Text XlxFromTagToEnd(Text txtXml, // Original Xml text
                     Text tagIni) // Initial tag without < >
////////////////////////////////////////////////////////////////////////////
{ XlxFromLblToEnd(XlxFromLblToEnd(txtXml, "<"+tagIni), ">") };
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna todo el texto a la derecha de la primera ocurrencia del tag tagIni.
Tiene en cuenta la forma de los tags Xml <tagIni . . .>.
Es adecuada cuando no interesan los atributos del tag.
Si tagIni no aparece o esta mal formado retorna el texto vacio.",
XlxFromTagToEnd);
////////////////////////////////////////////////////////////////////////////
```

Set XlxFromTagToAtrVal()

```
////////////////////////////////////////////////////////////////////////////
Set XlxFromTagToAtrVal(Text txtXml, // Original Xml text
                       Text tagIni) // Initial tag without < >
////////////////////////////////////////////////////////////////////////////
{
  Text atrVal = XlxFromLblToEnd(txtXml, "<"+tagIni);
  Text atrTxt = Sub(atrVal, 1, TextFind(atrVal, ">")); // De > a la izquierda
  Text valTxt = XlxFromLblToEnd(atrVal, ">");          // De > a la derecha
  SetOfText(atrTxt, valTxt)
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna un par con el contenido de los atributos del tag y el valor hasta el
final, por ejemplo, <c a=1>valor a la derecha -> [a=1, valor a la derecha].
Es adecuada cuando si interesan los atributos del tag.
Si tagIni no aparece o esta mal formado retorna textos vacios.",
XlxFromTagToAtrVal);
```

```
////////////////////////////////////////////////////////////////////////////
```

Set XlxSplitByTag()

```
////////////////////////////////////////////////////////////////////////////
Set XlxSplitByTag(Text txtXml, // Original Xml text
                  Text tagBrk) // Tag for break, without < </ >
////////////////////////////////////////////////////////////////////////////
{
  Set tokSet = Tokenizer(Replace(txtXml, "</"+tagBrk+">", XlxChr), XlxChr);
  For(1, Card(tokSet)-1, Text(Real posSet)
      { XlxFromTagToEnd(tokSet[posSet], tagBrk) })
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna el conjunto de todos los textos entre <tagBrk> y </tagBrk>.
Es adecuada cuando no interesan los atributos del tag.
  XlxSplitByTag(a<v>1</v>b<v>2</v>c<v>3</v>d, v)-> [1,2,3]
pero no los textos a b c d.",
XlxSplitByTag);
////////////////////////////////////////////////////////////////////////////
```

Set XlxSplitByTagAtrVal()

```
////////////////////////////////////////////////////////////////////////////
Set XlxSplitByTagAtrVal(Text txtXml, // Original Xml text
                        Text tagBrk) // Tag for break, without < </ >
////////////////////////////////////////////////////////////////////////////
{
  Set tokSet = Tokenizer(Replace(txtXml, "</"+tagBrk+">", XlxChr), XlxChr);
  For(1, Card(tokSet)-1, Set(Real posSet)
      { XlxFromTagToAtrVal(tokSet[posSet], tagBrk) })
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna el conjunto de pares formados por (todos los atributos,
valor del contenido) de un tag XML.
Siendo:
a) los atributos del tag <tagBrk...a1=1 a2=2...aN=n> y
b) el texto entre <tagBrk...> y </tagBrk> el valor de su contenido.
Es adecuada cuando si interesan los atributos del tag.",
XlxSplitByTagAtrVal);
////////////////////////////////////////////////////////////////////////////
```

Real Xlx7ZipExtract()

```
////////////////////////////////////////////////////////////////////////////
Real Xlx7ZipExtract(Text inpPth, // Input path, the Xlsx Excel file path
                    Text sheIde, // Sheet number 1, 2,... or sheet path
                    Text outPth) // Output path, the Xml output file
////////////////////////////////////////////////////////////////////////////
{
  Text xmlPth = If(TextBeginWith(sheIde, "xl"), sheIde, // Dan path completo
                "xl/worksheets/sheet"+sheIde+".xml");

  Text cmdTxt = Xlx7zp + " e " + Q(inpPth) + " " + xmlPth +  // Extrae de
                " -so > " + outPth;                          // Vuelca en
  Text cmdDos = W(cmdTxt);
  Text WriteLn(cmdDos);
  System(cmdDos)
};
////////////////////////////////////////////////////////////////////////////
```

```
PutDescription(
"Retorna cierto si puede extraer la hoja numero sheIde del fichero Excel Xlsx
de camino inpPth y guardarla en el fichero de salida de camino outPth.
Para los casos sencillos sheIde puede ser un numero en forma de texto,
por ejemplo si es un 2 se asume xl/worksheets/sheet2.xml.
Si sheIde, no es un numero y empieza por xl, entonces se asume que se esta
proporcionando el path completo de la hoja, por ejemplo,
xl/externalLinks/externalLink1.xml, que en Excels dinamicos pueden contener
la informacion que no contienen las hojas visibles.",
Xlx7ZipExtract);
////////////////////////////////////////////////////////////////////////////
```

Real Xlx7ZipTest()

```
////////////////////////////////////////////////////////////////////////////
Real Xlx7ZipTest(Text inpPth) // Input path, the Xlsx Excel file path
////////////////////////////////////////////////////////////////////////////
{
  Text cmdTxt = Xlx7zp + " t " + Q(inpPth) + " > " + XlxTmp;
  Text cmdDos = W(cmdTxt);

  Real Show(FALSE, "ERROR"); // No mostrar errores durante el test
  Real cmdExe = System(cmdDos);
  Real Show(TRUE,  "ERROR"); // Volver a mostrar errores

  GT(TextFind(ReadFile(XlxTmp), "Everything is Ok"),0)
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna cierto si el fichero inpPth tiene una estructura PK correcta.",
Xlx7ZipTest);
////////////////////////////////////////////////////////////////////////////
```

Real XlxIsExcelXlsx()

```
////////////////////////////////////////////////////////////////////////////
Real XlxIsExcelXlsx(Text inpPth) // Input path, the Xlsx Excel file path
////////////////////////////////////////////////////////////////////////////
{ If(TextEndAt(ToLower(inpPth), ".xlsx"), Xlx7ZipTest(inpPth), FALSE) };
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna cierto si inpPth tiene la extension y estructura PK correctas.",
XlxIsExcelXlsx);
////////////////////////////////////////////////////////////////////////////
```

Text XlxReadText()

```
////////////////////////////////////////////////////////////////////////////
Text XlxReadText(Text inpPth, // Input path, the Xlsx Excel file path
                 Text sheIde) // Sheet number 1, 2,... or sheet path
////////////////////////////////////////////////////////////////////////////
{ If(Xlx7ZipExtract(inpPth, sheIde, XlxTmp), ReadFile(XlxTmp), "") };
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna el texto Xml del Excel Xlsx de camino inpPth de su hoja sheIde.",
XlxReadText);
////////////////////////////////////////////////////////////////////////////
```

Set XlxReadTable()

```
////////////////////////////////////////////////////////////////////////////
Set XlxReadTable(Text inpPth, // Input path, the Xlsx Excel file path
```

```
                  Text sheIde) // Sheet number 1, 2,... or sheet path
////////////////////////////////////////////////////////////////////////////
{
  Text allXml = XlxReadText(inpPth, sheIde);
  Set  sheXml = XlxSplitByTag(allXml, "sheetData");
  If(NE(Card(sheXml), 1), Empty, // Solo puede haber 1
  {
    Text datXml = sheXml[1]; // El contetido con datos
    Set  rowSet = XlxSplitByTag(datXml, "row"); // El conjunto de filas
    EvalSet(rowSet, Set(Text rowXml) { XlxSplitByTag(rowXml, "v") })
  })
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna una tabla, como conjunto de filas de conjuntos de celdas, con el
contenido en texto del Excel Xlsx de camino inpPth de su hoja sheIde.
Se trata de una version simple que puede retornar tablas no extrictamente
rectangulares si hay celdas sin contenido o celdas combinadas.
En caso de error retorna el conjunto vacio.
Usa un metodo simple retornando el contenido de las celdas sin tener en
cuenta la posible existencia de shared strings.",
XlxReadTable);
////////////////////////////////////////////////////////////////////////////
```

Set XlxReadShared()

```
////////////////////////////////////////////////////////////////////////////
Set XlxReadShared(Text inpPth) // Input path, the Xlsx Excel file path
////////////////////////////////////////////////////////////////////////////
{
  Text allXml = XlxReadText(inpPth, "xl/sharedStrings.xml");
  Set  shaXml = XlxSplitByTag(allXml, "sst");
  If(NE(Card(shaXml), 1), Empty, // Solo puede haber 1
  {
    Text datXml = shaXml[1];   // El contetido con shared strings
    XlxSplitByTag(datXml, "t") // Lo correcto seria 1º por <si> y 2º por <t>
  })
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna una lista como un conjunto lineal de los shared strings de un fichero
Excel Xlsx.
Esta lista de textos compartido se guarda en el Xml xl/sharedStrings.xml
con el formato:
  <sst...>
    <si><t>Estos son los strings compartidos y</t></si>
    <si><t>enumerados del 0 al n</t></si>
    <si><t>aunque en tol del 1 al n+1</t></si>
  </sst>
En caso de error esta funcion retorna el conjunto vacio.",
XlxReadShared);
////////////////////////////////////////////////////////////////////////////
```

Set XlxReadTableShared()

```
////////////////////////////////////////////////////////////////////////////
Set XlxReadTableShared(Text inpPth, // Input path, the Xlsx Excel file path
                       Text sheIde) // Sheet number 1, 2,... or sheet path
////////////////////////////////////////////////////////////////////////////
{
  Set  shaStr = XlxReadShared(inpPth);       // Lista de shared strings
  Real shaLen = Card(shaStr);                // Numero de shared strings

  Text shaGet(Text posTxt) // Numero de posicion como texto
  {
    Real posNum = Eval(posTxt+"+1; "); // En tol empieza en 1 no en cero
    If(posNum > shaLen, "?", shaStr[posNum]) // Retorna ? si se pasa
```

```
  };

  Text allXml = XlxReadText(inpPth, sheIde); // Contenido Excel Xmlx
  Set  sheXml = XlxSplitByTag(allXml, "sheetData");
  If(NE(Card(sheXml), 1), Empty, // Solo puede haber 1
  {
    Text datXml = sheXml[1]; // El contetido con datos
    Set  rowSet = XlxSplitByTag(datXml, "row"); // El conjunto de filas
    EvalSet(rowSet, Set(Text rowXml)
    {
      Set  celTab = XlxSplitByTagAtrVal(rowXml, "c"); // Pares atributo valor
      EvalSet(celTab, Text(Set atrVal) // Para todo par atributo valor
      {
        Real atrSha = TextFind(atrVal[1], XlxSha); // Cierto si es shared
        Text valTxt = XlxSplitByTag(atrVal[2], "v")[1]; // Solo hay 1 valor
        If(atrSha, shaGet(valTxt), valTxt) // Shared->buscar, sino es el mismo
      })
    })
  })
};
////////////////////////////////////////////////////////////////////////////
PutDescription(
"Retorna una tabla, como conjunto de filas de conjuntos de celdas, con el
contenido en texto del Excel Xlsx de camino inpPth de su hoja sheIde.
Se trata de una version simple que puede retornar tablas no extrictamente
rectangulares si hay celdas sin contenido o celdas combinadas.
A diferencia de XlxReadTable() lee previamente la tabla de shared string y
comprueba en cada celda si el contenido es el original o si es un indice a
la tabla de textos compartidos. Estos Xml tienen el formato:
  <c r='A1'><v>Soy un contenido original pero la celda C1 no lo tiene</v></c>
  <c r='B1'><v>Lo marca el atributo t=s y hay que usar el indice 652</v></c>
  <c r='C1' t='s'><v>652</v></c>
En caso de error retorna el conjunto vacio.",
XlxReadTableShared);
////////////////////////////////////////////////////////////////////////////
```

inc.tol de Xlsx.Reader

Inclusion de ficheros comunes

Declaraciones

Inclusiones comunes

- `Set txtInc`

  Text functions.

- `Set xlxInc`

  Funciones de lectura de Excel Xlsx.

Set txtInc

```
////////////////////////////////////////////////////////////////////////////
Set  txtInc = Include("cmm/txt.tol");
////////////////////////////////////////////////////////////////////////////
PutDescription("Text functions.", txtInc);
////////////////////////////////////////////////////////////////////////////
```

Set xlxInc

```
////////////////////////////////////////////////////////////////////////////
Set  xlxInc = Include("cmm/xlx.tol");
////////////////////////////////////////////////////////////////////////////
PutDescription("Funciones de lectura de Excel Xlsx.", xlxInc);
////////////////////////////////////////////////////////////////////////////
```