

Tcluno y Itcluno: Extensiones para automatizar OpenOffice

TOMANDO EL CONTROL

OpenOffice viene de serie con su propio dialecto Basic, un potente lenguaje de macros. Pero no Basic es del gusto de todo el mundo. Los que prefieren Tcl pueden usar este lenguaje para controlar su Office. Con Tcluno e Itcluno se obtienen informes listos para imprimir y tablas con datos a partir de cualquier tipo de fuente.

POR CARSTEN ZERBST

Para muchos usuarios el procesador de textos y la hoja de cálculo pertenecen a los programas más importantes del escritorio. Pero a veces es preciso generar documentos automáticamente o retocar partes controladas por programas. En este caso sería útil poder utilizar el lenguaje script con el que se trabaja habitualmente en vez de usar el Basic integrado.

Office desde la distancia

Gracias a Arnulf Wiedman y Wolfgang Großer disponemos de una extensión de Tcl, Tcluno [1], que permite a los scripts un acceso pleno a todas las funcionalidades de OpenOffice para generar y cambiar documentos. El potente modelo de componentes de UNO (véase el recuadro "Universal Network Objects") dispone de las funcionalidades necesarias para controlar OpenOffice a distancia desde otros programas. El resultado es un documento OpenOffice que se puede exportar también a Winword/Excel, PDF o bien imprimirse.

Las fuentes de Tcluno se encuentran en Sourceforge [2]. Los desarrolladores trabajan bastante en su proyecto; mientras

que las primeras versiones se basaban en las bibliotecas C++ de OpenOffice, la versión actual es una extensión del script que no requiere compilación. Por ello es suficiente depositar la extensión en cualquier parte del ordenador y cambiar la ruta de búsqueda de Tcl respectivamente. En el servidor de Linux Magazine [3] se encuentran un paquete preparado con los ejemplos de este artículo y una versión probada de la extensión.

Procesos enlazados

La comunicación entre OpenOffice y Tcluno se realiza a través de la red, incluso si ambos procesos corren en la misma máquina. OpenOffice trabaja entonces como servidor. El programa tiene que estar arrancado para que Tcl pueda consultarlo. Desgraciadamente OpenOffice arranca por defecto sin soporte de red. Una llamada con la opción `accept` lo cambia:

```
ooffice "-accept=socket,
host=localhost, port=2002;urp;"
```

Sólo los clientes (o sea scripts Tcl) que corren en la misma máquina obtienen

acceso a OpenOffice con estas propiedades. Quien necesite esta funcionalidad más a menudo, puede registrar el soporte de red de manera permanente en el archivo de configuración. Para esto hay que añadir tres líneas en `Setup.xcu` en la rama XML `node oor:name="Office"`, son las líneas 9 a 11 con la propiedad `ooSetupConnectionURL` en el listado 1.

Acceso permitido

Según la versión, distribución e instalación usadas, el archivo de configuración se encuentra en sitios distintos. En OpenOffice 2.0 el archivo de configuración específico del usuario está en `~/.openoffice.org2/user/registry/data/org/openoffice/Setup.xcu`, el archivo de configuración de todo el sistema, por ejemplo, en `/usr/lib/ooo-2.0/share/registry/data/org/openoffice/Setup.xcu`. En la versión más antigua 1.1 el archivo específico del usuario se llama `~/OpenOffice.org1.1/user/registry/data/org/openoffice/Setup.xcu`, para todo el sistema `/usr/lib/ooo-1.1/share/registry/data/org/openoffice/Setup.xcu`.

Universal Network Objects

Cuando la empresa Star Division empezó a desarrollar el Star Office a mediados de los noventa, eligió a C++ como lenguaje de programación. Pero le faltó un modelo de componentes para dividir las grandes aplicaciones en trozos pequeños independientes de la plataforma. Entonces Star Division desarrolló su modelo de componentes UNO siguiendo el ejemplo de las ideas del OLE (Object Linking and Embedding) y del OMG CORBA (Object Management Group, Common Object Request Broker [4]) de Microsoft. UNO significa Universal Network Objects (esto es, Objetos de Red Universal).

Descripción de interfaces

UNO usa un lenguaje propio (parecido a CORBA) para describir métodos y propiedades de las clases disponibles. Comparable con OLE de Microsoft, los objetos soportan varias interfaces, un cliente las consulta en tiempo de ejecución y se decide por una de ellas. La implementación para una hoja de cálculo ("SpreadsheetImplementation" Figura 1) conoce por ejemplo varias interfaces. Un programa de Java las consultaría de la siguiente manera:

```
XStorable xStorable =
(XStorable)
    UnoRuntime.queryInterface(
        XStorable.class, spreadsheet);
xStorable.storeAsUrl( ... );
```

```
XPrintable xPrintable =
(XPrintable)
    UnoRuntime.queryInterface(
        XPrintable.class, spreadsheet);
xPrintable.print( ... );
```

La ventaja del modelo de componentes es que el desarrollador puede elegir libremente durante la implementación las interfaces a disposición de un nuevo objeto. El principio se parece a las interfaces de Java. Al contrario de C++, evita las herencias múltiples. Si se cambia la implementación de un objeto, no tiene consecuencias

para el resto del código fuente, porque éste solo usa las interfaces disponibles y no el objeto en sí.

La agrupación proporciona una vista general

Para no perder la vista general con tantas interfaces, es posible agruparlas en unidades más grandes, llamadas "servicios" (servicios). En UNO un simple constructor no es suficiente para generar un objeto, el programa más bien tiene que solicitarlo al "Servicemanager" (o Administrador de servicios, también llamado "factory" o fábrica). Es curioso que la empresa francesa Dassault Systèmes use un entorno casi idéntico en Catia V5. Desgraciadamente la versión Linux de este sistema CAD no está disponible todavía.

Tal y como sugiere la palabra "Network" de las siglas UNO, el acceso a las interfaces de los componentes funciona también a través de la red. Éste es el camino más sencillo para controlar OpenOffice desde lenguajes de programación ajenos. Quien quiera escribir sus propios objetos UNO, tiene que usar actualmente C++ o Java. Quien sólo quiera usar las funcionalidades existentes de OpenOffice, puede usar también otros lenguajes como Tcl, Python, Dotnet o OpenOffice Basic.

Ojeada en tiempo de ejecución

Especialmente práctico para lenguajes script: se pueden consultar las interfaces disponibles en tiempo real a través de la introspección. Tcuno aplica y solicita por sí mismo las interfaces necesarias. Los propios programadores de Java o C++ tienen que afanarse en este paso antes de poder aplicar métodos a un objeto. Si los programadores de Tcl conocen las interfaces de una implementación, entonces pueden aplicarlas directamente.

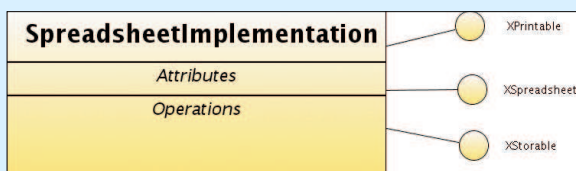


Figura 1: Gracias a UNO la implementación y la interfaz están separadas, un objeto suele soportar varias interfaces.

Puro UNO

Los objetos OpenOffice funcionan para programadores de Tcl igual que los de Tk o Itcl. En Tcl están listos como objetos, según la tradición de Tk el script pasa los

métodos definidos en la descripción del interfaz al objeto. El listado 2 sólo usa una de las 2600 clases de OpenOffice para llenar una hoja de cálculo y exportarla en formato Excel.

Listado 1: Soporte de red

```
01 <?xml version="1.0"
02 encoding="UTF-8"?>
03 <oor:component-data
04 xmlns:oor="http://openoffice.o
05 rg/2001/registry"
06 xmlns:xs="http://www.w3.org/20
07 01/XMLSchema" oor:name="Setup"
08 oor:package="org.openoffice">
09 <node oor:name="L10N">
10 <prop oor:name="ooLocale"
11 oor:type="xs:string">
12 <value>en-US</value>
13 </prop>
14 </node>
15 <node oor:name="Office">
16 <prop
17 oor:name="ooSetupConnectionURL"
18 oor:type="xs:string">
19 <value>socket,host=localhost,p
20 ort=2002;urp</value>
21 </prop>
22 <prop
23 oor:name="ooSetupInstCompleted"
24 oor:type="xs:boolean">
25 <value>1</value>
26 </prop>
27 </node>
28 </oor:component-data>
```

Los comandos *lappend* en las líneas 3 y 4 amplían la ruta de búsqueda de Tcl con la instalación de TcLurtp y TcLuno. Para la comunicación con OpenOffice, los ejemplos trabajan con una solución de script TcLurtp pura. Pero en la versión actual TcLuno usa la variante TcLurtp solamente si el script corre con la opción *-urtp*. La línea 7 procura que la variable *::argv* contenga siempre esta cadena, independientemente de lo que el usuario indicó en la línea de comandos.

A continuación el código empieza a solicitar la extensión para la hoja de cálculo. La línea 12 intenta establecer el contacto con OpenOffice con *::tcluno_soffice::initDesktop*. Si esto falla, las siguientes llamadas *puts stderr* generan un amplio mensaje de error. En caso de éxito la variable *desktop* contiene el objeto de arranque central.

Por cierto, en TcLuno el programador no tiene que preocuparse por consultar la referencia de objeto solicitada antes de poder usarla (como sucede en Java o C++). No obstante estas consultas son posibles y dan sentido para informarse sobre las interfaces soportadas por un objeto. Las líneas 23 hasta 26 lo muestran; el resultado se puede ver en la figura 2.

Tipos de datos

La mayoría de los comandos UNO esperan entradas simples como cadenas, números o listas. Pero hay también unos tipos de datos especiales, que no tienen equivalentes en Tcl. TcLuno lo compensa con unos comandos auxiliares, por ejemplo, para la secuencia tan usada del tipo Any (línea 29). Este contenedor genérico puede recibir cualquier tipo de datos.

El objeto *desktop* sirve de generador de documentos. El método *loadComponentFromURL* abre un documento (línea 32). El primer parámetro indica su URL. El URL normal es suficiente para documentos ya existentes (por ejemplo *file:/home/cz/test.odt*). Pero para cada tipo de archivo existe además un URL especial:

- private:factory/swriter
- private:factory/scalc
- private:factory/sdraw
- private:factory/simpress

Generan un documento nuevo y vacío. El segundo parámetro indica la ventana en la que aparece el documento. *_blank* proporciona una ventana nueva. Los últimos dos argumentos no se usan y son valores predeterminados.

El resultado de lo anterior es un objeto nuevo, cuya referencia termina en *\$spreadsheet*; se trata de un documento de hoja de cálculo. La línea 36 solicita la lista de todas las hojas de este documento, y la línea 37 elige de ellas la primera página. El título de esta hoja se llama por defecto *tabla1*. La línea 39 lo cambia a *Números_aleatorios*.

Navegación en el documento

OpenOffice conoce dos técnicas para dirigirse a las celdas de una hoja: bien como se puede ver en la línea 44, a través de *getCellByPosition* con el índice de la columna y la fila como número, o

bien como en la línea 51, con *getCellRangeByName* y la combinación común de letra y cifra. Cuidado: el índice para *getCellByPosition* empieza con 0, mientras que la primera fila en *getCellRangeByName* lleva el número 1.

Con *setValue* el programa pone los valores de las cifras en las celdas (línea 46). Además de valores numéricos, las celdas pueden contener también fórmulas o textos. Esto se consigue con *setFormula*, como se puede ver en las líneas 56 (texto) y 58 (fórmula). Hay que procurar que las fórmulas estén en forma inglesa, por ejemplo *=median(A1:A2)* (línea 62) y no *=mediana(A1:A2)*, que sólo está permitido desde el GUI.

Para guardar el documento terminado (figura 2b), el programa tiene que decidir entre todos los formatos soportados por OpenOffice, por ejemplo Excel, Star Calc o CSV (Comma Separated Values). Normalmente, para estas

Listado 2: TcLuno

```

01 #!/usr/bin/tclsh
02
03 lappend auto_path [file join
  [pwd] tclurtp]
04 lappend auto_path [file join
  [pwd] tcLuno ]
05
06 # Solicitar versión Tcl-Only
  de TcLuno
07 set ::argv "-urtp"
08 # Cargar extensión
09 package require tcLuno_scalc
10
11 # Establecer contacto con
  OpenOffice
12 if {[catch
  {::tcLuno_soffice::initDesktop
  } desktop]} {
13   puts stderr "Conexión con
  OpenOffice fallado!"
14   puts stderr "Error fue:"
15   puts stderr $desktop
16   puts stderr ""
17   puts stderr "Por favor,
  arrancar OpenOffice con
  soporte de red:"
18   puts stderr "ooffice
  \"-accept=socket,host=localhos
  t,port=2002;urp;"
19   exit 1
20 }
21
22 # Mostrar interfaces
23 puts ">desktop< tiene los
  siguientes interfaces:"
24 puts [join [$desktop getTypes]
  "\n"]
25 puts "\n>desktop< es el
  servicio siguiente:"
26 puts [$desktop
  getSupportedServiceNames]
27
28 # Crear lista vacía
29 set filterSequence [$desktop
  tcLuno::createUnoSequence Any]
30
31 # Nuevo documento en hoja de
  cálculo
32 set spreadsheet [$desktop
  loadComponentFromURL \
  "private:factory/scalc"
  "_blank" 0 $filterSequence]
33
34
35 # Encontrar primera página
36 set sheets [$spreadsheet
  getSheets]
37 set sheet [$sheets getByIndex
  0]
38 puts "\nLa tabla se llama:
  [$sheet getName]"
39 $sheet setName
  "Números_aleatorios"
40
41 # Llenar tabla con números
  aleatorios
42 for {set col 0} {$col < 10}
  {incr col} {
43   for {set row 0} {$row <
  10} {incr row} {
44     set cell [$sheet
  getCellByPosition $col $row]
45     set value [expr
  {rand() * 100}]
46     $cell setValue $value
47   }
48 }
49
50 # Dirección de la celda
51 set cell [$sheet
  getCellRangeByName "A12"]
52 puts "\nDirección completa:
  [$cell getCellAddress]"
53
54 # Fórmulas y textos
55 set cell [$sheet
  getCellRangeByName "A12"]
56 $cell setFormula "Suma:"
57 set cell [$sheet
  getCellRangeByName "B12"]
58 $cell setFormula
  "=sum(B1:B10)"
59 set cell [$sheet
  getCellRangeByName "A13"]
60 $cell setFormula "Mediana:"
61 set cell [$sheet
  getCellRangeByName "B13"]
62 $cell setFormula
  "=median(B1:B10)"
63
64 # Exportar como hoja de
  cálculo Excel
65 set filterSequence [::$desktop
  tcLuno::createUnoSequence Any]
66 set msExcelFilter [::$desktop
  tcLuno::createUnoStructHelper
  \
  com.sun.star.beans.PropertyVal
  ue \
  {FilterName -1 {MS Excel
  97} 0}]
67
68
69 ::desktop
  tcLuno::appendUnoSequence
  $filterSequence $msExcelFilter
70
71 $spreadsheet storeAsURL
  "file:[file normalize
  ~/prueba.xls]" $filterSequence

```

	A	B	C	D	E	F	G	H	I	J
1	29,05	52,49	49,11	87,74	8,27	65,05	80,71	86,17	91,25	17,37
2	59,85	57,64	8,85	95,69	21,68	64,8	20,16	90,34	86	41,31
3	27,13	20,3	87,59	72,29	44,55	48,33	70,56	44,88	26,9	22,91
4	84,32	51,83	45,73	53,2	14,59	88,74	9,33	3,87	32,4	92,85
5	97,57	25,6	27,78	14,79	73,9	34,64	84,09	0,68	30,91	84,35
6	5,24	48,43	94,3	36,71	88,33	12,9	86,94	85,23	24,48	46,2
7	9,9	22,71	82,01	58,45	17,81	44,93	18,79	96,02	63,38	45,87
8	41,47	10,36	66,75	16,91	60,27	23,48	27,26	10,55	84,76	73,79
9	3,22	96,69	15,62	51,95	40,02	37,7	87,18	32,52	27,77	42,98
10	54,85	99,72	22	95,35	35,24	45,87	73,19	18,35	90,01	19,57
11										
12	Suma:	485,78								
13	Mediana:	50,13								
14										
15										
16										

Figura 2: El script Tcl ha cambiado el nombre de la primera tabla. La ha llenado de números aleatorios y ha insertado el cálculo de la suma y el valor medio.

configuraciones se usa una secuencia Any. La línea 65 genera una lista vacía de este tipo genérico.

Para guardarlos como Excel, la lista tiene que contener una estructura del tipo *com.sun.star.beans.PropertyValue*. La función de ayuda *createUnoStructHelper* es la responsable de ello en Tcuno (línea 66 hasta 68). Espera como parámetros el nombre y el contenido de la estructura. La función de ayuda *appendUnoSequence* incorpora en la línea 69 la estructura nueva en la secuencia vacía, lo cual aprovecha la línea 71 para guardar el documento nuevo bajo el nombre *prueba.xls* en la carpeta home.

Más cómodo

Itcuno está aún mejor adaptado al mundo de componentes orientado a objetos de OpenOffice que Tcuno. Ahorra mucho trabajo minucioso al programador. Como el nombre indica, Itcuno se basa en la extensión de Tcl orientada a objetos Itcl [5], pero necesita también la extensión Tcivfs (Virtual File System o Sistema de Ficheros Virtual[6]). Ambos faltan en muchas distribuciones de Tcl. Active Tcl de Activestate [7] es una manera fácil de cosnequirlos.

Con Itcuno, un programa no usa los objetos UNO directamente como en el

listado 2, sino de manera indirecta a través de un surtido de clases Itcl, fáciles de usar. Actúan como la capa superior, que encapsula los objetos OpenOffice. Así un script consigue realizar su tarea con menos comandos. El ejemplo en el listado 3 genera un documento nuevo con Itcuno, lo llena de valores registrados y produce con ellos un diagrama.

Clases de hoja de cálculo

La clase *SpreadSheet* es la responsable de la hoja de cálculo. Sus aproximadamente 30 métodos se ocupan de casi todo lo relacionado con la hoja de cálculo. Sólo

Listado 3: Itcuno

```

01 #!/usr/bin/tclsh
02
03 lappend auto_path [file join
[pwd] tclurtp]
04 lappend auto_path [file join
[pwd] itcuno ]
05 lappend auto_path [file join
[pwd] itcuno]
06
07 # Solicitar la versión Tcl-Only
de Tcuno
08 set ::argv "-urtp"
09 # Cargar extensión
10 package require itcuno
11
12 # Generar documento
13 set spreadsheet
[::itcuno::SpreadSheet #auto]
14 $spreadsheet renameSheet Tabla1
Valores
15
16 # Poner valores
17 $spreadsheet setCellValue
Valores [list 0 0] \
18     {{Fecha Hora Altura}}
19
20 # Altura del mar en Cádiz,
23.3.2006
21 # Fuente: http://indamar.ieo.es/
(Mareas | Acceso a datos |
Cádiz)
22 set datos [list \
23 [list 21.03.06 00:00 13.25 ] \
24 [list 21.03.06 01:00 15.33 ] \
25 [list 21.03.06 02:00 18.80 ] \
26 [list 21.03.06 03:00 22.51 ] \
27 [list 21.03.06 04:00 26.31 ] \
28 [list 21.03.06 05:00 29.00 ] \
29 [list 21.03.06 06:00 29.57 ] \
30 [list 21.03.06 07:00 27.78 ] \
31 [list 21.03.06 08:00 24.66 ] \
32 [list 21.03.06 09:00 20.14 ] \
33 [list 21.03.06 10:00 16.33 ] \
34 [list 21.03.06 11:00 14.35 ] \
35 [list 21.03.06 12:00 14.24 ] \
36 [list 21.03.06 13:00 15.58 ] \
37 [list 21.03.06 14:00 18.40 ] \
38 [list 21.03.06 15:00 21.73 ] \
39 [list 21.03.06 16:00 25.16 ] \
40 [list 21.03.06 17:00 27.85 ] \
41 [list 21.03.06 18:00 28.89 ] \
42 [list 21.03.06 19:00 28.60 ] \
43 [list 21.03.06 20:00 25.95 ] \
44 [list 21.03.06 21:00 21.81 ] \
45 [list 21.03.06 22:00 17.77 ] \
46 [list 21.03.06 23:00 14.92 ] \
47 [list 21.03.06 24:00 13.75 ]]
48 $spreadsheet setCellValue
Valores [list 0 1] $datos
49
50 # Generar diagrama
51 $spreadsheet createChart Valores
\
52     [list 12 1 12 12 cm] \
53     [list hasRowHeader 1
hasColumnHeader 1 \
54         title "Altura del mar
en Cádiz" \
55         type LineDiagram] \
56     Valores [list 1 1 2 26]
57
58 tcluno_soffice::windowTitle \
59     [$spreadsheet cget -desktop]
\
60     "Altura del mar en Cádiz"
61
62 $spreadsheet exportToPdf
altura.pdf

```

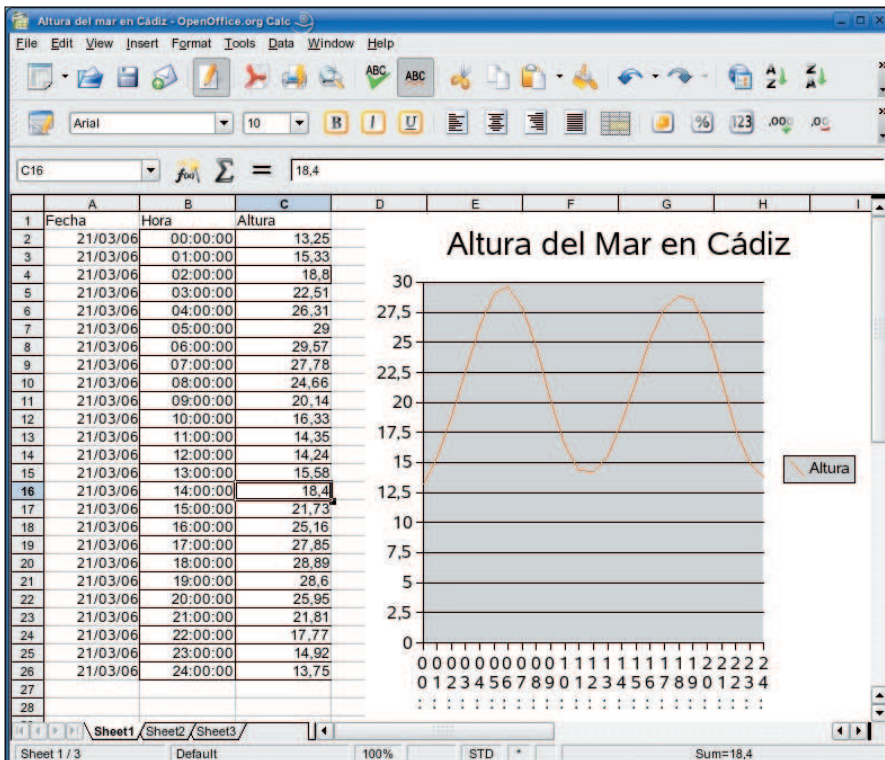


Figura 3: Sólo siete comandos `lcluno` bastan para generar esta tabla completa. En el listado 3 el programa genera una nueva hoja de cálculo, lo llena con los datos registrados de la marea y produce con ellos un diagrama de líneas.

con el constructor en la línea 13 se establece el contacto con OpenOffice y se genera un documento nuevo. La línea 14 da un nombre a la tabla, y la línea 17 pone los primeros valores en la tabla con `setCellValue`. El código es el siguiente:

```
$spreadsheet setCellValue
Nombre_de_tabla {ColumnaComienzo
FilaComienzo} {{Valor1 Valor12}
{Valor21 Valor22}}
```

Los valores en la lista embebida pueden ser números, textos o fórmulas. A los títulos de las columnas les siguen los datos, que esta vez provienen del Instituto Español de Oceanografía (RED DE INFORMACIÓN DE DATOS OCEANOGRÁFICOS) [8]. Las líneas 23 hasta 47 contienen directamente los valores; pero un script podría consultar los valores directamente desde una página web. La línea 48 introduce este bloque de datos en OpenOffice.

Un diagrama proporciona una mejor vista general de los cambios de la altura del mar: Las líneas 51 hasta 56 generan

Lo último

La combinación de base de datos y Tcl parece ser muy adecuada para el desarrollo rápido de aplicaciones. Como prueba, casi no hay una base de datos sin extensión de Tcl. Unas usan Tcl hasta como herramienta de prueba en el desarrollo. Jerry LeVan acaba de publicar su nueva versión 1.6 de PgBrowser [10]. Se trata de un navegador de bases de datos bastante potente para PostgreSQL.

La extensión Tile proporciona un mejor aspecto para las aplicaciones Tk, pero todavía no tiene tantas funcionalidades como Tk. Para una transición suave, los desarrolladores integran ahora Tile paso a paso al Tk normal. El nuevo kit de herramientas mezcladas se llama Ttk [11].

Websh [12] es un entorno avanzado para aplicaciones web. La versión 3.6.0b3 contiene parches para errores y es compatible con Apache 1.3 y 2.0 y con Tcl 8.4 y 8.5.

Miradas curiosas

Cuando hay que solucionar un problema con clientes o en máquinas ajenas, muchas veces sólo queda disponible el `korn-shell` o tal vez sólo `cmd.exe`. No es una buena base para resolver pequeños problemas y otros más grandes en un tiempo razonable.

Aquí los `Tclkits` [13] son una gran ayuda, porque contienen en un único archivo el intérprete completo de Tcl y no necesitan derechos de root. En combinación con la consola TK [14] desarrollada por Jeffrey Hobbs, se arranca un entorno de desarrollo práctico en cuestión de segundos. Esto funciona no solamente en plataformas Linux-X86, sino también en entornos menos dotados como Linux-S390, HP-UX, Solaris o OSF-Alpha.

En la caja de herramientas falta ahora sólo una aplicación que vigila cualquier comunicación TCP/IP de navegador o Telnet. Esto es lo que hace Sockspy (figura 4 y [15]) y cabe en un dispositivo de memoria USB para urgencias. Se pueden registrar las transferencias grabadas de manera cómoda y también mostrarlas como volcado hexadecimal. Sockspy no ofrece las posibilidades del líder Ethereal, pero en cambio no requiere derechos root.

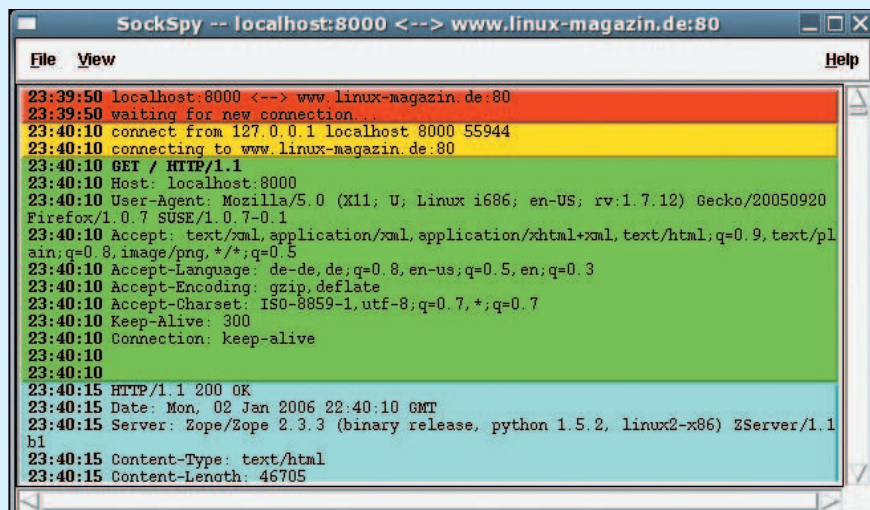


Figura 4: Sockspy se coloca entre cliente y servidor para escuchar las transferencias TCP/IP y mostrarlo en orden cronológico.

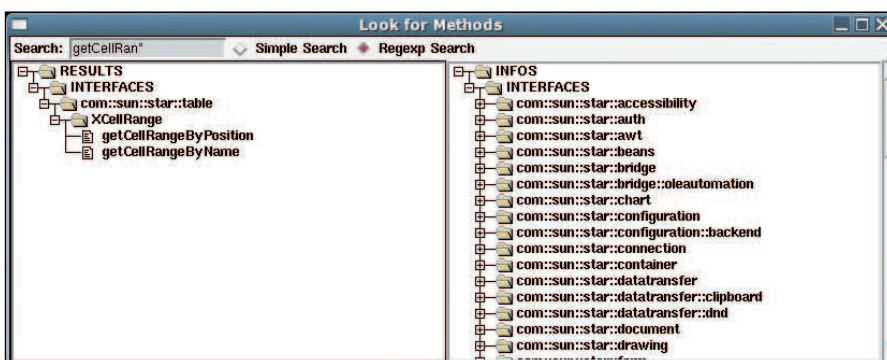


Figura 5a: Unospection muestra una estructura de árbol de las interfaces UNO de OpenOffice. El usuario puede ver todos los métodos para un objeto y llamarlos directamente.

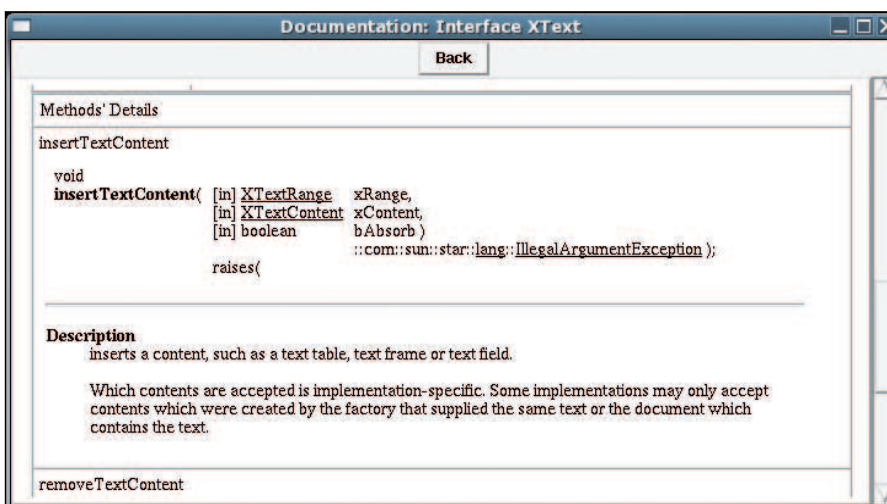


Figura 5b: Especialmente práctico: la herramienta lleva consigo la documentación completa del API de OpenOffice.

el diagrama de líneas (figura 3). La línea 58 pone el título y la 62 exporta el documento como PDF. Gracias a Itcluno todo ha costado sólo 7 comandos (líneas 13, 14, 17, 48, 51, 58 y 62). Itcluno incluye una amplia colección de ejemplos útiles, que abarcan desde tareas sencillas de formato hasta una extensa tabla de Pivot.

El programa completo

Tcluno proporciona a los scripts Tcl el API completo de OpenOffice. Los desarrolladores generan con ello documentos y modifican su contenido sin aprender un lenguaje nuevo. Los ejemplos de este artículo tratan solamente de generar hojas de cálculo, pero tanto Tcluno como Itcluno incluyen un procesador de textos, aplicaciones de diseño y presentaciones.

Itcluno lleva más rápido al éxito en muchas tareas, porque cubre las posibilidades esenciales con poco esfuerzo de aprendizaje. El comienzo,

en cambio, puede resultar bastante penoso, porque el API de OpenOffice es difícil de entender con sus aproximadamente 2600 clases y 6000 métodos. Por suerte suelen bastar 20 clases, pero hay que aprenderlos también.

Documentación

La documentación de OpenOffice proporciona un buen comienzo. Se puede encontrar en línea y en el kit de desarrollo de OpenOffice [9]. Además de los secciones de introducción, vale la pena leer los capítulos 7 hasta 10, que explican con muchos ejemplos el manejo de los distintos tipos de documentos. Los ejemplos están pensados para Java, pero quien recorre todas las líneas con *queryInterface*, puede usar el resto para Tcluno.

Navegación por la jungla API

Tcluno contiene también junto con Unospection (Figuras 5a y 5b) una

herramienta útil para probar scripts propios y para buscar en el API de OpenOffice. La herramienta es una mezcla entre navegador de objetos (Figura 5a) y visor de ayuda con documentación integrada (Figura 5b). Si lo pide, el desarrollador ve el texto de ayuda para todos los métodos viables para un objeto, y puede llamar el método directamente de prueba.

Unospection merece la pena ya sólo por su navegador de ayuda. Esta función se encuentra en el menú *Edit | Search Methods*. La herramienta contiene la documentación para todas las clases y todos los métodos. Unospection todavía tiene algunos errores por subsanar, pero tiene toda la base para ser un ayudante práctico para desarrollar scripts para OpenOffice. Los dos autores Wolfgang Großer y Arnulf Wiedman han hecho un buen trabajo, que se merece más que un vistazo para la automatización de Office. ■

RECURSOS

- [1] Página del proyecto Tcluno: <http://www.tugm.de/Projekte/TCLUNO/index.html/en>
- [2] Tcluno en Sourceforge: <https://sourceforge.net/projects/tcluno/>
- [3] Programas de este artículo: <http://www.linux-magazine.es/Magazine/Downloads/17>
- [4] OMG Corba: <http://es.wikipedia.org/wiki/Corba>
- [5] Incr Tcl: <http://incrtcl.sourceforge.net/itcl/>
- [6] Tclvfs: <http://sourceforge.net/projects/tclvfs>
- [7] Activestate Tcl: <http://www.activestate.com/Tcl.plex>
- [8] Instituto Español de Oceanografía, "Mareas | Acceso a datos | Cádiz": <http://indamar.ieo.es/>
- [9] Kit de desarrollo de software OpenOffice (SDK): http://www.openoffice.org/dev_docs/source/sdk/
- [10] PgBrowse: <http://homepage.mac.com/levanj/TclTk/>
- [11] Tile Tk: <http://wiki.tcl.tk/14796>
- [12] Websh: <http://tcl.apache.org/websh/>
- [13] Tclkit: <http://www.equi4.com/pub/tk/downloads.html>
- [14] Tkcon: <http://tkcon.sourceforge.net>
- [15] Sockspy: <http://sockspy.sourceforge.net/sockspy.html>