Programmare OpenOffice.org[1]

Come estendere le funzionalità della suite di applicazioni, con particolare riferimento alla scrittura di macro in OpenOffice.org Basic

di Ivan Venuti (ivanvenuti@yahoo.it)

Alcune applicazioni mettono a disposizione degli strumenti per la loro personalizzazione e per estenderne le funzionalità. In quest'articolo vedremo cosa offre OpenOffice.org (sito di riferimento http://www.openoffice.org); si tenga presente che gran parte degli argomenti trattati sono applicabili alla versione commerciale del prodotto Sun: StarOffice.
OpenOffice.org permette di aggiungere componenti utilizzando una apposita Application Programming Interface (API) e usare vari linguaggi di programmazione per la creazione di nuovi componenti.
D'altro canto chi vuole automatizzare semplici operazioni può ricorrere anche alla creazione di Macro in un dialetto del Basic: *OpenOffice.org Basic*. Tali macro saranno associate allo specifico documento/template e interpretate al run-time. È possibile anche creare macro associate all'installazione locale di una specifica applicazione, in modo che siano sempre disponibili per tutti i documenti.

Sviluppo di componenti

Chi volesse sviluppare componenti deve eseguire il download del kit di sviluppo (UDK, UNO Development Kit) dalla pagina Web http://udk.openoffice.org/. L'architettura software che permette lo sviluppo di nuovi componenti è UNO (Universal Network Objects: lo si può pensare come il modello corrispettivo a COM in ambiente Microsoft).
Ciascun componente deve essere sviluppato a parte in un apposito linguaggio (tipicamente C++ o Java, ma in generale è possibile sviluppare componenti in qualsiasi linguaggio di programmazione per cui esista un bridge per l'architettura UNO).
Una volta che il componente è stato sviluppato, è necessario che l'utente lo installi nella propria installazione di OpenOffice.org per poterne fare uso.
Il vantaggio dei componenti è che diventano a tutti gli effetti parti della applicazione, andando ad integrarsi e ad estendere le funzionalità esitenti.

Sviluppo di macro

È possibile anche utilizzare un IDE (Integrated Development Editor) per lo sviluppo di macro in OpenOffice.org Basic. Esse, una volta scritte, vengono interpretate ed eseguite al runtime.
Per richiamare l'editor (per esempio in Writer) è sufficiente accedere al menu “Tools > Macro” e quindi editare una macro esistente o crearne una nuova.
Lo sviluppo di macro è sicuramente più semplice dello sviluppo di componenti e richiede una conoscenza meno approfondita del modello sottostante.
La scelta del Basic è stata fatta per venire incontro a quanti sono abituati a usare VBA nella suite Microsoft Office. Le macro scritte per Microsoft Office non sono eseguibili direttamente in OpenOffice.org, perchè si basano su architetture diverse e perchè le applicazioni espongono funzionalità diverse. Ciononostante, nella maggior parte dei casi, non è particolarmente difficoltoso eseguire il porting di macro esistenti.

---

1 Questo articolo è stato pubblicato su *Linux Journal Edizione Italiana* N. 37 del Giugno 2003 edito dalla Duke Italia (http://www.duke.it); quest'ultima detiene i diritti di pubblicazione e utilizzazione economica di quest'articolo; l'autore è stato autorizzato a renderlo pubblico sul proprio sito Web personale.

La “terza via”

Si sta lavorando allo sviluppo di un ulteriore framework che unisca le due possibilità viste (sviluppo di componenti e sviluppo di macro): *Language Independent Office Scripting Framework* (sito di riferimento http://framework.openoffice.org/scripting/index.html).
In pratica si cerca di fornire all'utente finale la possibilità di creare degli script in una varietà di linguaggi (teoricamente in qualsiasi linguaggio per cui esista un bridge all'architettura UNO), senza dover sviluppare componenti esterni, e di associare tali script o a specifici documenti o all'installazione di OpenOffice.org, il tutto caricato ed eseguito al runtime.
Al momento di scrivere l'articolo la versione disponibile di questo framework è la Early Developer release (versione 0.1), compatibile con OpenOffice.org 643C: è indubbio che dovrà passare ancora un pò di tempo prima di vedere una release sufficientemente completa del framework.

Fondamenti di OpenOffice.org Basic

Uno degli aspetti fondamentali della programmazione attraverso le API di OpenOffice.org Basic è quello di ottenere un “servizio”. I servizi sono di due tipi:

- indipendenti
- dipendenti dal contesto

i primi possono operare in qualsiasi situazione, senza assunzioni sul contesto in cui vengono eseguiti, mentre i secondi possono essere eseguiti solo dopo aver settato opportunamente il contesto su cui vanno ad operare.
I servizi indipendenti vengono creati usando

```
createUnoService(nomeServizio)
```

dove nomeServizioè una variabile o una stringa contenente il nome del servizio.
Per usare i servizi dipendenti dal contesto è necessario conoscere quali servizi richiamare per primi e poi creare i servizi ausiliari. Per esempio per creare un nuovo documento di testo è necessario prima avere a disposizione un servizio di tipo Desktop() (o, in alternativa, eseguire il codice all'interno di un documento attivo per cui tale servizio è già disponibile) e poi creare un riferimento ad un nuovo documento:

```
mioDesktop = createUnoService( _
          "com.sun.star.frame.Desktop")
docUrl = "private:factory/swriter"
docWriter = mioDesktop.loadComponentFromUrl( _
          docUrl, "_blank", 0, mNoArgs)
```

mentre per aprire un documento esistente, basta cambiare il contenuto della variabile docUrl, specificando la URL del file da aprire (in **Tabella 1** vengono riportati i diversi modi di specificare una Url), per esempio:

```
docUrl = "file:///home/ivan/docs/test.sxw"
```

o, nel caso si usi sistemi Windows:

```
docUrl = "file:///C|/document/test.sxw"
```

Si noti come i servizi hanno un nome che inizia con il prefisso "com.sun.star" (da esso risulta evidente l'origine delle API, ovvero StarOffice di Sun).
Tale prefisso è necessario anche per accedere ad eventuali costanti predefinite. Per esempio, le costanti che identificano il testo normale e quello grassetto sono, rispettivamente:

```
com.sun.star.awt.FontWeight.NORMAL
com.sun.star.awt.FontWeight.BOLD
```

Come si vede, dopo il solito prefisso, ci sono alcuni nomi, che identificano i moduli che forniscono i vari servizi. In **Tabella 2** sono riportati alcuni dei principali moduli a disposizione.

Tabella 1: diversi modi di specificare una Url

Esempio	***Significato***
http://www.nome.ext	Specifica un file (locale o remoto) attraverso il protocollo HTTP
ftp://ftp.nome.ext	Specifica un file (locale o remoto) attraverso il protocollo FTP
file:///C|/documenti/test.txt	Specifica un file presente sul file system locale (C rappresenta l'unità logica nel caso di sistemi Windows)
news://host.add.com	Connessione ad un server di news attraverso il protocollo NNTP
private:factory/nomeApp	Url di tipo privato usato in OpenOffice.org per creare documenti (nomeApp può essere swriter, scalc e così via)

Tabella 2: i principali moduli e i loro servizi.

Moduli	***Servizi***
com.sun.star.awt	Servizi per elementi grafici di base (punti, font, ...)
com.sun.star.chart	Servizi per la realizzazione di diagrammi
com.sun.star.drawing	Permette la realizzazione di figure geometriche da includere nei documenti
com.sun.star.frame	Utile per creare nuovi documenti o aprire quelli esistenti
com.sun.star.presentation	Servizi per creare presentazioni
com.sun.star.sheet	Servizi per creare fogli di calcolo
Com.sun.star.style	Servizi per la gestione della formattazione (stili)
com.sun.star.table	Servizi per inserire tabelle in documenti di testo o fogli di calcolo
com.sun.star.text	Servizi per la gestione/manipolazione del testo nei documenti di Writer

Alcuni servizi, ed in particolare quelli ottenibili dai documenti, offrono un particolare tipo di interfaccia (XmultiServiceFactory) che può essere usata per acquisire nuove istanze di servizi richiamando il metodo createInstance() a cui viene passato come parametro il nome del servizio da creare.
Una volta creato un servizio è possibile settarne/reperirne le caratteristiche attraverso le sue proprietà (properties). Tali proprietà sono delle coppie (nome, valore). Per esempio per settare la proprietà FillColor con il colore blu del servizio RectangleShape:

```
rettangolo = documento.createInstance( _
          "com.sun.star.drawing.RectangleShape")
rettangolo.FillColor = RGB(0, 0, 255)
```

Alcuni componenti possiedono delle collezioni (o altri tipi di contenitori) di oggetti. Per accedere agli elementi contenuti è possibile accedere attraverso:

1. il nome degli elementi, se tale nome esiste;
2. accedere attraverso gli indici;
3. in maniera sequenziale.

Tutti i contenitori offrono almeno uno dei metodi di accesso sopraindicati, alcuni permettono di combinare due o più metodi di accesso.
Un altro aspetto fondamentale è la possibilità di intercettare e gestire particolari eventi. Il modello di gestione è quello usato in Java: alcune azioni particolari scatenano degli eventi, per intercettare i quali è necessario aver registrato dei listeners (ascoltatori).
Una eccezione a questo modello è rappresentata da tipi particolari di procedure che gestiscono le finestre di dialogo grafico. Tali procedure seguono una convenzione sui nomi (per ogni evento che si vuole gestire è necessario usare un nome specifico, un po' quello che accade per VBA in ambiente Windows).

Macro: alcuni esempi

Proviamo a scrivere alcune semplici macro d'esempio, facendo uso delle funzionalità di Writer e di Calc. Supponiamo di voler sostituire delle parole con altre in un documento di testo. Le parole da sostituire sono memorizzate in un foglio di calcolo. La macro apre il foglio di calcolo, legge una riga alla volta e, per ogni riga, ricerca il termine della prima colonna e sostituisce ogni sua occorrenza con il termine presente nella seconda colonna.
Per prima cosa è necessario aprire il foglio di calcolo (siccome ogni documento ne contiene diversi, faremo riferimento al primo, avente indice 0):

```
desktop = createUnoService( _
          "com.sun.star.frame.Desktop")
url = "file:///c|/documenti/sostituisci.sxc"
fogli = desktop.LoadComponentFromURL( _
          url, "_blank", 0, mNoArgs)
foglio = fogli.Sheets(0)
```

A questo punto si esegue un ciclo su ogni cella fintantoché la cella in esame sulla prima colonna (indice 0) non è vuota:

```
riga = 0
cerca = foglio.getCellByPosition(0, _
           riga).String
```

```
Do While cerca<>""
 rem sostituisci testo
  ...
 riga = riga + 1
 cerca = foglio.getCellByPosition(0, _
          riga).String
Loop
```

All'interno del ciclo, oltre a incrementare il contatore di riga, è necessario effettuare la sostituzione. Per farlo si ricorre ad un servizio particolare, disponibile sui documenti di testo. Esso si chiama "ReplaceDescriptor". In pratica è necessario settare sia la parola da ricercare che quella da sostituire, poi si invoca "replaceAll()" sul documento passandogli il ReplaceDescriptor settato:

```
descrittore = createReplaceDescriptor
descrittore.SearchString = cerca
descrittore.ReplaceString = _
         foglio.getCellByPosition(1, riga).String
ThisComponent.replaceAll(descrittore)
```

Rispetto ai frammenti di codice presentati in precedenza, si noti come non viene creato nessun riferimento ad un documento testuale, ma viene usato un riferimento a "ThisComponent". Questo codice funzionerà solo se esiste un documento aperto di Writer su cui si esegue la macro di sostituzione (eseguendo la macro all'inderno dell'IDE Basic, si avrà un errore).

Documentazione

Il problema principale per chi si accinge a imparare un nuovo linguaggio o a prendere dimestichezza con un nuovo modello è quello di trovare della buona documentazione. In bibliografia trovate alcuni riferimenti, ma è utilissimo sia l'help contestuale (in particolare quello dedicato al Basic) sia il sito http://www.ooodocs.org/ (*The OpenOffice Documentation Project*). Inoltre potete far riferimento ad uno delle numerose liste di discussione presenti alla pagina http://www.openoffice.org/mail_list.html.

Registratore di macro?

Per programmare le applicazioni che compongono OpenOffice.org è necessario avere chiaro il modello di riferimento citato (seppur a grandi lineee) e prendere dimestichezza sia con le funzionalità specifiche delle applicazioni che con i diversi servizi offerti a livello di programmazione.Un modo semplice per iniziare a conoscere OpenOffice.org Basic sarebbe quello di registrare alcune azioni con il Registratore di Macro e poi "curiosare" sul codice effettivamente generato. È sorprendente che tale registratore non compaia (mentre è presente nelle versioni di StarOffice precedenti al rilascio di OpenOffice.org).
Alla pagina http://framework.openoffice.org/proposals/macro/macrorecording.html viene spiegato, con dovizia di particolari, il motivo che ha portato all'esclusione di tale strumento.
Anche se gli ostacoli tecnici sono molti è auspicabile fornire questo strumento in quanto è un indubbio vantaggio per chi si accinge a programmare con OpenOffice.org Basic.

Organizzazione delle macro

Quando le macro create non sono più funzioni o procedure "semplici" si può ricorrere a diversi tipi di organizzazione del codice.
Già la suddivisione del codice in funzioni e procedure permette di "isolare" funzionalità specifiche. Esse vengono sempre create all'interno di un modulo di default. Ogni modulo ha però un vincolo:

non può contenere più di 64 KByte di testo. Se si eccede tale limite (o semplicemente risulta comodo isolare ulteriormente certe funzionalità) si possono creare nuovi moduli. Esistono anche moduli particolari, chiamati “Dialog Modules”: essi combinano la struttura di dialog box, le proprietà degli elementi ivi contenuti e gli eventi associati in un singolo modulo.
I moduli che contengono codice in qualche modo correlato possono essere raggruppati in una libreria. Ad ogni documento (o template) si può associare un numero qualsiasi di librerie (l'organizzazione viene schematizzata in **figura 1**).

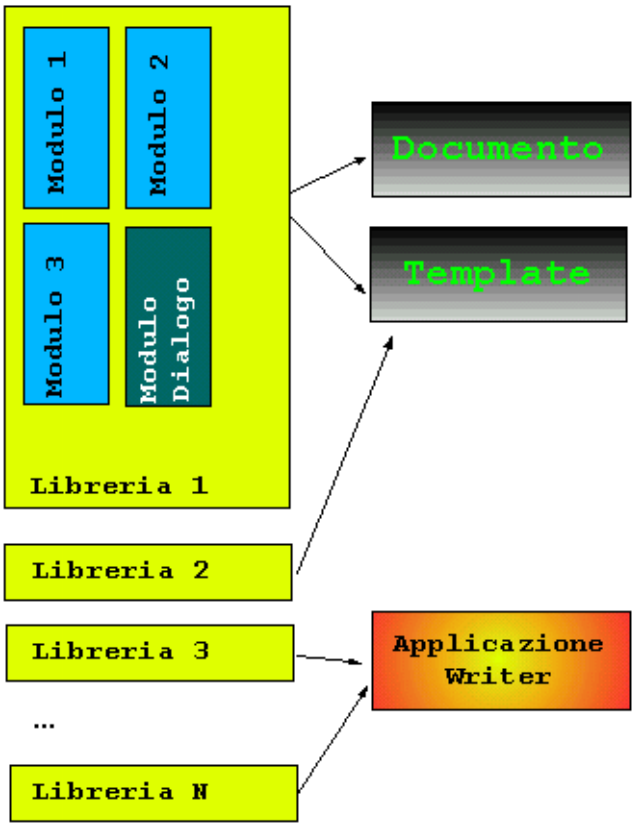

Figura 1: Possibile organizzazione delle macro

In questo modo è possibile ottenere la modularizzazione del codice organizzandolo in diversi livelli. Ogni volta che un documento viene salvato, vengono salvate anche tutte le librerie associate.
È possibile salvare le macro anche in file separati. In tal caso esse non vengono caricate automaticamente all'apertura di OpenOffice.org o del documento, ma è necessario farlo manualmente o attraverso un'altra macro.

Debug delle macro

Per agevolare eventuali operazioni di debug esistono delle API particolarmente interessanti, utili per ispezionare gli oggetti al run-time [1]:

- DBG_properties(): restituisce tutte le proprietà definite per l'oggetto su cui viene applicata. Prima di accedere a questa funzione è meglio verificarne l'esistenza con isNull();
- DBG_methods(): simile alla precedente, ma che restituisce tutti i metodi definiti per l'oggetto su cui viene applicata;
- DBG_supportedInterfaces(): restituisce tutte le interfacce supportate dall'oggetto corrente.

Per esempio, possiamo definire la funzione:

```
Sub displayInfos(oObject as Object)
  MsgBox oObject.DBG_properties()
  MsgBox oObject.DBG_methods()
  MsgBox oObject.DBG_supportedInterfaces()
End Sub
```

richiamandola con:

```
displayInfos(ThisComponent.createReplaceDescriptor)
```

Si ottiene, come risultato, quanto mostrato in **figura 2**.

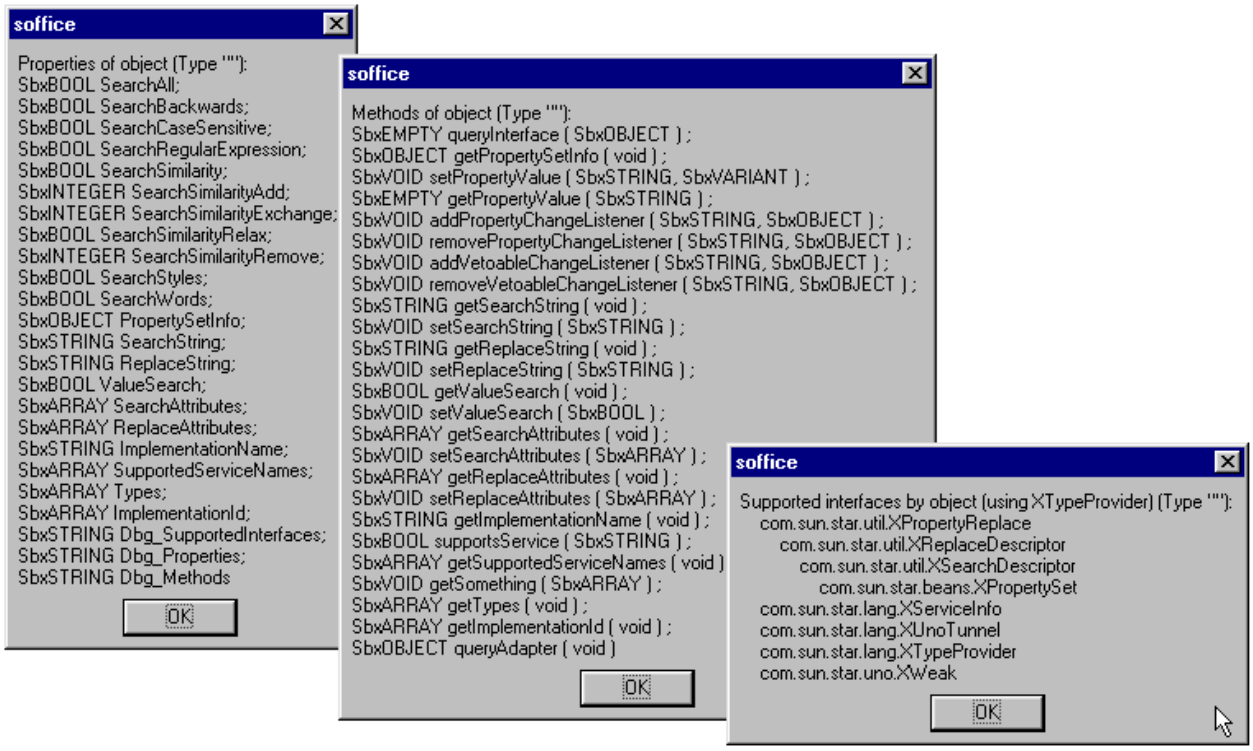

Figura 2: Proprietà, metodi e interfacce supportate di ReplaceDescriptor

Conclusioni

Si è visto che OpenOffice.org offre diversi strumenti di sviluppo per personalizzare le applicazioni. Essendo un prodotto sviluppato da programmatori volontari e non destinato alla vendita, è naturale che non segua le logiche commerciali e che ci sia un interesse a fornire tutti gli strumenti e la documentazione possibile per far evolvere il prodotto e per renderlo “aperto” a nuove idee e nuovi sviluppi.
Il mio consiglio è di inizare ad usare OpenOffice.org per le operazioni quotidiane e, via via che ci si impratichisce nel suo uso, iniziare la scrittura di macro per poi passare ad approfondirne la programmazione di componenti in uno specifico linguaggio.

Bibliografia

[1] "*StarOffice Programmer's Tutorial*", C. Kirsch, Sun Microsystems, http://soldc.sun.com/staroffice
[2] "*OpenOffice API reference manual*", AAVV, http://api.openoffice.org/common/ref/com/sun/star/module-ix.html
[3] "*OpenOffice.org Developers's Guide*", AAVV, http://api.openoffice.org/DevelopersGuide/DevelopersGuide.html