IL PRIMO MAGAZINE ITALIANO DEDICATO A VISUAL BASIC, C#, .NET, ACCESS E SQL SERVER

Poste Italiane S.p.A. - Spedizione in Abbonamento Postale - D.L. 353/2003 conv. in L. 27/02/2004 n°46 art.1,comma 1, DCB PERUGIA

VISUAL BASIC & .NET JOURNAL

n.68

Marzo/Aprile 2006 - bimestrale - Anno 12 - Euro 7,75

Progettazione di sistemi di DataWarehouse

Introduzione alla Computer Simulation

Compressione di file in VB

Creare Add-in per VBE in MS Access

Analisi degli accessi ad un sito web

La programmazione Template-Oriented

Controllo remoto in VB .NET: il registry

GRUPPO EDITORIALE INFOMEDIA

Programmazione delle Smart Card

Standard, specifiche e piattaforme di sviluppo per Java, C e Visual Basic

Internet non è in grado di proporre un mezzo di identificazione certificata degli interlocutori né un livello adeguato di protezione delle trasmissioni: le tecnologie basate su smart card permettono di sviluppare applicazioni sicure per i settori delle telecomunicazioni mobili, PayTV, commercio elettronico, sistemi bancari e di accesso sicuro ai sistemi informativi.

Tra gli argomenti affrontati nel testo:

lo standard ISO 7816 - la piattaforma Javacard - Architettura PC/SC, le specifiche PKCS#11 - il Framework OpenCard - utilizzo di smart card con Windows, Outlook e Internet Explorer - Crittografia.

Viene inoltre descritto un emulatore di smart card e di dispositivo di lettura utilizzabile per esercitarsi con le istruzioni APDU ISO 7816.

€ 18.00 - pagg. 160
ISBN 8881500140

GRUPPO EDITORIALE INFOMEDIA
THE SOFTWARE SIDE OF YOUR MIND

WEB: http://www.infomedia.it - e-mail: book@infomedia.it - tel: 0587/736460

Your potential. Our passion.™

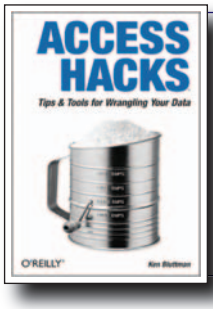
Microsoft

Nuovo Visual Studio 2005.
La differenza si vede.

Scopri la differenza! Appena inizierai a sviluppare sarà evidente. Visual Studio® 2005 ti offre più di 400 nuove funzionalità, tra cui controlli Web e Windows®, che riducono la parte più noiosa e ripetitiva del tuo lavoro. Così puoi creare codice di qualità ed essere più produttivo. **microsoft.com/italy/difference/**

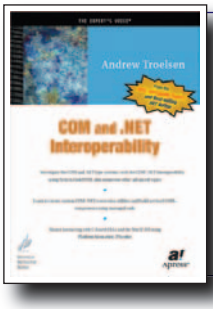
Microsoft®
Visual Studio 2005

IN OFFERTA VBJ 68



Access Hacks Tips & Tools for Wrangling Your Data
di K. Bluttman

O' Reilly
ISBN 0596009240
352 pp – 28,00€

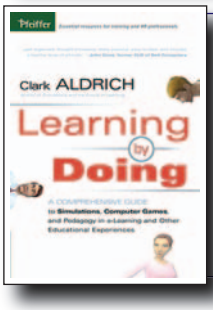


COM and .NET Interoperability
di A. Troelsen

Apress
ISBN 1590590112
770 pp – 64,15€

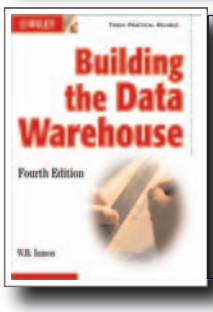
Programmare Microsoft Access 2003
di R. Dobson

Mondadori Informatica
ISBN 8804528907
896 pp – 75,00€



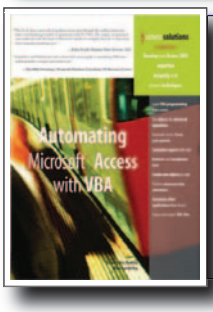
Learning by Doing:
A Comprehensive Guide to Simu–lations, Computer Games, and Pedagogy in e–Learning and Other Educational Experiences
C. Aldrich

John Wiley
ISBN 0787977357
400 pp – 60,80€



Building the Data Warehouse, 4th Edition
di W. H. Inmon

John Wiley
ISBN 0764599445
543 pp – 52,90€



Automating Microsoft Access with VBA
di M. Gunderloy e S. Harkins

Que
ISBN 0789732440
408 pp – 38,95€

Scrivi a book@infomedia.it specificando nell'oggetto della e-mail:

IN OFFERTA VBJ n. 68

OPPURE

inviaci il coupon sottostante al numero di fax

0587/732232

Potrai acquistare i libri qui riportati con uno

SCONTO ECCEZIONALE

del **10%** anche se acquisti solo **un libro**

OPPURE

del **20%** se acquisti **3 libri**

GRUPPO EDITORIALE INFOMEDIA

Web: http://book.infomedia.it • E-mail: book@infomedia.it • Fax: 0587/732232

THE SOFTWARE SIDE OF YOUR MIND — VBJ 68

DATI ANAGRAFICI

NOME	COGNOME	CODICE CLIENTE	
VIA/PIAZZA	CAP	CITTÀ	PROV.
TELEFONO (*) FAX	E-MAIL (*)		
DITTA			
INTESTAZIONE FATTURA		P.IVA	

Garanzia di riservatezza - Gruppo Editoriale Infomedia garantisce la massima riservatezza dei dati da lei forniti e la possibilità di richiederne gratuitamente la rettifica o la cancellazione scrivendo a: Responsabile Dati Gruppo Editoriale Infomedia Srl - v. Valdera P. 116 - 56038 Ponsacco (PI). Le informazioni custodite nel nostro archivio elettronico verranno utilizzate al solo scopo di inviarle proposte commerciali. In conformità alla legge 196/03 sulla tutela dati personali.

ORDINE

TITOLO	CODICE ISBN	PREZZO	QUANTITÀ

(*) campo obbligatorio

SPESE DI SPEDIZIONE

☐ PER POSTA - +€3,00 ☐ A MEZZO CORRIERE - euro 7,00

SI, mi voglio iscrivere gratuitamente alla newsletter Infomedia che mi tiene aggiornato sui prossimi argomenti delle mie riviste di programmazione e mi informa delle offerte, le nuove uscite e le prossime pubblicazioni; questo è il mio indirizzo di posta elettronica _____@_____ So che posso annullare la mia iscrizione in qualsiasi momento alla pagina http://www.infomedia.it/newsletter.htm

TIPO DI PAGAMENTO

☐ ALLEGO ASSEGNO BANCARIO INTESTATO A "GRUPPO EDITORIALE INFOMEDIA Srl" - NON TRASFERIBILE

☐ CONTRASSEGNO (solo per spedizione a mezzo posta + €3,00)

☐ ALLEGO VERSAMENTO SU CC. POSTALE N.14291561 INTESTATO A "GRUPPO EDITORIALE INFOMEDIA SRL - PONSACCO". Sul modulo di C.C.POSTALE scrivere la causale del versamento.

☐ ALLEGO FOTOCOPIA DEL BONIFICO BANCARIO EFFETTUATO SU C/C 10005424 CAB 71120 ABI 6370 CASSA DI RISPARMIO DI VOLTERRA AGENZIA DI PONSACCO

☐ CARTA DI CREDITO VISA/MASTERCARD/CARTASÌ

N. _____ SCADENZA _____

TITOLARE _____

FIRMA _____ NATO IL _____

Le spese di spedizione sono soggette a variazioni in base alle tariffe postali. Per i pagamenti con assegno, c/c postale e bonifico bancario consigliamo di chiedere conferma in redazione oppure di consultare il nostro sito internet. Grazie.

VISUALBASIC & .NET JOURNAL
online.infomedia.it
n. 68 - marzo/aprile 2006
bimestrale - anno dodicesimo

Direttore Responsabile
Marialetizia Mari *(mmari@infomedia.it)*

Direttore Esecutivo
Francesco Balena *(fbalena@infomedia.it)*

Managing Editor
Renzo Boni *(rboni@infomedia.it)*

Collaboratori
Gionata Aladino Canova
Stefano Corti
Daniele De Pedis
Raffaele Di Natale
Fabio Perrone
Stefano Savo
Scott Swigart
Fabio Testa
Lorenzo Vandoni

GRUPPO EDITORIALE INFOMEDIA

Direzione
Natale Fino *(nfino@infomedia.it)*

Marketing & Advertising
Segreteria: 0587/736460
marketing@infomedia.it

Amministrazione
Sara Mattei
(amministrazione@infomedia.it)

Grafica
Manola Greco *(mgreco@infomedia.it)*

Technical Book
Lisa Vanni (*book@infomedia.it)*

Segreteria
Enrica Nassi
(info@infomedia.it)

Stampa
TIPOLITOGRAFIA PETRUZZI
Citta' di Castello (PG)

Ufficio Abbonamenti
Tel. 0587/736460 – Fax 0587/732232
e–mail: abbonamenti@infomedia.it
www.infomedia.it

Gruppo Editoriale Infomedia srl
Via Valdera P., 116 – 56038 Ponsacco (PI) Italia
Tel. 0587/736460 – Fax 0587/732232
red_vbj@infomedia.it
Sito Web www.infomedia.it

Visual Basic Journal è *una rivista di Gruppo Editoriale Infomedia S.r.l. Via Valdera P, 116 Ponsacco – Pisa.*
Registrazione presso il Tribunale di Pisa n. 20/1999

EDITORIALE

La migrazione della applicazioni "legacy"

In questo periodo sto analizzando più da vicino il processo di migrazione delle applicazioni VB6 verso il mondo .NET. Devo confessare che in questi anni mi sono concentrato soprattutto sulle enormi potenzialità di VB.NET, senza prestare molta attenzione ai problemi posti dal processo di migrazione. Per mia fortuna, infatti, non ho avuto la sfortuna di dover migrare molte delle mie applicazioni, e quelle poche che ho effettivamente dovuto portare in VB.NET le ho riscritte da zero, per approfittare delle nuove feature di scalabilità di ADO.NET o della potenza dei nuovi controlli di Windows Forms. Per mia fortuna si trattava di applicazioni relativamente piccole, scritte per giunta con uno stile molto object–oriented, quindi la traduzione è stata relativamente semplice. Ma è stato comunque un lavoro duro, perchè ho preferito fare tutto a mano e stare lontano dal wizard Artinsoft, quello incluso gratuitamente in Visual Studio.

Purtroppo il mio approccio al problema non è di fatto applicabile in molti casi, e sicuramente non è applicabile al progetto su cui sto lavorando al momento, che consta di molte migliaia di righe di codice e che è frutto di numerosi anni–uomo di lavoro. Di certo non è semplice effettuare una migrazione manuale di "mostri" di questo tipo, che spesso non sono neanche scritti in modo ordinato e con stile OOP. Senza contare che per effettuare una migrazione corretta occorre conoscere davvero bene le tante sottili differenze tra i due linguaggi, differenze che spesso si traducono in bug davvero insidiosi.

Per farla breve, da un po' di tempo sto lavorando a un wizard di migrazione da VB6 a VB.NET che funzioni in modo sensibilmente migliore di quello incluso in Visual Studio. Sono ancora in una fase di "alfa test", ma i risultati sono decisamente in–coraggianti. Il mio wizard è già in grado di gestire correttamente costrutti normal–mente considerati "intraducibili" come i Gosub (calcolati e non), gli array con indice inferiore qualsiasi, i controlli Line, Shape, e Data control (con relativo binding), i menu popup, la creazione dinamica di controlli, e un sacco di altra roba. Chi fosse interessato può tenersi aggiornato sugli sviluppi di questo progetto e fornire consigli e suggerimenti sul mio blog.

Francesco Balena
fbalena@codearchitects.com

MARZO/APRILE

N.68

SOMMARIO

RUBRICHE

SPECIALE

VB6

ACCESS

WEB

SOFTWARE ENGINEERING

APPLICATIVI

Progettazione di sistemi di Data Warehouse

Le linee guida per la comprensione e la progettazione dei sistemi di Data Warehouse

di Fabio Testa

Per definire cosa sia un data warehouse (da ora in poi DWH) ci rifacciamo alla definizione di uno dei padri fondatori della disciplina del data warehousing, William H. Inmon:

"*Il data warehouse è una collezione di dati: orientata al soggetto, integrata, non volatile, dipendente dal tempo*"

I concetti chiave

- *orientata al soggetto*: perché il DWH è *orientato a temi specifici* dell'azienda piuttosto che alle applicazioni o alle funzioni. L'obiettivo, quindi, non è più quello di minimizzare la ridondanza mediante la normalizzazione ma quello di fornire dati che abbiano una struttura in grado di favorire la produzione e fruizione delle informazioni;
- *Integrata*: il DWH deve essere orientato all'integrazione della raccolta dati, in esso confluiscono *dati* provenienti da più sistemi transazionali e da fonti esterne. L'integrazione può essere raggiunta percorrendo differenti strade che non si escludono tra: omogeneità semantica di tutte le variabili, utilizzo di metodi di codifica uniformi, utilizzo delle stesse unità di misura, ecc.;
- *non-volatile*: i dati contenuti nel DWH consentono accessi in sola lettura;
- *dipendente dal tempo*: i dati di un DWH hanno un orizzonte temporale molto più esteso rispetto a quelli archiviati nei sistemi operazionali. L'osservazione "storica" del dato è una peculiarità del DWH che ha dati che si legano alla misura tempo e si astraggono dalla pura scansione di eventi del processo aziendale (es. singoli scontrini fiscali nella vendita al dettaglio).

La struttura di un generico DWH comprende:

- una parte di ETL (estrazione, trasformazione e caricamento) che permette la "pulitura" e conformazione dei dati provenienti dai vari sistemi operazionali;
- una serie di data mart come unità di osservazione dei processi aziendali;
- una parte di presentazione che non è altro che un insieme di applicativi desktop e/o web e di report creati su strumenti proprietari (es. Crystal Reports, Business Objects, vari strumenti client OLAP, ecc.).

Fabio Testa *si occupa di progettazione, consulenza e project management su software web/desktop based e di sistemi di DW ed ha lavorato nei settori servizi IT per Public utilities, PMI, banche e centri servizi bancari.*

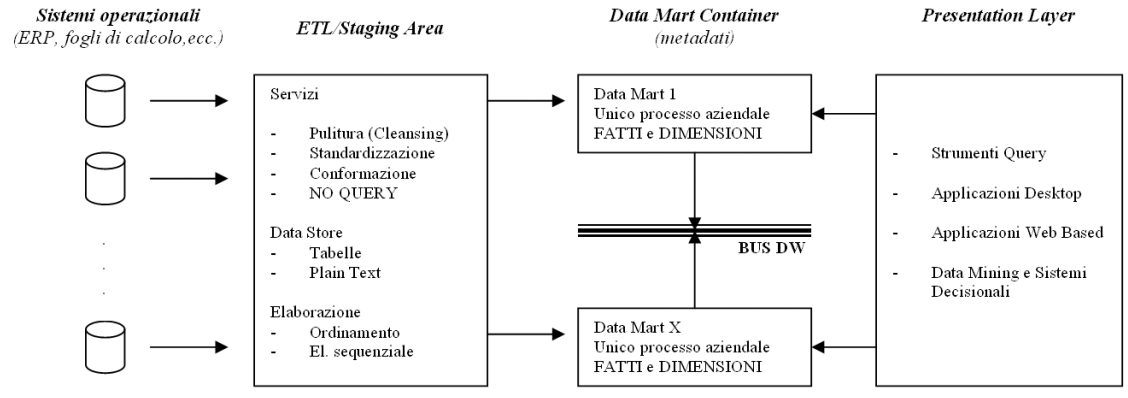


Figura 1 Sistema di DW

La **Figura 1** rappresenta lo schema di un generico sistema di DWH. Alcuni dei concetti espressi nella figura saranno esplicati nel seguito dell'articolo.

Concetti fondamentali sulla modellazione dimensionale

Il modello di dati dimensionale differisce dal modello normalizzato, ma ne eredita i concetti fondamentali.

Sono quindi ancora presenti le entità (nell'esempio in **Figura 2**, allievo e scuola), gli attributi (nome, cognome, ecc.) e, quando il modello dimensionale è relazionale, le relazioni. Cadono invece tutte le altre regole di normalizzazione dei DB relazionali (prima, seconda e terza forma normale).

Come si può osservare in **Tabella 1** si tende a denormalizzare il DB "appiattendo" nelle dimensioni valori "*ripetuti*".

Questo porta sicuramente ad uno spreco di spazio fisico su disco, ma per contro evita JOIN superflui per recuperare i dati sparsi su più tabelle (query più prestazionali).

Riguardo alle chiavi primarie (PK) e chiavi esterne (FK) delle tabelle di dimensioni e fatti, bisogna aggiungere che è opinione assunta dalla comunità degli esperti di DWH che le chiavi dei sistemi operazionali vanno sostituite da chiavi create "su misura" in area di staging, per separare completamente le logiche dei sistemi operazionali da quelle del DWH, rendendolo indipendente in tutto e per tutto. Queste chiavi così create sono chiamate *chiavi surrogate*, mentre le vecchie chiavi operazionali restano all'interno della tabella dei fatti diventando *dimensioni degenerate* che non sono legate a tabelle di dimensioni (es. numero fattura, numero di POS, ecc.). Le entità nei DB di DWH sono organizzate in: *tabelle dei fatti* (*fact tables*) e *tabelle delle dimensioni (dimension tables)*.

Le tabelle dei fatti sono le principali nel modello ed in esse sono memorizzate le *misure*

Tabella 1 Esempio di tabella denormalizzata

Rating	Descrizione	Livello	Qualità	Grado
1	Molto poco	Scarso	Bassa	speciale
2	Medio	Scarso	Media	standard
3	Buono	Buono	Media	standard
4	Molto buono	Buono	Scarsa	speciale

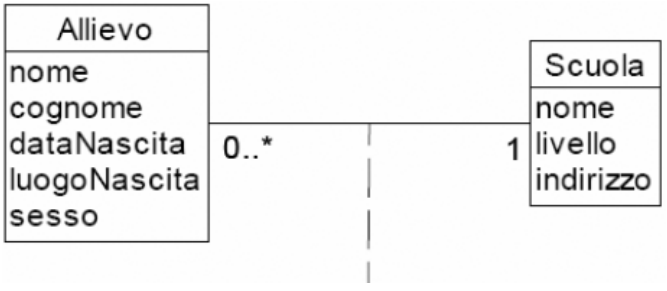


Figura 2 Entità Relazioni

(measures) delle prestazioni numeriche dell'azienda per un determinato processo aziendale. Supponiamo di analizzare la vendita di prodotti in un mercato e di scrivere l'importo e quantità delle vendite ogni giorno, per ciascun prodotto in tutti i negozi delle catena.

L'elenco degli attributi di dimensione definisce la *grana* ovvero la profondità dimensionale della tabella dei fatti. Le tabelle delle dimensioni sono quelle contenenti i "descrittori" della tabella dei fatti e di solito hanno molte colonne e poche righe, anche se è possibile incontrare tabelle delle dimensioni con milioni di righe (come ad esempio la dimensione cliente in alcuni modelli dimensionali – ad esempio entità clienti delle compagnie telefoniche). Spesso non è chiaro se la tabella debba essere considerata una tabella dei fatti o di dimensione; una buona regola può essere quella di chiedersi:

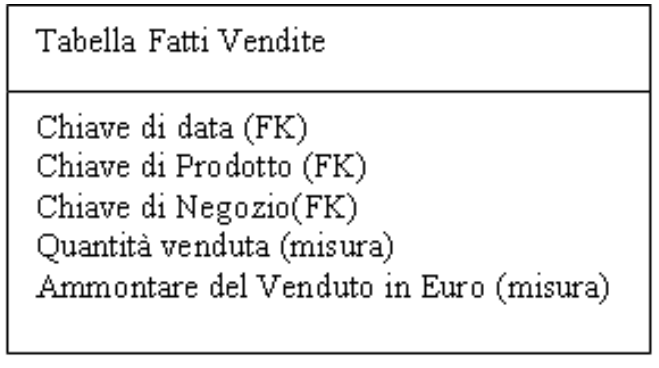


Figura 3 Esempio di tabella dei Fatti

La tabella contiene misure che parteciperanno a calcoli?

Se la risposta è SI la tabella è dei fatti, altrimenti siamo di fronte ad una candidata dimensionale.

Nell'analisi dimensionale si tende ad uniformare uno schema standard definito Schema a Stella (Star Schema).

La **Figura 5** mostra lo schema di un data mart che rappresenta i fatti di vendita di una ipotetica azienda. Si può osservare il classico schema a stella con la tabella dei fatti "Vendite Giornaliere", le sue misure "Quantità" e "Ammontare" e le chiavi esterne verso le varie dimensioni descrittive dei fatti (prodotti, promozioni, negozio, ecc.).

Non è sempre vero che bisogna attenersi ad uno schema star stretto.

In alcuni casi è necessario utilizzare anche schemi più articolati (uso di Outriggers Junk, ecc.) che non trattiamo nel presente articolo.

Quando lo schema indica dimensioni figlie, è definito a *fiocco di neve* (*snowflake,* **Figura 6**).

Il processo in quattro passi per la progettazione di un DWH

Per la progettazione di un sistema di data warehouse si può indicare uno schema in quattro passi che è stato proposto per primo dal guru del DWH Ralph Kimball.

Fase I: Selezionare il processo da modellare

Ascoltare gli utenti è il modo più efficiente per scegliere il processo aziendale ed è bene ricordare che quando si parla di processi aziendali non ci si riferisce ad un reparto o ad una funzione organizzativa; guardando ai reparti si rischia di duplicare le estrazioni, trasformazioni, pubblicazioni e viste dei dati.

Fase II: Dichiarare la grana del processo aziendale

Significa specificare esattamente ciò che rappresenta una singola riga della tabella dei fatti.

La domanda che ci si deve porre è: *Come si descrive un'unica riga della tabella dei fatti?*

Ecco alcuni esempi di dichiarazione di grana:

- Una istantanea mensile per ogni conto corrente
- Una voce su una fattura ricevuta da un medico
- Una singola voce di riga dello scontrino di vendita

La definizione di grana è fondamentale e se si scopre che nelle fasi III e IV c'è un errore di definizione, bisogna ritornare alla fase II.

Fase III: Scegliere le dimensioni che si applicano alla tabella dei fatti

Se la definizione della grana è chiara e soddisfacente trovare le dimensioni di solito è banale.

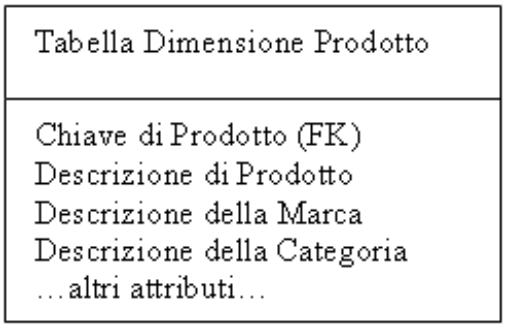


Figura 4 Esempio di Tabella Dimensioni

La domanda a cui rispondere è:

In che modo coloro che lavorano in una azienda descrivono i dati generati dal processo aziendale?

Fase IV: Identificare i fatti numerici che popoleranno ogni riga della tabella dei fatti.

La domanda per individuare le misure è: *Cosa stiamo misurando?*

È fondamentale che tutti gli attributi della tabella dei fatti candidati come misure rispet-

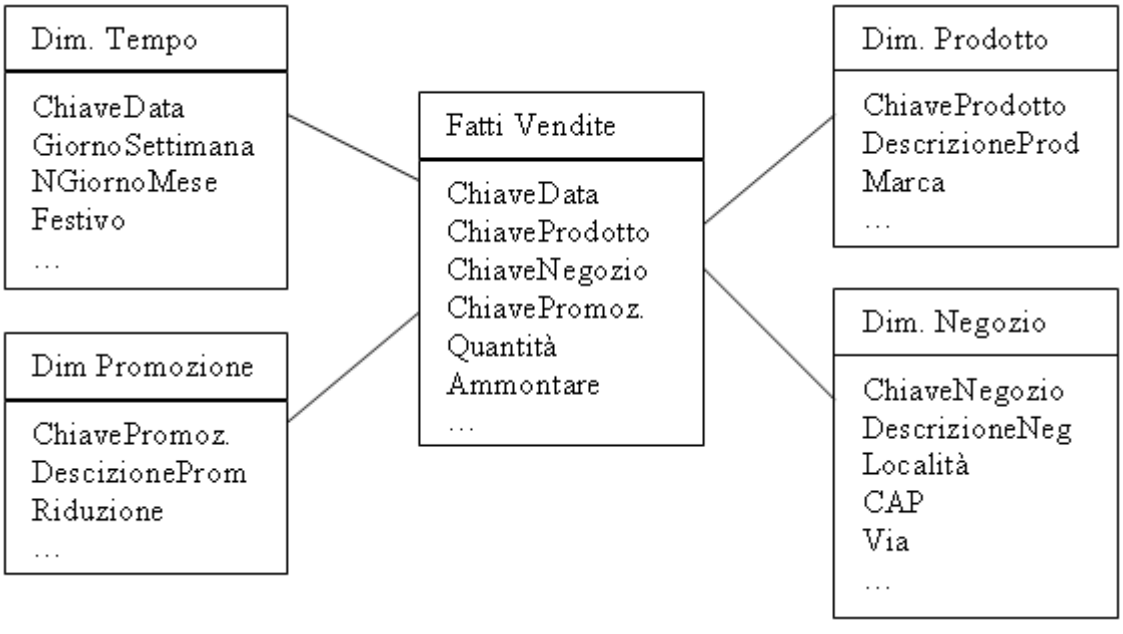


Figura 5 Star Schema

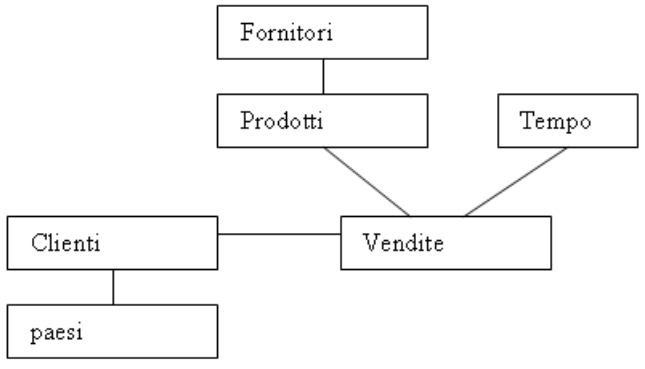


Figura 6 Snowflake Schema

tino la grana. I fatti che appartengono ad una grana diversa devono trovarsi in una tabella dei fatti distinta. In sintesi, tabelle di fatti diverse per misurazioni diverse.

Architettura BUS del data warehouse

L'implementazione di un sistema di DWH è molto complessa e va scomposta in più obbiettivi raggiungibili e condivisi dal committente.

Pur essendo questa una ottima regola da seguire in tutti i progetti informatici complessi, bisogna tener presente che scomporre il problema NON significa creare il DWH in parti isolate.

Un metodo di progettazione che si è rivelato molto utile nella progettazione dei sistemi DWH complessi è *l'architettura BUS*.

L'idea è quella di applicare alcuni dei paradigmi della programmazione object oriented ai DWH, pensando ai fatti ed alle dimensioni come ad oggetti riutilizzabili all'interno di più data mart.

Appare chiaro che nella fase di progettazione bisogna identificare fatti e dimensioni comuni che sono detti *conformati* da potersi utilizzare nei data mart.

Perché le dimensioni ed i fatti siano conformati è necessario abbiano chiavi di dimensioni e fatto coerenti, nomi di colonne e attributo coerenti, valori attributo e definizioni coerenti. In concreto le dimensioni ed i fatti confor-

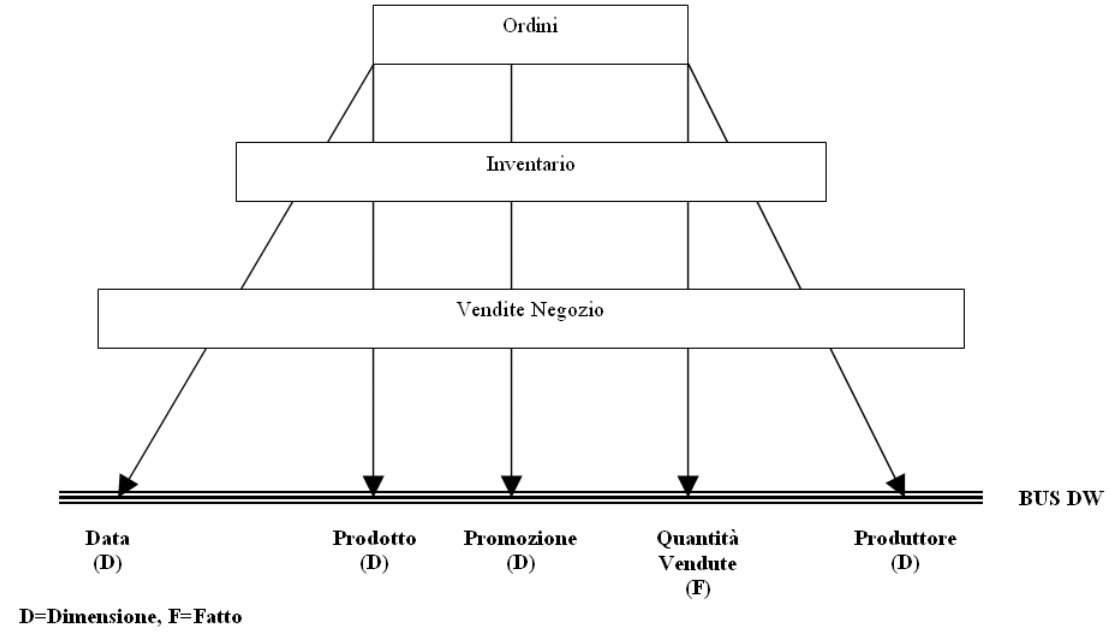


Figura 7 Bus Data Warehouse

mati sono un subset matematico identico oppure rigido rispetto alla dimensione più dettagliata.

Per l'individuazione delle dimensioni e fatti "riutilizzabili" ci viene in aiuto la *matrice bus del data warehouse* che ha sulle righe i data mart e sulle colonne le dimensioni. I data mart così individuati sono definiti *consolidati*. Parlando delle dimensioni conformate è necessario specificare che l'utilizzo del *subsetting* va esteso quando i modelli dimensionali affrontano problemi di analisi che richiedono livelli di dettaglio differenti su subset dello stesso insieme. Pensiamo ad esempio alla situazione di un DWH nel settore finanziario. In questo settore i prodotti possono essere eterogenei tra loro. In una banca le differenze che esistono tra un conto corrente di deposito e i depositi vincolati come i certificati di deposito sono minime ma significative (es. i depositi vincolati non hanno saldi minimi e limiti di scoperto). Una osservazione globale con possibilità di *slice and dice* (analisi a scelta dell'utente lungo tutte le dimensioni) su ciascun prodotto è necessaria, ma sicuramente può sorgere la necessità di analisi particolari sulle caratteristiche non in comune.

Tabella 2 Matrice Bus Data Warehouse

Processi Aziendali	Data	Prodotto	Negozio	Promozione	Magazzino	Produttore	Contratto	Spedizioniere
Vendite al dettaglio	X	X	X	X				
Inventario al dettaglio	X	X	X					
Consegne al dettaglio	X	X	X					
Inventario al magazzino	X	X			X	X		
Consegne di magazzino	X	X			X	X		
Ordini di acquisto	X	X			X	X	X	X

A questo punto un solo data mart non può più rispondere alle esigenze del committente e noi dobbiamo creare un nuovo data mart che risponda alle nuove esigenze di analisi. Osserviamo nella **Figura 9** l'operazione di conformazione sui due prodotti bancari, una operazione di intersezione o *Drill Cross* utile per rappresentare e quindi comprendere le caratteristiche comuni e non comuni dei due sottoinsiemi di prodotti.

Conclusioni

La realizzazione di un sistema di data warehouse è di solito un progetto complesso che richiede competenze disparate, da quelle di design del processo a quelle di modellazione DB, alla realizzazione di software per le aree

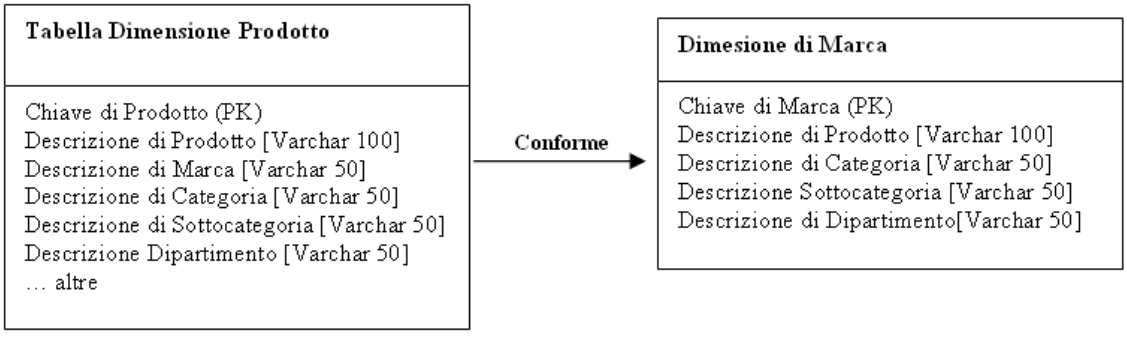


Figura 8 Dimensioni Conformate

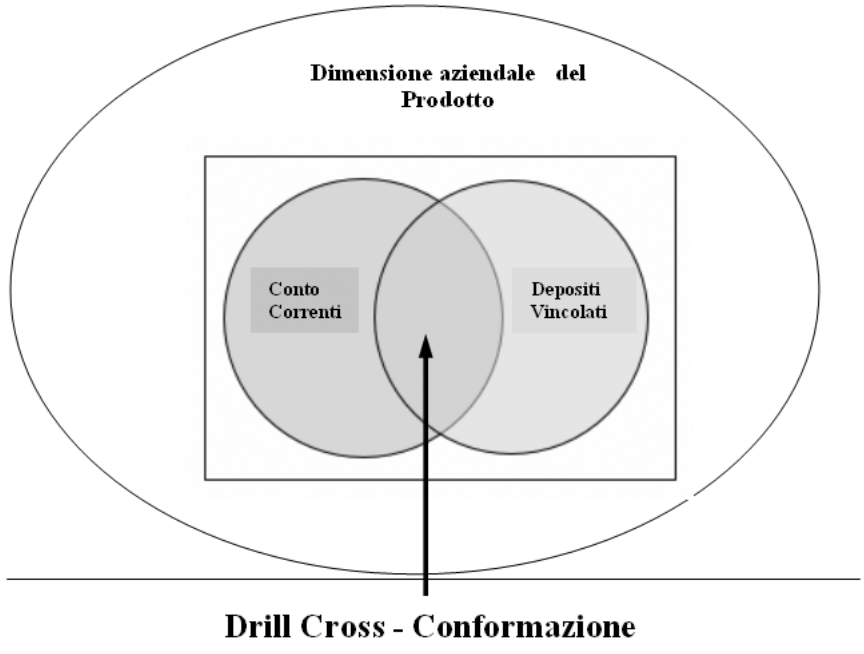


Figura 9 Bus Data Warehouse

di staging e dei sistemi di delivery della reportistica. In quanto complessi, questi progetti presentano molte difficoltà ed è importante che gli scopi del sistema di DWH siano chiari e definiti all'interno dei seguenti punti riassuntivi:

1. Rendere le informazioni dell'azienda facilmente accessibili ed i dati devono essere *intuitivi e leggibili:*
   - Fortemente deprecato l'utilizzo di codici numerici
   - Fortemente deprecato l'uso di codici parlanti
   - Utilizzo dei nomi attributi e tabelle chiari e per esteso
   - Struttura denormalizzata con pochi join (possibilmente struttura a stella)
2. Possibilità di presentazione in combinazioni infinite dei dati (*slicing and dicing*)
3. I dati devono essere *credibili*, *puliti* e la loro *qualità* deve essere *garantita*
4. Il DWH deve essere *elastico* ed adattarsi prontamente alle esigenze dell'utente
5. Il DWH deve essere *ACCETTATO* dal committente e dagli utenti finali (non sempre è scontato)
6. Il DWH deve essere un *bastione sicuro* per le informazioni
7. Il DWH deve servire al committente per *prendere decisioni migliori* e quindi la sua progettazione deve essere orientata a questo scopo

Bibliografia

[1] Kimball Ralph, Ross Margy – "*Data Warehouse la guida completa*", Hoepli, 2003
[2] Golfarelli Matteo, Rizzi Stefano – "*Data Warehouse - Teoria e pratica della progettazione*", Mc Graw Hill, 2002
[3] W. H. Inmon – "*Building the Data Warehouse (3rd Edition)*", John Wiley & Sons, 2005
[4] Ralph Kimball, Laura Reeves – "*The Data Warehouse Lifecycle Toolkit*", John Wiley & Sons, 1998

Riferimenti

[5] http://www.dmreview.com
[6] http://www.ittoolbox.com
[7] http://www.kimballuniversity.com
[8] http://www.datawarehouse.com
[9] http://www.intelligententerprise.com/info_centers/data_warehousing

Introduzione alla computer simulation

Un breve viaggio nel mondo della simulazione, di cui analizzeremo le possibilità, le tecniche di programmazione e faremo esperienza con alcuni esempi in VB6.

di Daniele De Pedis

Si sente spesso parlare di simulazioni fatte al computer per risolvere problemi scientifici, finanziari o anche problemi di carattere più generale. Ma cosa significa *in pratica* fare una simulazione al computer?

Prima di addentrarci in questo mondo, facciamo subito una distinzione fra *simulazione* ed *emulazione*. Molto spesso i due termini vengono usati in maniera scambievole ma in realtà essi stanno a significare due tecnologie molto diverse (e noi ci atterremo a questa distinzione).

- Con il termine *simulazione* si intende la riproduzione, mediante un algoritmo software, di processi intrinsecamente stocastici o governati da regole aleatorie. Tipico esempio è il lancio di un dado o di una moneta di cui, a priori, non possiamo prevedere il risultato, ma solo la probabilità che esso accada. Per la loro capacità di trattare processi aleatori, i programmi di simulazione vengono usualmente chiamati anche *programmi Monte-Carlo*, proprio in riferimento al gioco della roulette.
- Con il termine *emulazione* invece si intende la riproduzione di processi governati da rigide leggi fisico/matematiche. Tipici esempi sono il gioco degli scacchi, i programmi CAD o i simulatori di volo (che in realtà dovrebbero chiamarsi *emulatori di volo*) dove il comportamento dell'oggetto in esame è perfettamente prevedibile in base alle leggi che li governano.

In questo articolo affronteremo gli aspetti pratici dei programmi di simulazione.

Modellizzazione della simulazione

Prima di procedere alla scrittura di un programma di simulazione è necessario astrarre, dal sistema che si vuole simulare, quei componenti – e le loro interazioni – considerati importanti al fine della modellizzazione del sistema stesso; questo significa fare determinate assunzioni semplifica-

***Daniele De Pedis**, laureato in Fisica, è un ricercatore dell'Istituto Nazionale di Fisica Nucleare. Ha partecipato a numerosi esperimenti presso i laboratori del CERN e del FNAL. È autore o coautore di numerosi articoli su riviste scientifiche internazionali. Attualmente è responsabile del database di costruzione dell'esperimento Atlas in allestimento al CERN.*

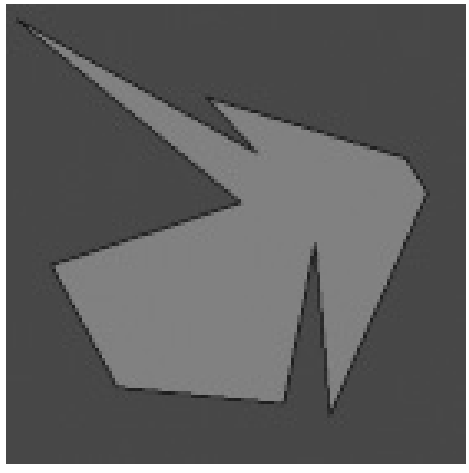


Figura 1 Figura geometrica di cui si vuole calcolare la superficie

tive che ne permettano l'implementazione software. Tuttavia, l'accuratezza dei risultati ottenuti dalla simulazione dipenderà strettamente da quanto tali assunzioni siano valide, perciò il primo passo sarà quello di definire esattamente i parametri importanti che si vogliono misurare e quali componenti del sistema li influenzeranno. Inoltre occorre tener presente che nello sviluppo di un programma di simulazione si procede (quasi) sempre per successive approssimazioni, cioè si inizia descrivendo in maniera semplice il processo da simulare, si confrontano i risultati di questo primo tentativo con la realtà e, nel caso, si apportano le necessarie migliorie al programma, si ripete la simulazione e si fa un nuovo confronto e così via.

Tale processo di iterazione *simulazione-verifica* non è un futile e inutile lavoro; la tipica domanda che sorge è: "*Qual è il beneficio di una simulazione se poi devo confrontarla con la realtà*?". Occorre puntualizzare, infatti, che in ogni nuova iterazione possiamo introdurre differenti valori dei parametri (o una descrizione che a priori non è prevedibile o difficilmente verificabile) e la nuova iterazione ci darà risultati differenti. L'insieme di tutte queste fasi di simulazione ci darà quindi un potere predittivo enorme sul comportamento del sistema in esame.

Per fare un semplice esempio supponiamo di simulare la caduta di un asteroide sulla Terra e supponiamo di saper descrivere, in termini di effetti distruttivi, il comportamento di un tale evento. Ci possiamo chiedere "*È preferibile che l'asteroide abbia un impatto come un singolo corpo oppure è meglio provocarne la frantumazione in corpi più piccoli prima che raggiunga la Terra*?". È chiaro che non abbiamo a disposizione un asteroide per confrontare i risultati della nostra simulazione con la realtà ma, per esempio, possiamo sulla base di eventi successi nel passato o su analisi di eventi analoghi accaduti sulla Luna inferire quali possano essere le conseguenze di un tale evento. Questi parametri allora possono essere introdotti nel nostro programma di simulazione, che ci darà quindi previsioni diverse a seconda della massa, traiettoria e composizione dell'asteroide in studio, o dei suoi frammenti, e quindi una risposta alla domanda precedente. È da notare che in questo esempio la parte di *simulazione*, cioè il processo aleatorio, è quello della frantuamazione e degli effetti distruttivi dei frammenti; viceversa, la loro traiettoria e i punti di impatto, essendo determinati da leggi fisiche, rappresentano la parte di *emulazione*.

Come è facilmente intuibile, una simulazione, essendo la riproduzione di processi alea-

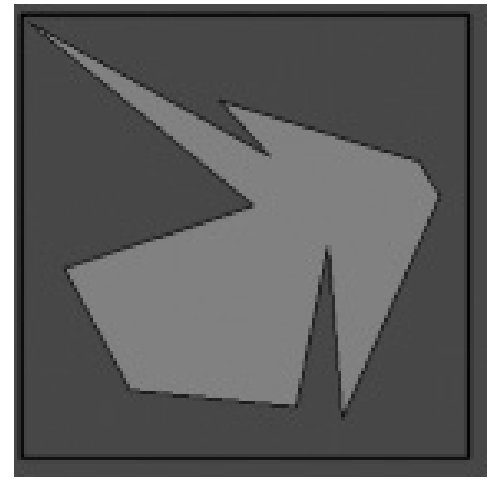


Figura 2 Figura geometrica di cui si vuole calcolare la superficie, inscritta in un quadrato

Tabella 1 La funzione Rnd in Visual Basic

Se il valore di seme è	La funzione Rnd ritorna
minore di zero	lo stesso numero ogni volta usando numero come seme
maggiore di zero	il successivo numero casuale nella sequenza
uguale a zero	il risultato più recente già generato
omesso	il successivo numero casuale nella sequenza

tori, sarà capace di riprodurre la realtà solo in senso statistico, cioè solo se il numero di eventi simulati è grande. Infatti un fondamentale teorema di statistica ci assicura, sotto ipotesi largamente generali, che l'errore fra la grandezza *misurata* e il suo valore *vero* è inversamente proporzionale alla radice quadrata del numero di misure effettuate [1].

Il cavallo di battaglia della simulazione: la funzione Rnd

Per quanto detto precedentemente appare evidente che nell'affrontare una simulazione è fondamentale la disponibilità di una procedura (funzione) che ci permetta di estrarre dei numeri a caso (random) in un intervallo predefinito di valori. Tutti i linguaggi di programmazione mettono a disposizione tale strumento e, ovviamente, VB non è da meno, infatti mette a disposizione due funzioni: la *Rnd* e la *Randomize*.

Analizziamo da vicino la funzione Rnd, la cui sintassi è ([2]):

```
Rnd [(seme)]
```

Essa ritorna un numero a caso di tipo Single – compreso fra zero e uno – a seconda del parametro *seme,* che è opzionale (vedi **Tabella 1**). Per ogni dato seme iniziale viene generata la stessa sequenza di numeri poichè, per ogni chiamata, la Rnd usa come seme il numero generato nella chiamata precedente. Quindi successive iterazioni che partono con lo stesso seme daranno luogo a sequenze identiche. Per ovviare a tale problema VB mette a disposizione l'istruzione *Randomize* che inizializza la Rnd con un seme derivato dall'orologio di sistema e quindi sempre diverso dai precedenti.

La ragione dell'esistenza di due differenti funzioni è dovuta alla necessità, in alcuni casi, di riprodurre esattamente la stessa sequenza, per esempio quando si vuol debuggare un programma o quando si vuole eseguire un programma con condizioni iniziali fisse.

I numeri generati dalla Rnd hanno una distribuzione piatta nell'intervallo 0-1, cioè ogni numero in tale intervallo ha la stessa probabilità di essere estratto. Questa non è una limitazione; partendo infatti da una distribuzione piatta fra 0-1, è possibile generare una qualsivoglia distribuzione in qualsiasi intervallo.

Per esempio, se vogliamo una distribuzione piatta nell'intervallo 10-15, sarà sufficiente definire una funzione

```
Public Function MyRnd() as single
MyRnd = 10 + 5 * Rnd
End Function
```

e in generale potremo scrivere

```
MyRnd=Min + (Max-Min) * Rnd
```

dove Max e Min sono gli estremi dell'intervallo.

A volte necessita estrarre a caso un numero intero in un dato intervallo, per esempio nel simulare il lancio di un dado o nell'estrazione dei numeri del Lotto. In tal caso si può usare la seguente funzione

```
Public Function intRnd() as long
intRnd = Int (Min + (Max - Min + 1) * Rnd)
End Function
```

Listato 1 Listato "piove a catinelle" (continua...)

```
   Private Declare Function Polygon Lib "gdi32" (ByVal hdc As Long, _
                   lpPoint As POINTAPI, ByVal nCount As Long) As Long

Private Type POINTAPI
  x As Long
  y As Long
End Type
Dim kPoint() As POINTAPI, nCount As Long

Dim FirstPoint As Boolean
Dim XO As Single, YO As Single
Dim xPos As Single, yPos As Single
Dim inPoint As Long, outPoint As Long, totPoint As Long
Dim PicArea As Single, Area As Single

Private Sub Form_Load()

  Width = 6300
  Height = Width + 1700

  With Picture1
    .Move 0, 0, 6000, 6000
    .FillStyle = vbFSSolid      ' FillStyle non puo' essere trasparente
    .FillColor = vbRed            ' Colore di riempimento figura
    .BackColor = vbBlue        ' Colore sfondo
    .ScaleMode = vbPixels    ' Misura in pixel.
    .AutoRedraw = True         ' Rendi l'immagine permanente
  End With

  FirstPoint = True
  inPoint = 0
  outPoint = 0

  Randomize            ' Rende la Rnd indipendente dalla iterazione in corso (vedi articolo)
End Sub

Private Sub Picture1_MouseMove(Button As Integer, Shift As Integer, x As Single, y As Single)
  lblX.Caption = "X :" & x
  lblY.Caption = "Y :" & y
End Sub

Private Sub Picture1_MouseDown(Button As Integer, Shift As Integer, x As Single, y As Single)

' Colleziona i punti in un array di POINTAPI
  nCount = nCount + 1
  ReDim Preserve kPoint(nCount)
  kPoint(nCount).x = x
  kPoint(nCount).y = y

  If FirstPoint = True Then  ' Non disegnare se primo punto
    Picture1.CurrentX = x
    Picture1.CurrentY = y
    XO = x
    YO = y
    FirstPoint = False
    Exit Sub
  End If

  Picture1.Line -(x, y)       ' Disegna la linea congiungendola all'ultimo punto
End Sub
```

Listato 1 (...fine)

```
Private Sub Picture1_DblClick()
  Picture1.Cls                              ' Cancella i vecchi punti e
  Polygon Picture1.hdc, kPoint(1), nCount   ' disegna la figura in colore
End Sub

Private Sub cmdCalcola_Click()
  If nCount = 0 Then
    MsgBox "Non c'e' nessuna figura disegnata !", vbOKOnly
    Exit Sub
  End If

  If cmdCalcola.Caption = "Calcola Superficie" Then
    cmdCalcola.Caption = "Stop"
  Else
    cmdCalcola.Caption = "Calcola Superficie"
  End If

  PicArea = Picture1.ScaleWidth * Picture1.ScaleHeight

  Do While cmdCalcola.Caption = "Stop"
    xPos = Picture1.ScaleWidth * Rnd
    yPos = Picture1.ScaleHeight * Rnd

    If Picture1.Point(xPos, yPos) <> vbBlue Then
      inPoint = inPoint + 1       'conta punti interni
    Else
      outPoint = outPoint + 1     'conta punti esterni
    End If

    If chkShow.Value = vbChecked Then
      Picture1.PSet (xPos, yPos), vbGreen  'visualizza i punti
    End If

    totPoint = totPoint + 1
    Area = (inPoint / totPoint) * PicArea
    lblArea = "Area= " & Area
    lblErr = "Err = " & Area / Sqr(totPoint)

    lblXPos = "xPos =" & xPos
    lblYPos = "yPos =" & yPos
    lblIn = "nIn  = " & inPoint
    lblOut = "nOut = " & outPoint
    lblTot = "nTot = " & totPoint

    DoEvents
    If totPoint = 10000 Then Exit Sub
  Loop
End Sub
```

che ritorna un numero di tipo Long compreso fra Min e Max.

Una distribuzione particolarmente utile (data la sua implicazione nei processi di misura) è quella Gaussiana che, nella sua forma standard, si ottiene dalla somma di 12 numeri generati con la Rnd, cioè

```
Public Function Gauss() as single
Dim Sum as single, i as integer
Sum = 0
For i =1 to 12
Sum = Sum + Rnd
Next i
Gauss =Sum-6
End Function
```

che produce numeri casuali che seguono la distribuzione gaussiana con *valor medio* = 0 e *varianza* = 1. Nel software allegato al presente articolo e scaricabile dal sito ftp.infomedia.it è disponibile un controllo ActiveX che genera numeri a caso che seguono la distribuzione piatta, gaussiana o di Poisson con qualsivoglia intervallo e valor medio (riferirsi alle note d'uso accluse nei file scaricati per il suo corretto utilizzo).

Un primo esempio: "piove a catinelle"

Come primo esempio di simulazione affrontiamo il semplice problema di calcolare la superficie di una figura geometrica con contorni qualsiasi (**Figura 1**). Ci sono molte tecniche per risolvere questo problema ma noi lo affronteremo facendo una simulazione.

Supponiamo che la superficie di cui dobbiamo calcolare l'area sia esposta all'aperto in una giornata di pioggia. Consideriamo un quadrato, sufficientemente grande, che possa contenere tale superficie al suo interno (**Figura 2**). Allora, visto che piove, il numero di gocce d'acqua (distribuite in maniera uniforme) che cadono sul quadrato di riferimento e sulla superficie sotto misura sarà proporzionale alle rispettive aree. Il rapporto fra il numero di gocce cadute sul quadrato e quelle cadute sulla superficie sarà quindi uguale al rapporto fra le loro aree(*). Si intuisce ora qual è la strategia del nostro procedimento: data la figura di area incognita

1. racchiuderla in un quadrato di dimensioni note;
2. estrarre coppie di numeri a caso che definiscono la posizione dei punti del quadrato dove cadono le gocce di pioggia;
3. verificare se tali punti appartengano o no alla figura in esame;
4. conteggiare i due casi separatamente;
5. fare il rapporto fra questi due numeri che sarà uguale al rapporto fra le aree delle due figure.

È da evidenziare che il risultato fornito da questo algoritmo (a differenza di altri metodi utilizzabili) è virtualmente indipendente dal tipo di figura in esame e, sempre per teoremi generali di statistica, è affetto da un errore proporzionale all'inverso della radice quadrata del numero di punti impiegato, cioè err = 1/sqr(N).

La procedura precedente si traduce nel programma riportato nel **Listato 1**.

Ci sono pochi punti importanti da analizzare:

- Nella *Form_Load* dopo l'inizializzazione delle variabili incontriamo la funzione *Randomize* che permette ad ogni avvio del programma di avere una differente sequenza dei numeri casuali generati dalla Rnd. Commentando questa istruzione ad ogni avvio del programma si avrà sempre la stessa sequenza dei punti di caduta della pioggia.
- Nella *Picture1_MouseDown* viene creato un array di punti rappresentativi delle coordinate (X,Y) dei vertici della figura in costruzione. Per ogni nuovo punto inserito (a parte il primo) viene disegnata una linea che ha per estremi il punto corrente e l'ultimo punto precedentemente inserito. Si può inserire qualsivoglia numero di punti.
- Nella *Picture1_DblClick* si interrompe il processo di inserimento e viene disegnato un poligono di colore rosso che ha per vertici i punti inseriti.
- La *cmdCalcola_Click* è il cuore del programma:
  - Nel ciclo *While* vengono estratti due numeri a caso (xPos, yPos) che rappresentano un punto del quadrato su cui cade la goccia di pioggia e, analizzando

(*) Come i più smaliziati in matematica avranno già intuito, la stessa tecnica è applicabile per valutare gli integrali definiti non analiticamente calcolabili. Inoltre è possibile estendere l'algoritmo per calcolare il volume di un ipersolido in un iperspazio ad n dimensioni (con n>2).

il suo colore, viene verificato se esso è esterno o interno al poligono.

- Per mezzo del CheckBox è possibile visualizzare, in verde, il punto estratto. Occorre osservare che la visualizzazione dei punti estratti può portare ad un piccolo errore nel computo dell'area. Infatti, quando il numero di punti è elevato, può accadere che un punto venga generato più volte e quindi il suo colore può dare una indicazione falsata della sua posizione interna o esterna alla figura in esame.

- La parte restante del codice si incarica di mostrare i valori dei parametri più importanti. Da notare che l'unità di misura dell'area è il pixel.

Conclusioni

La computer simulation è un potente strumento di previsione e/o verifica del comportamento di sistemi che possono non essere accessibili o la cui reale costruzione può richiedere ingenti investimenti economici e di "man power".

La simulazione del loro comportamento, al variare dei parametri, può aiutare nella scelta della soluzione migliore.

Bibliografia

[1] H.D. Jung – *"Elaborazione statistica dei dati sperimentali"*, Veschi editore, 1971 [qualsiasi libro di statistica va bene]

Riferimenti

[2] http://msdn.microsoft.com

File compatti e illeggibili: Compressione

Seconda puntata

Crittografia e compressione di file in VB6 utilizzando il Microsoft .NET Framework

di Scott Swigart, Swigart Consulting LLC

In questa serie di due articoli, si vedrà come si possono aggiungere facilmente funzionalità di crittografia e di compressione dati (ZIP) alle applicazioni VB6 esistenti utilizzando il .NET Framework. Benché la crittografia e la compressione possano non sembrare tecnologie correlate, se si pensa ad esse, ciascuna prende un insieme di dati ed esegue una trasformazione si di esso. Nel caso della crittografia, i dati vengono resi illeggibili, mentre con la compressione i dati vengono resi più piccoli. Si vedrà anche che entrambe utilizzano molte delle medesime tecnologie sottostanti.

Introduzione

Sino ad ora, nella serie di articoli VB Fusion [3], è stata mostrata una significativa quantità di funzionalità fornite dal Microsoft .NET Framework, ma si deve anche sapere che è disponibile una enorme quantità di codice open-source di alta qualità che è altrettanto facile da utilizzare nelle proprie applicazioni VB6. Creando dei sottili wrapper di codice, si possono esporre funzionalità di librerie open-source e di terze parti (in origine sviluppate per l'utilizzo nelle applicazioni .NET) per estendere le proprie applicazioni VB6. Siti come www.sourceforge.net e www.gotdotnet.com contengono letteralmente migliaia di progetti open-source che forniscono una gamma estremamente ampia di ulteriori funzionalità.

Alcuni esempi comprendono librerie per la RS232 (la porta seriale) la comunicazione, la manipolazione di flussi RSS, la manipolazione di immagini e di file multimediali, librerie per la rete, librerie FTP, replicazione di file, e migliaia di altre librerie.

Questo articolo si occuperà dell'utilizzo di una di queste librerie open-source per la compressione di file. SharpZipLib è un progetto open-source che

Scott Swigart *fornisce consulenza alle aziende su come utilizzare al meglio l'attuale tecnologia e prepararsi al domani. A tal riguardo, Scott è un orgoglioso collaboratore del sito VB Fusion, dove offre informazioni e strategie di reale utilizzo per gli sviluppatori VB che vogliono realizzare il maggior numero di funzionalità con il minimo sforzo. Scott è anche un Microsoft MVP, ed è coautore di numerosi libri e articoli, ed è contattabile all'indirizzo email scott@swigartconsulting.com.*

fornisce funzionalità di compressione (nel formato zip). Se si desidera, si può liberamente comprendere questa libreria anche in software di tipo close-source, ossia da rivendere.

Prima di affrontare i dettagli dell'utilizzo di SharpZipLib, è bene confrontare come l'I/O su file .NET differisce dall'I/O su file in VB6. Come rapido ripasso, VB6 utilizza delle keyword per controllare l'input/output su file:

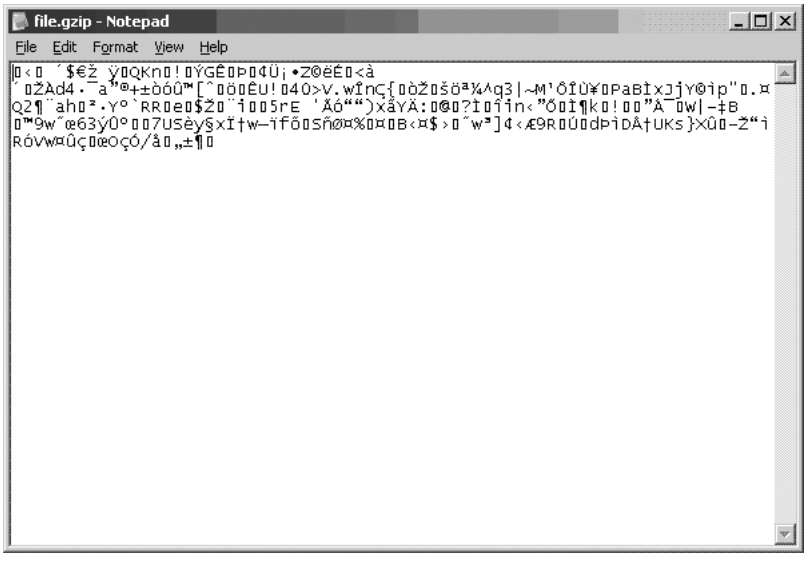


Figura 1 Se si prova ad aprire il file su disco con qualcosa tipo Notepad, si può vedere che è stato scritto in un formato binario compresso

```
' Lettura da un file con Visual Basic 6
Dim dataIn As String
Open someFileName For Input As #1
Do Until Eof(1)
    Line Input #1, dataIn
    Debug.Print dataIn
Loop
Close #1
```

In Visual Basic .NET, invece, si devono utilizzare alcune classi specifiche del .NET Framework per eseguire le operazioni di File I/O:

```
' File I/O con Visual Basic .NET
Dim dataIn As String
Dim sw As New StreamWriter("C:\log.txt")
sw.WriteLine("This is a test")
sw.Close()
```

Con VB6, si devono gestire gli handle dei file, e si devono conoscere le relative keyword del linguaggio necessarie per operare con questi handle di file.

Con Visual Basic .NET, la classe StreamWriter racchiude tutte le funzionalità associate ai file. Non è necessario gestire gli handle di file, e le operazioni su file sono più reperibili, poiché IntelliSense evidenzia tutte le operazioni disponibili (WriteLine, ecc.) quando si lavora con i file.

Ma il vantaggio reale di questi oggetti (noti anche come *stream*) è che permettono di impilare le funzionalità di I/O su file.

Nel primo articolo, si è visto come la crittografia fosse supportata semplicemente aggiungendo un CryptoStream allo stack degli stream di I/O:

```
'  Stack di stream per abilitare la crittografia in VB.NET
fs = New FileStream(fileName, FileMode.Create,
                                        FileAccess.Write)
cs = New CryptoStream(fs, rc2.CreateEncryptor(),
                                    CryptoStreamMode.Write)
sw = New StreamWriter(cs)
```

Pertanto, qualsiasi cosa scritta su StreamWriter viene poi inviata attraverso CyrptoStream, dove viene crittografata, e quindi inviata al FileStream dove viene scritta su disco.

Compressione

La compressione funziona esattamente allo stesso modo:

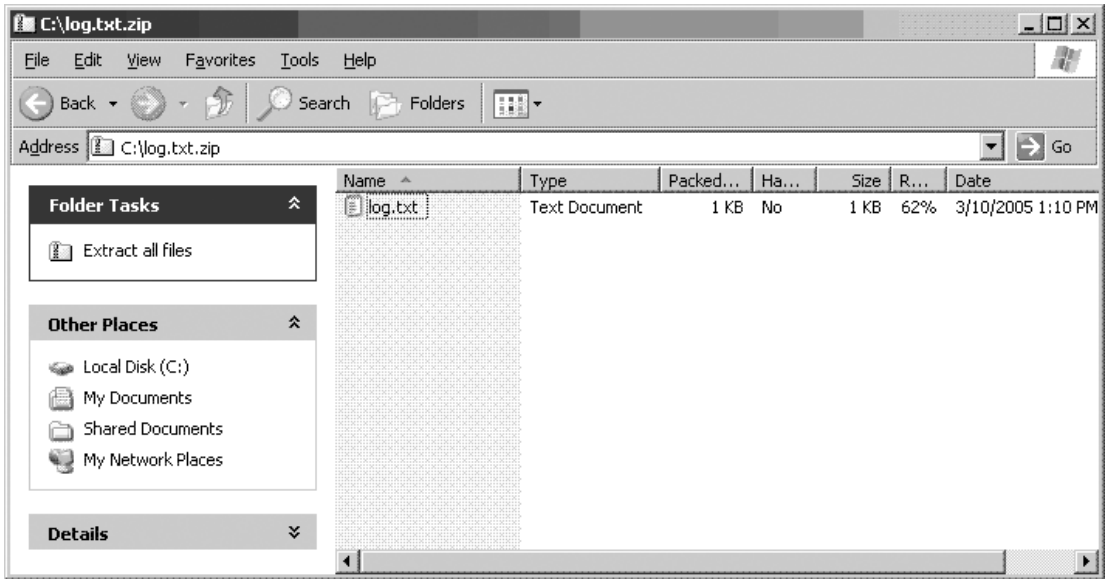


Figura 2 Il codice richiede all'utente un file da comprimere in formato zip, e poi utilizza il wrapper per "zippare" il file. Il risultato è un archivio Zip leggibile da Windows XP o con altri lettori del formato Zip

```
'  Stack di stream per abilitare la compressione in VB.NET
Dim fs = New FileStream(fileName, FileMode.Create,
                                        FileAccess.Write)
zs = New GZipOutputStream(fs)
sw = New StreamWriter(zs)
```

Come si può vedere, la compressione si compone con l'I/O su file esattamente allo stesso modo della crittografia, il che ha senso. Sia la crittografia sia la compressione sono in realtà una trasformazione dei dati. Nella crittografia, i dati vengono trasformati in un formato non leggibile senza una chiave. Con la compressione, i dati vengono trasformati in modo da richiedere meno byte per rappresentare gli stessi dati. L'intera classe .NET che permette di scrivere informazioni su un file compresso è riportata nel **Listato 1**.

È giunto il momento di utilizzarla da una applicazione VB6, per scrivere informazioni su un file compresso. Grazie a questo wrapper, il codice VB6 è banale:

```
'  Codice VB6 per scrivere su un file compresso
Dim e As wrappers.GZipFileWriterWrapper
Set e = New wrappers.GZipFileWriterWrapper
e.Open "c:\file.gzip"
e.WriteLine (txtMessage)
e.Close

txtMessage = ""
MsgBox "Compressed and Saved"
```

Come si può vedere, basta utilizzare il wrapper per aprire il file, scrivere quanto si vuole, e quindi chiuderlo al termine.

Se si prova ad aprire il file su disco con qualcosa tipo Notepad, si può vedere che è stato scritto in un formato binario compresso (**Figura 1**).

Tuttavia, utilizzando una applicazione VB6, si può facilmente utilizzare il wrapper per rileggere le informazioni compresse:

```
'  Lettura del file compresso con VB6
e.Open "c:\file.gzip"
While Not e.EOF
    txtMessage = txtMessage & e.ReadLine & vbCrLf
    ' Si potrebbe anche utilizzare semplicemente
    ' e.ReadToEnd per leggere l'intero file.
Wend
e.Close
```

"Zippare" file e cartelle

Il tipo di compressione mostrata sinora permette di comprimere le informazioni prima di memorizzarle in un file. Tuttavia, capita spesso di voler creare un archivio Zip, ossia un file Zip che contiene uno o più file. Questa funzionalità viene fornita dal wrapper attorno alla SharpZipLib, il cui codice è riportato nel **Listato 2**.

Listato 1 Classe VB.NET necessaria per la scrittura su un file compresso

```
Public Class GZipFileWriterWrapper
Private fs As FileStream
Private zs As GZipOutputStream
Private sw As StreamWriter

Public Sub Open(ByVal fileName As String)
    fs = New FileStream(fileName, FileMode.Create, FileAccess.Write)
    zs = New GZipOutputStream(fs)
    sw = New StreamWriter(zs)
End Sub

Public Sub WriteLine(ByVal value As String)
    sw.WriteLine(value)
End Sub

Public Sub Write(ByVal value As String)
    sw.Write(value)
End Sub

Public Sub WriteBytes(ByRef bytes() As Byte)
    zs.Write(bytes, 0, bytes.Length)
End Sub

Public Sub Close()
    On Error Resume Next
    If Not sw Is Nothing Then sw.Close()
    If Not zs Is Nothing Then zs.Close()
    If Not fs Is Nothing Then fs.Close()
End Sub
End Class
```

Qui, la classe ZipEntry viene utilizzata per scrivere informazioni sul file che verrà aggiunto all'archivio. Una volta scritta la ZipEntry sullo ZipOutputStream, viene scritto il contenuto effettivo del file. Il risultato è un archivio Zip che può essere aperto utilizzando WinZip, o la funzionalità Zip nativa di Windows XP. Realizzato il wrapper, per "zippare" un file da VB6 bastano poche righe:

```
'  Zippare un file da VB6
CommonDialog1.ShowOpen
Dim source As String
source = CommonDialog1.FileName

Dim e As wrappers.FileZipWrapper
Set e = New wrappers.FileZipWrapper
e.ZipFile source, source & ".zip"
```

Il codice richiede all'utente un file da comprimere in formato zip, e poi utilizza il wrapper per "zippare" il file. Il risultato è un archivio Zip leggibile da Windows XP o con altri lettori del formato Zip (**Figura 2**).

Infine, può essere necessario creare un file Zip di un'intera cartella, e anche per fare ciò viene fornito un wrapper:

```
'  Codice VB6 per creare lo zip di una intera cartella
Dim e As wrappers.FileZipWrapper
Set e = New wrappers.FileZipWrapper

e.ZipFolder App.Path, App.Path & "\..\file.zip"
```

Eseguire l'esempio

Nella prima puntata dell'articolo, abbiamo creato un esempio che permette di esercitarsi con la funzionalità di crittografia. Ho modificato questo esempio per utilizzare i wrapper di compressione forniti in questo articolo. Per utilizzare l'esempio può essere necessario:

- Scaricare e installare il Microsoft .NET Framework SDK (se si è installato Visual Studio .NET, si può saltare questo passo);

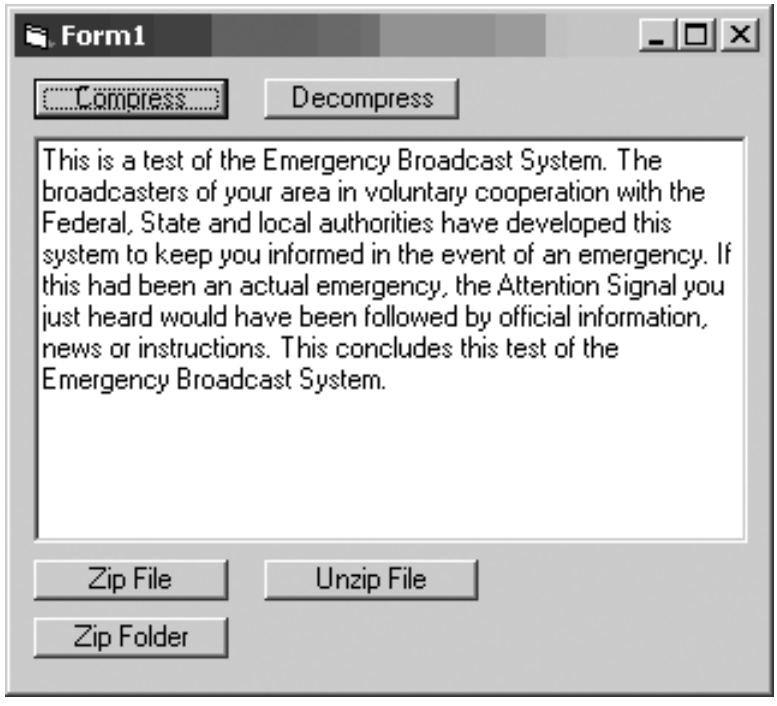


Figura 3 F5 fa eseguire l'applicazione d'esempio

- Scaricare il codice allegato all'articolo [2];
- Eseguire il file "Build and Register.bat" incluso in questo codice. Così facendo si esegue il build del componente di crittografia e lo si registra in modo da poterlo utilizzare dalla propria applicazione VB6;
- Nel codice dell'articolo, aprire il progetto VB6 presente nella cartella "Compression";
- Premere F5 per eseguire l'applicazione d'esempio (**Figura 3**).

Cliccando su "Compress", il testo nella textbox verrà scritto su un file utilizzando l'algoritmo di compressione GZip. Cliccando su "Decompress", il contenuto verrà decodificato e visualizzato.

Cliccando su "Zip File", verrà richiesto di scegliere un file, e verrà creato un archivio .zip che contiene il file. "Unzip file" estrarrà il file dall'archivio. Infine, "Zip Folder" comprime il contenuto di una intera cartella.

Per utilizzare questa funzionalità dalla propria applicazione, basta eseguire "Build and Register.bat", e poi aggiungere un riferimento a "wrappers.tlb" presente nella cartella "c:\tlbs".

Conclusioni

Microsoft .NET Framework comprende vaste funzionalità per molti scenari, ma la comunità open-source ha prodotto una quantità sbalorditiva di ulteriori funzionalità che possono essere avvolte da un wrapper e utilizzate altrettanto facilmente da VB6.

In un certo senso, non si tratta di nulla di nuovo. VB6 ha sempre goduto di una grande comunità che è stata generosa nel fornire esempi di codice. Talvolta si trattava di frammenti, e talvolta erano complete librerie.

Tuttavia, con VB6, si era limitati a codice sviluppato e condiviso da altri sviluppatori VB6. Con .NET, sia gli sviluppatori VB.NET sia C# possono produrre codice che la propria applicazione può utilizzare (SharpZipLib è stato scritto in C#, ma ciò non lo rende più difficile da avvolgere e utilizzare da VB6 di quanto lo sarebbe se fosse scritto in VB.NET).

Inoltre, un numero cospicuo di librerie scritte in origine in C++ e Java è stato portato in .NET.

Ciò significa che si hanno a propria disposizione migliaia di ulteriori librerie, mol-

Listato 2 Wrapper VB.NET per creare un archivio Zip

```
Public Sub ZipFile(ByVal source As String, ByVal-
                                    dest As String)

    Dim entry As New ZipEntry(Path.GetFileName(source))
    Dim fi As New FileInfo(source)
    entry.ExternalFileAttributes = fi.Attributes
    entry.Size = fi.Length

    Dim input As FileStream = File.OpenRead(source)
    Dim output As New ZipOutputStream(File.Create(dest))

    output.PutNextEntry(entry)

    Dim buffer(8191) As Byte
    Dim len As Integer
    Do
        len = input.Read(buffer, 0, buffer.Length)
        If len > 0 Then
            output.Write(buffer, 0, len)
        End If
    Loop Until len = 0
    output.Close()
    input.Close()
End Sub
```

te delle quali sono state portate da librerie sviluppate, migliorate e utilizzate per molti anni nelle applicazioni di produzione.

Come ha mostrato la serie di articoli VB Fusion [3], si può sfruttare tutto ciò che .NET fornisce pur preservando il proprio investimento pregresso in codice VB6.

Come ha mostrato specificamente questo articolo, non si è in alcun modo limitati al solo .NET framework e alle librerie fornite da Microsoft.

La florida comunità .NET è a vostra disposizione.

Riferimenti

[1] http://www.icsharpcode.net/OpenSource/SharpZipLib/Default.aspx/

[2] ftp.infmedia.it/pub/VBJ

[3] http://msdn.microsoft.com/vbrun/vbfusion

Creare Add-in per VBE in Microsoft Access

Utilizzare strumenti che svolgono per noi i compiti ripetitivi, permette di concentrarsi sui concetti della programmazione ed essere più produttivi.

di Gionata Aladino Canova

Nell'articolo "Creare Add-in per Microsoft Access" (VBJ n. 62) abbiamo visto come realizzare una barra di strumenti che si integra nell'ambiente di Access, per automatizzare certe operazioni. In questo articolo vedremo come aggiungere al menu dell'editor di VBA (VBE *Visual Basic Editor*) dei comandi personalizzati che ci aiutino nella stesura del codice.

Analizzeremo poi un add-in che utilizzo comunemente per sviluppare, visibile in **Figura 1**. Tale strumento è allegato all'articolo e scaricabile dal sito FTP di Infomedia ed il suo utilizzo è gratuito; è fornito completo di sorgenti che possono essere modificati e ridistribuiti a condizione di non rimuovere le indicazioni sull'autore. Per la comprensione dell'articolo serve una discreta padronanza dell'ambiente di Access e la conoscenza del VBA; i concetti esposti sono validi da Microsoft Access 2000 in poi.

Le complessità nascoste

La programmazione in un ambiente visuale è un problema complesso. Chi ha visto cosa ci voleva in C per creare una finestra in Windows, sa a cosa mi riferisco. Tanti sistemi moderni, Access compreso, consentono di fare molte cose in modo semplice, nascondendo le complessità sottostanti. Creare un add-in in Access è sicuramente più facile che creare un add-in per il VBE, poiché quest'ultimo nasconde un po' meno cose. Ad esempio, personalizzare i menu di VBE è più complicato che personalizzare i menu di Access. Questo perché VBE è basato sul modello di Visual Studio e non su quello di Office.

Progettare un add-in

La prima operazione da fare per progettare un add-in è analizzare quali operazioni desideriamo implementare. Da questa lista otterremo una serie di funzioni che potranno logicamente essere raggruppate. Ad esempio, le funzioni che trattano la generazione di codice per la gestione di recordset potranno essere organizzate tutte in un'unica form. Creata la lista delle

Gionata Aladino Canova *programma dai bei tempi del Sinclair Spectrum. Laureato in Informatica, è titolare della Aladino Informatica e socio di TDE Informatica srl. Sviluppa con Microsoft Access, VB.NET e realizza siti in ASP/ASP .NET. Può essere contattato a g.canova@tdeinformatica.it.*

form e delle eventuali funzioni da chiamare direttamente, sarà facile progettare l'interfaccia utente dell'add-in, ossia stabilire quali menu o barre degli strumenti vadano create o quali siano gli eventi da intercettare. Nel wizard di esempio vedremo la creazione di un menu e l'intercettazione dell'evento di compilazione. La parte finale della progettazione sarà il suddividere quali funzioni vadano create in quali moduli. La creazione dell'add-in si svilupperà nella stesura del codice che realizza le funzioni e nella parte che crea l'interfaccia con l'utente.

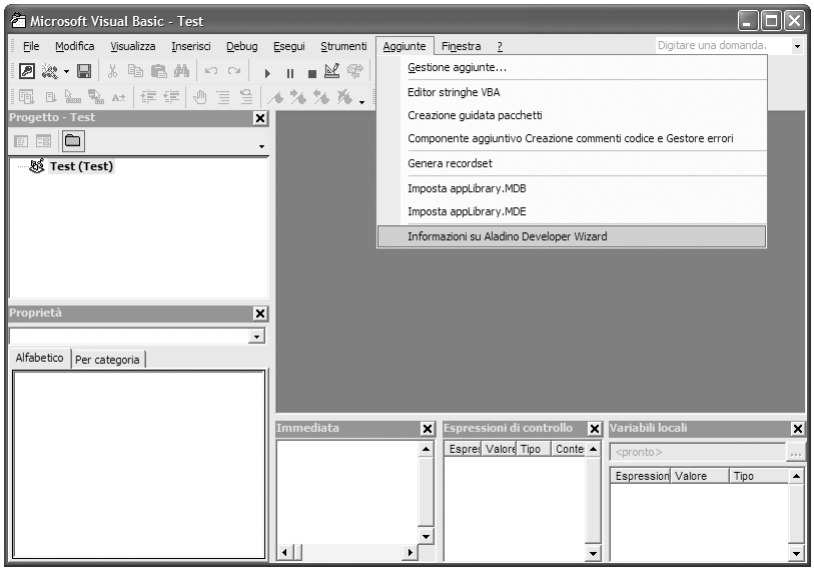


Figura 1 Il menu del VBE personalizzato tramite un add-in

Costruire lo scheletro dell'add-in

Iniziamo aprendo Access, cliccando sulla caption e spostandoci nell'editor VBE (tasto rapido ALT+F11). La finestra è desolantemente vuota. Se la finestra Progetto non è visualizzata, dal menu *Visualizza* selezioniamo *Gestione Progetti* (CTRL+R). Dal menu *File*, selezioniamo l'opzione *Nuovo Progetto* e poi *Progetto Aggiunta*. Nella finestra che compare, compiliamo le voci *Nome aggiunta visualizzato* inserendo *VBJ Wizard* e *Descrizione aggiunta* con un commento a piacere, poi impostiamo la voce *Applicazione* a *Visual Basic for Application IDE*. In futuro, per far ricomparire questa finestra, dalla finestra Gestione Progetti, andremo nella sezione Finestre di progettazione e faremo doppio click sulla voce che, di default, si chiama *AddInDesigner1*. Salviamo il file con il nome *VBJAddIn*. Nei riferimenti (Strumenti – Riferimenti) dovremo aggiungere:

Listato 1 Codice per le operazioni da implementare

```
Option Explicit

Public oHostApplication As Object
Public oAddIn As Object
Public conNomeApp As String
Public Const conVersione = "1.0 del 12-02-2006"
Public Const conTitoloApp = "VBJ Wizard"

Function wizCompila()
    MsgBox "Stai compilando!", vbInformation, "VBJAddIn"
End Function

Sub wizInfo()
    MsgBox "VBJ Wizard ver." & conVersione, vbInformation, "VBJAddIn"
End Sub

Sub wizNumeroCasuale()
    MsgBox "Il primo numero che mi viene in mente è: " & Int(Rnd() * 1000)
End Sub
```

- Microsoft Access 10.0 Object Library (per l'oggetto VBE)
- Microsoft Office 10.0 Object Library (per la gestione delle barre degli strumenti, es. CommandBarButton)
- Microsoft Visual Basic For Applications Extensibility 5.3 (per intercettare gli eventi delle Barre di VBE)

Aggiungiamo il modulo che conterrà le operazioni da implementare. Inseriamo il codice contenuto nel **Listato 1**. La funzione *wizCompila* ci avviserà che l'utente sta compilando. In un progetto reale potrebbe incrementare un numero di build (funzione mai gestita in Access!). La funzione *wizInfo* fornisce informazioni sul wizard, mentre *wizNumeroCasuale* visualizza una message-box che stampa un numero a caso. Dal riquadro Proprietà, impostiamo il nome del modulo a *modMioCodice*. Dobbiamo adesso costruire l'interfaccia con l'utente.

Facciamo un click destro su *AddInDesigner1* e selezioniamo Visualizza codice.

Inseriamo il codice contenuto nel **Listato 2**. Se proviamo a compilare, non dovremmo ottenere errori. Prima di analizzare il codice del **Listato 2**, collaudiamo l'add-in.

Collaudare l'add-in

Dopo essere riusciti a compilare senza errori, la sequenza per collaudare l'add-in è:

- *Esegui/Esegui Progetto/Ok* (c'è anche il corrispondente pulsante sulla barra Standard.
- *Aggiunte/Gestione Aggiunte*. Fare doppio click sul nostro add-in (nell'esempio, VBJ Wizard), per caricarlo, poi premere *Ok*.
- Nel menu *Aggiunte* troveremo le due opzioni da noi create. Per provarle è sufficiente selezionarle. Per provare la compilazione, apriamo un database vuoto, aggiungiamo un modulo ed una routine anch'essa vuota. Selezioniamo poi *Debug/Compila*. Se tutto funziona, una message-box ci avviserà che stiamo compilando.
- Finito il test, ritorniamo in *Aggiunte/Gestione Aggiunte* e facciamo doppio click sul nostro add-in (nell'esempio, VBJ Wizard), per scaricarlo. Questa operazione è necessaria per ripulire i menu di VBE dalle voci che vi abbiamo aggiunto. Interrompendo semplicemente il progetto, non viene eseguita la routine di pulizia.
- Selezioniamo *Esegui/Interrompi progetto*

Se saltiamo lo scaricamento dell'Add-in, oppure abbiamo commesso errori nel codice, il menu potrebbe rimanere sporco.

Per rimediare, se non abbiamo altre personalizzazioni manuali, è sufficiente andare in modifica delle barre dei menu, selezionare la *Barra dei menu* e selezionare la funzione *Reimposta*.

Come funziona l'add-in

Vediamo adesso in dettaglio il funzionamento dell'add-in.

Le operazioni che esso compie sono volutamente banali, per potersi concentrare solo sul codice di base necessario per costruire l'add-in.

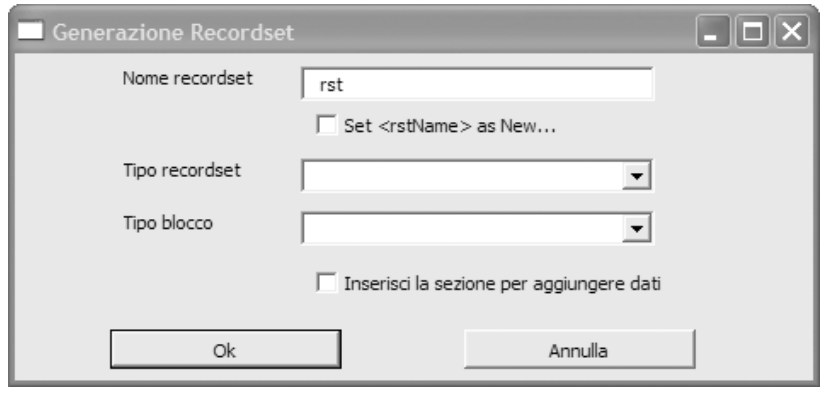


Figura 2 La form per la generazione di recordset, contenuta nell' "Aladino Developer Wizard"

Le dichiarazioni

```
Private WithEvents mnuItem1 As Office.CommandBarButton
Private WithEvents mnuItem2 As Office.CommandBarButton
```

impostano due riferimenti per le due voci di menu che andremo a creare nella routine *CreaMenu*. La parola chiave *WithEvents* indica al compilatore che, di quei due oggetti, vogliamo gestirne gli eventi. Il modo per gestire gli eventi è dichiarare una routine, come Access fa in maniera automatica quando gli chiediamo di gestire un evento di un oggetto, con la sintassi

```
Private Sub <nomeVariabile>_<NomeEvento>()[parametri])
```

come si vede nell'esempio seguente, dove viene gestito l'evento Click dell'oggetto mnuItem1.

```
Private Sub mnuItem1_Click(ByVal Ctrl As
        Office.CommandBarButton, CancelDefault As Boolean)
```

Queste routine devono stare nel modulo di codice associato all'AddInDesigner e, ovviamente, possono contenere chiamate a routine contenute in moduli esterni, come succede in quella appena vista.

La dichiarazione

```
Private WithEvents cCompila As CommandBarEvents
```

ci permette invece di gestire l'evento legato all'opzione già esistente *Compila*. L'associazione tra la variabile *cCompila* ed il pulsante di compilazione viene effettuata all'interno della routine *AddinInstance_OnConnection*, dalle righe

```
' Intercetta l'evento di compilazione (Debug / Compila)
Dim c As CommandBarControl
Set c = VBE.CommandBars(1).Controls(5).Controls(1)
Set cCompila = VBE.Events.CommandBarEvents(c)
```

All'avvio dell'add-in, viene eseguita la routine *AddinInstance_OnConnection*, nella quale creiamo i menu ed impostiamo l'intercettazione dell'evento di compilazione, come visto sopra. Alla chiusura dell'add-in, tramite la routine *AddinInstance_OnDisconnection* provvediamo all'eliminazione delle voci di menu.

La routine *CreaMenu* è abbastanza semplice, se si ha un'idea di come sia il modello ad oggetti delle barre di menu. Impostato il riferimento in *cbrMenu*, alla barra *Add-Ins* di VBE, per evitare duplicazioni di opzioni, proviamo a cercare se esiste già l'opzione che vorremmo creare, altrimenti la aggiun-

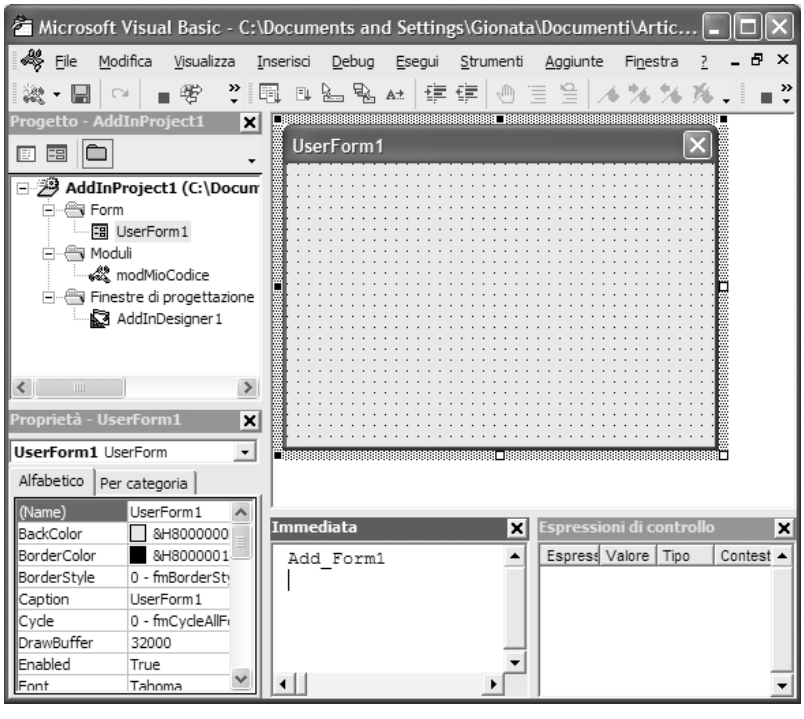


Figura 3 Aggiunta di una UserForm all'add-in

Listato 2 Il corpo dell'Add-in (continua...)

```
  Option Explicit

Private WithEvents mnuItem1 As Office.CommandBarButton
Private WithEvents mnuItem2 As Office.CommandBarButton

Private WithEvents cCompila As CommandBarEvents

Private Sub AddinInstance_OnConnection(ByVal Application As Object, ByVal Connect
                    Mode As AddInDesignerObjects.ext_ConnectMode, ByVal AddInInst As Object, custom() As Variant)
    'Memorizzo le referenze all'avvio
    Set oHostApplication = Application
    Set oAddIn = AddInInst

    conNomeApp = AddInInst.ProgId

    'Crea le opzioni aggiuntive nel menu di VBE
    CreaMenu (AddInInst.ProgId)

    ' Intercetta l'evento di compilazione (Debug / Compila)
    Dim c As CommandBarControl
    Set c = VBE.CommandBars(1).Controls(5).Controls(1)
    Set cCompila = VBE.Events.CommandBarEvents(c)
End Sub

Private Sub AddinInstance_OnDisconnection(ByVal RemoveMode As AddInDesignerObjects.ext_DisconnectMode, custom()
                                                                                                    As Variant)
    'Rimuove le opzioni dal menu
    RimuoviMenu
    Set oAddIn = Nothing
    Set oHostApplication = Nothing
End Sub

Sub CreaMenu(strProgID As String)
    ' Crea un elemento nel menu e lo collega all'addin.
    Dim cbrMenu As Office.CommandBar
    Dim ctlBtnAddIn As Office.CommandBarButton
    Dim strKey As String

    ' Stabilisce un riferimento al menu Add-Ins
    Set cbrMenu = VBE.CommandBars("Add-Ins")

    ' Prova a cercare il pulsante "VBJ - Numero casuale"
    strKey = "VBJWizNumCasuale"
    Set mnuItem1 = cbrMenu.FindControl(Tag:=strKey)

    If mnuItem1 Is Nothing Then
        ' Aggiunge il pulsante
        Set mnuItem1 = cbrMenu.Controls.Add(msoControlButton, , strKey, , True)

        With mnuItem1
            .Caption = "VBJ - Numero casuale"
            .Tag = strKey
            .Style = msoButtonCaption
            .FaceId = 634
            .OnAction = "!<" & strProgID & ">"
            .BeginGroup = True
        End With
    End If

    ' Prova a cercare il pulsante "Informazioni su..."
    strKey = "VBJWizInfo"
    Set mnuItem2 = cbrMenu.FindControl(Tag:=strKey)
```

Listato 2 (...fine)

```
    If mnuItem2 Is Nothing Then
        ' Aggiunge il pulsante
        Set mnuItem2 = cbrMenu.Controls.Add(msoControlButton, , strKey, , True)

        With mnuItem2
            .Caption = "Informazioni su VBJ Developer Wizard"
            .Tag = strKey
            .Style = msoButtonCaption
            .FaceId = 634
            .OnAction = "!<" & strProgID & ">"
            .BeginGroup = True
        End With
    End If

End Sub

Sub RimuoviMenu()
    On Error Resume Next
    VBE.CommandBars("Add-Ins").Controls("VBJ - Numero casuale").Delete
    VBE.CommandBars("Add-Ins").Controls("Informazioni su VBJ Developer Wizard").Delete
End Sub


' Qui inizia la sezione che gestisce gli eventi
Private Sub mnuItem1_Click(ByVal Ctrl As Office.CommandBarButton, CancelDefault As Boolean)
    wizNumeroCasuale
End Sub

Private Sub mnuItem2_Click(ByVal Ctrl As Office.CommandBarButton, CancelDefault As Boolean)
    wizInfo
End Sub

Private Sub cCompila_Click(ByVal CommandBarControl As Object, Handled As Boolean, CancelDefault As Boolean)
    ' Viene gestito l'evento di compilazione
    wizCompila
End Sub
```

giamo. L'operazione viene ripetuta tante volte quante sono le voci da inserire nel menu. La routine *RimuoviMenu* è molto semplice, limitandosi solo a tentare di cancellare le voci create in precedenza. In pratica, avendo la lista delle opzioni che vogliamo creare, è necessario dichiarare n variabili come *mnuItemX* all'inizio della routine, inserire la creazione e la distruzione delle opzioni in *CreaMenu* e *RimuoviMenu* ed infine creare le corrispondenti routine di intercettazione degli eventi. Il codice del **Listato 1** non ha bisogno di commenti, essendo un esempio ridotto all'osso.

Un Add-in reale

Nel codice allegato trovate anche un add-in completo, che uso per sviluppare. Il file si chiama *AladinoDevWizard.vba* e contiene, oltre a tutta l'infrastruttura vista fino ad ora, una form per generare recordset, visibile in **Figura 2**.

> Vedremo come aggiungere comandi personalizzati al menu dell'editor di VBA

Analizzando la sub *wizGeneraRecordSet* troviamo del codice la cui analisi esula da questo articolo, ma che merita di essere studiato.

La routine, come primo compito, controlla di essere stata chiamata con il cursore puntato in VBE all'interno di una definizione di sub o di function. Poi, dopo aver visualizzato la form e letto i relativi dati, costruisce una stringa che contiene il codice generato. La stringa viene inserita nella posizione in cui si trova il cursore, dalla riga

```
VBE.ActiveCodePane.CodeModule.InsertLines lngStartLine,
                                                strCode
```

Utilizzando come base questo codice, possiamo costruire una form che ci chieda l'input che vogliamo, generare il codice corrispondente ed inserirlo nel progetto corrente. Per chi non ha mai lavorato con il modello ad oggetti di VBE, consiglio di cercare esempi in rete e di leggere [1] e [2].

> La prima operazione da fare per progettare un add-in è analizzare quali operazioni desideriamo implementare

Esistono metodi per aggiungere sub e function, spostarsi all'interno del codice e fare altre operazioni in maniera molto semplice. Nell'add-in è inclusa una UserForm. A differenza di Word ed Excel, pare che in Access non ci sia un modo immediato per aggiungerla. Probabilmente perché si pensa di utilizzare le form di Access. Per aggirare il problema, ci sono due modi. Possiamo personalizzare la barra dei menu aggiungendo il comando *UserForm*, nel menu *Inserisci*. Oppure possiamo inserire in un modulo il codice seguente che, richiamato dalla finestra di debug, crea una UserForm.

```
Sub Add_Form1()
    ' Declare a variable to hold the UserForm.
    Dim x As Object
    ' Create a new UserForm. You can use this new VBCompo-
                                    nent object to manipulate the User Form.
        Set x = Application.VBE.ActiveVBProject.
                                    VBComponents.Add (vbext_ct_MSForm)
End Sub
```

Il risultato è visibile in **Figura 3**.

Installazione e distribuzione

Per distribuire il nostro add-in, dovremo prima creare la DLL tramite l'opzione File/CreaVBJAddIn.DLL, poi selezioniamo *Aggiunte/Creazione guidata pacchetti* (disponibile con la versione Developer di Office). Seguendo le istruzioni, verrà creato un pacchetto che consentirà la distribuzione della nostra DLL, tramite un normale programma di setup.

Conclusioni

La stesura di un Add-in per VBE è un'operazione non banale, ma fattibile tranquillamente da chi programma tutti i giorni. È importante, nel lavoro quotidiano, capire se scriviamo spesso parti di codice ripetitive, perché in questo caso si può trarre parecchio beneficio creando un add-in personalizzato.

Per funzioni standard tipo conteggio di righe, ordinamento di procedure o simili, conviene invece fare ricerche in rete, poiché si trovano add-in già pronti e collaudati, spesso gratuiti, veramente potenti. Un esempio è *VBA MZ-Tools 3.0* [3].

Riferimenti

[1] *Programming the Microsoft Office Visual Basic Editor*
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnofftalk/html/office07042002.asp

[2] *Before You Create a VB Add-In, Learn the IDE's Object Model*
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dninvb00/html/IDEObjectModel.asp

[3] *MZ-Tools 3.0 for Visual Basic 6.0, 5.0 and VBA*
http://www.mztools.com/v3/mztools3.htm

Una nuova rubrica su Computer Programming!

ordina la tua copia su www.infomedia.it

Nanotecnologia, domotica, robotica, computer quantistici, informatica applicata alla medicina, ai motori, qubit, realtà virtuale ed altro ancora!
Argomenti futuri e futuristici discussi
dalla frontiera che separa la fantascienza dalla realtà.

L'unità elementare d'informazione, il bit, sarà il minimo comune multiplo di ogni argomento e per riflesso anche l'informatica considerata nelle sue "macro" divisioni; software ed hardware.
Si passerà dalla descrizione di recenti "device" applicati alle strutture più disparate (anche il corpo umano) allo studio del nuovo software, da microchip sempre più "micro" a protocolli che alla logica binaria applicano elementi di fisica moderna. L'attenzione sarà focalizzata sui concetti, sulle idee, motore di ogni innovazione.

Ai confini del bit include anche quegli argomenti non proprio informatici ma che con tale disciplina sono legati a filo diretto, come approfondimenti sulla legge del diritto d'autore per prodotti software, sui Digital Right Management (DRM) e su quale vasto mondo oggi abbraccia il termine "informatica"!

E poi si immaginerà il futuro, cocktail di silicio e pensiero, byte e... arseniuro di gallio.

Analisi degli accessi ad un sito web

Prima puntata

Tra i mezzi di comunicazione di massa il web è quello che meglio si presta ad essere misurato; ma quanto sono affidabili queste rilevazioni? E qual è il loro valore?

di Stefano Savo

In questa serie di due articoli cercheremo di rispondere a queste domande facendo il punto sullo stato dell'arte. Riepilogheremo i motivi per i quali è utile rilevare gli accessi ad un sito web, analizzeremo cosa è utile e cosa è possibile misurare, e quali sono i gradi di certezza di questi dati. Infine analizzeremo quali sono i metodi più usati dai numerosi software esistenti e come orientarsi nel mare di offerte, che va da prodotti open source e gratuiti a servizi dal costo di decine di migliaia di euro all'anno.

Perchè analizzare il traffico

Cominciamo a vedere quali sono i motivi per i quali è utile analizzare il traffico di un sito web.

Conoscere i visitatori del sito vuol dire sapere da dove vengono, cosa cercano, ma anche come cercano quello che vogliono. Questa conoscenza ci aiuta a disegnare un sito fruibile e di successo. Un sito di facile e di veloce fruizione non stanca i suoi visitatori e ne attira di nuovi. Non è semplice avere un'idea precisa di quello che sono e di quello che vogliono gli utenti (molto spesso non lo sanno nemmeno loro) ma è importante cercare di distinguere quello che succede veramente sul nostro sito da quello che noi supponiamo che succeda. Un sistema di rilevazione degli accessi al sito ci consente di misurare alcuni dati oggettivi, che poi andranno interpretati nella maniera più opportuna.

Per cominciare è sicuramente importante capire quali sono le pagine più viste del sito. È anche utile raggruppare tutte quelle pagine che rivelano un interesse in un certo settore e misurare quindi quali sono i settori di maggior interesse.

Una volta individuate le risorse più apprezzate, rilevando i percorsi che vengono seguiti dai nostri utenti possiamo capire se i visitatori raggiungono facilmente quello che cercano e possiamo pensare di accorciare il cammino necessario a raggiungere le risorse più apprezzate del sito. D'altra parte è possibile che alcuni dei servizi che il nostro sito offre non risultino particolarmente apprezzati. In questo caso o rivediamo la nostre convinzioni sugli utenti o dobbiamo pensare che

Stefano Savo *è ingegnere informatico e lavora da più di cinque anni nella analisi, progettazione e sviluppo di web application. Tra i suoi interessi: Information Architecture e Web Semantico.*

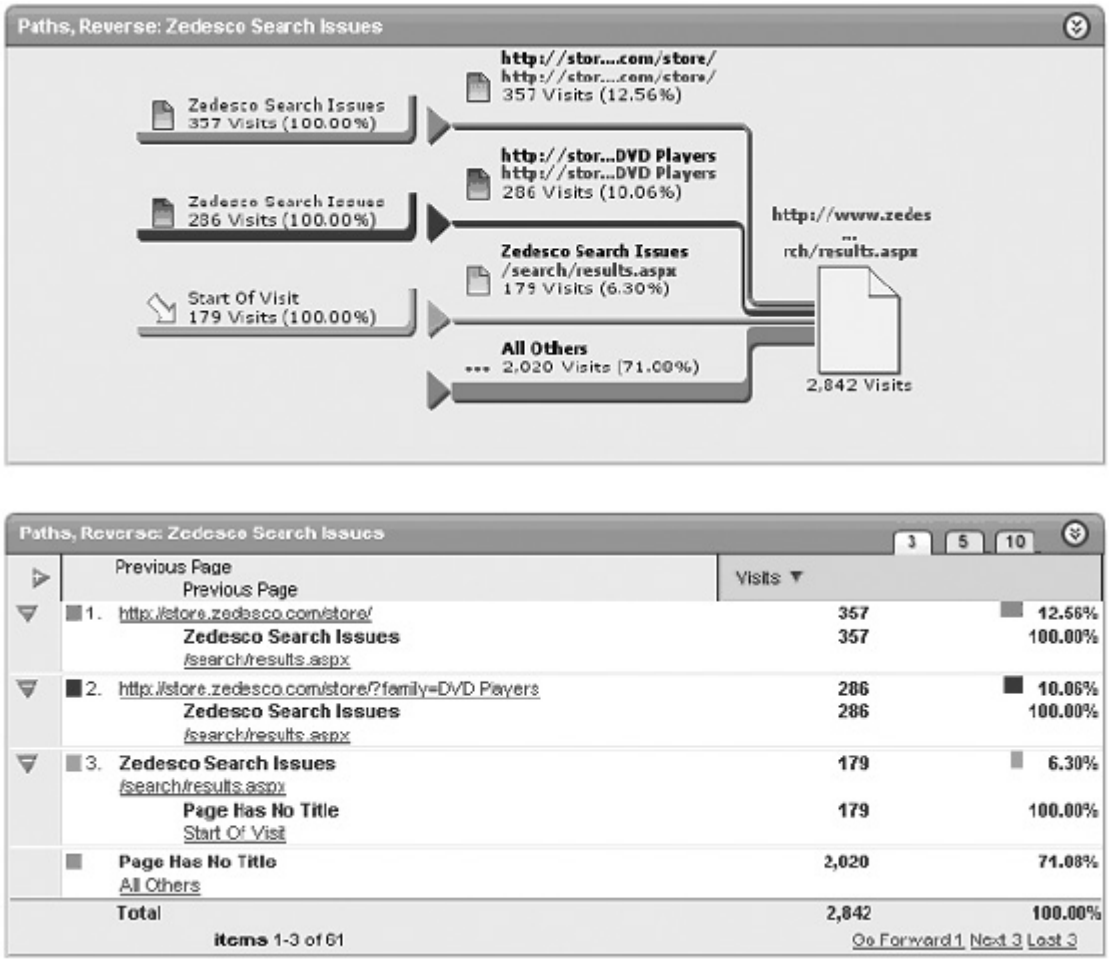


Figura 1 Path backwards (WebTrends)

il servizio non sia usato perché non è sufficientemente noto o in evidenza. Nel primo caso dovremo rivedere il servizio o addirittura eliminarlo, mentre nel secondo dovremo cercare di comunicare ai visitatori l'esistenza del servizio e le sue qualità. Questa esigenza di comunicazione di novità agli utenti è assai comune nelle attività online che sono sempre in evoluzione: si può ad esempio voler comunicare la nascita di un nuovo prodotto o la disponibilità di una particolare offerta commerciale. Un sistema di analisi del traffico del sito ci aiuta a individuare quali sono gli spazi più opportuni per effettuare questa comunicazione. Spesso si pensa che la posizione più naturale per annunci di questo tipo sia la *home page*. Questo è spesso vero, ma dobbiamo evitare che la home page sia sovraccarica di informazioni e ricordare che non tutti i visitatori passano dalla home page: i visitatori regolari hanno probabilmente dei bookmark, mentre quelli che provengono dai motori di ricerca finiscono probabilmente su altre pagine. Misurando gli accessi possiamo quantificare questi diversi tipi di accesso al sito. Se verifichiamo che un buon numero di utenti segue un determinato percorso e accede a particolari risorse, possiamo provare a guidare questi utenti verso informazioni e servizi affini ai quali non avevano pensato. Introdurremo cioè dei link o dei banner ad

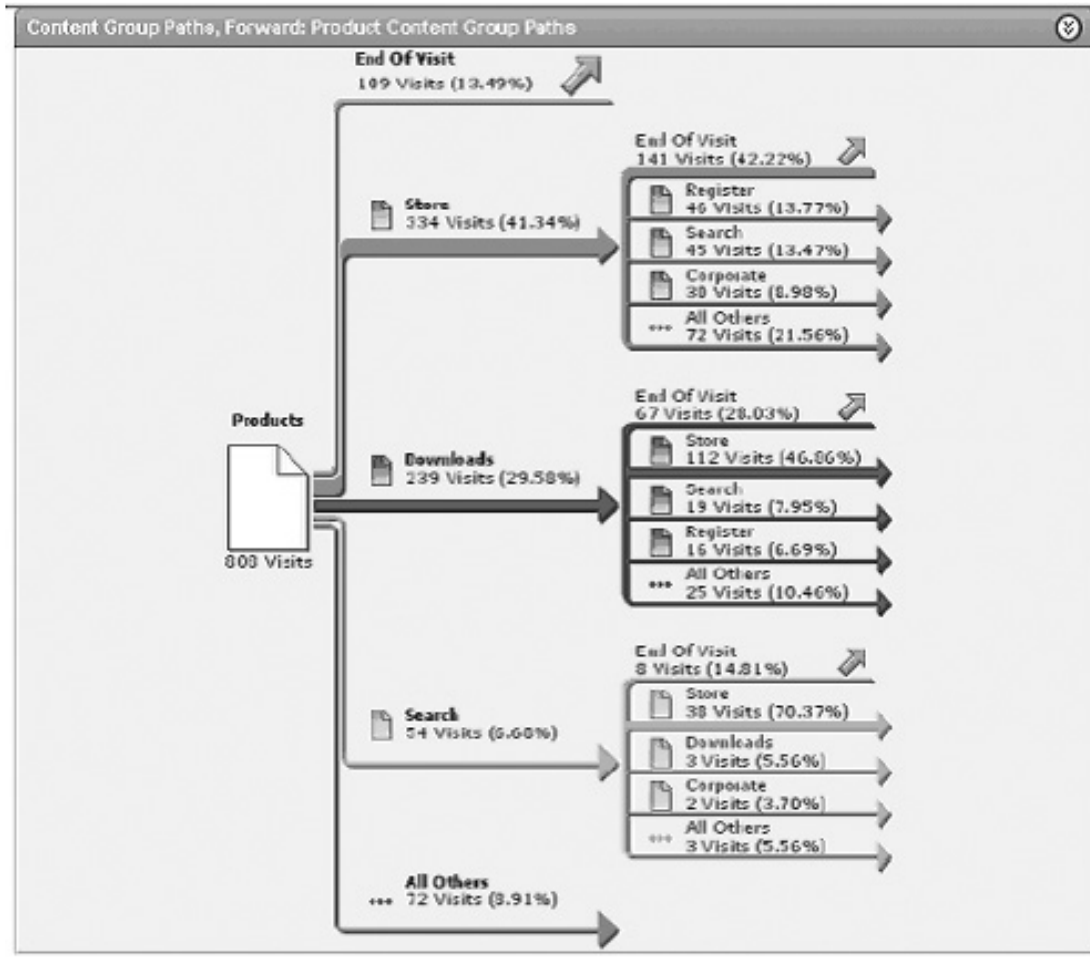


Figura 2 Path forward (WebTrends)

altri servizi offerti dal nostro sito che forse sono poco noti o troppo complicati da raggiungere. È la logica del supermercato: se siete entrati per comprare la birra e prima di uscire passate davanti ai salatini, è probabile che vi venga voglia anche di quelli (assumendo che ci sia un briciolo di Homer Simpsons in ognuno di voi...).

Un altro motivo valido per rilevare le visite ad un sito è che un dato simile può costituire un solido riscontro agli investimenti necessari alla realizzazione e alla gestione di un servizio web.

Attraverso il sistema di rilevazione degli accessi al sito potrete portare una prova tangibile dell'interesse dei visitatori per un dato servizio, e giustificare così gli investimenti fatti per realizzarlo o quelli utili per migliorarlo. Viceversa potrete decidere di chiudere i servizi che non vengono apprezzati dagli utenti e dirigere verso altri obiettivi le risorse impegnate.

Particolarmente importante è capire da dove vengono i visitatori del nostro sito: da link su siti partner, da banner pubblicitari su portali generalisti, dai motori di ricerca o da email che noi stessi abbiamo inviato. Può essere anche molto utile capire se i visitatori che provengono da fonti diverse hanno comportamenti diversi, ossia se è possibile segmentare gli utenti sulla base della loro provenienza. Questi dati consen-

tono di decidere una strategia efficace per portare utenti sul nostro sito: è meglio rafforzare le partnership esistenti o cercarne di nuove? Rendono di più i banner sui portali generalisti o le keyword sponsorizzate sui motori di ricerca? Abbiamo molti utenti affezionati che però navigano solo in una parte ristretta del sito? Allora probabilmente piuttosto che cercare di acquisire nuovi utenti con campagne di web marketing, conviene cercare di offrire un servizio migliore ai visitatori regolari. Una volta individuate le fonti principali di utenti, cerchiamo di caratterizzarle: gli utenti che hanno una determinata provenienza hanno interessi omogenei? Può essere utile allora presentare a questi utenti una pagina ad hoc che evidenzi chiaramente quello che possiamo offrire per servire i loro interessi.

Ultimo ma non meno importante: un controllo regolare degli accessi consente di individuare eventuali problemi tecnici di fruizione: dal link sbagliato, al file troppo pesante da scaricare al plug-in troppo poco diffuso o di installazione troppo complessa.

Cosa si può misurare

Gli accessi ad un sito sono per la maggior parte anonimi, l'identificazione degli utenti viene solitamente riservata ai casi in cui è strettamente necessaria. Nonostante ciò, tra i mezzi di comunicazione di massa, il web è quello che ci consente di raccogliere più informazioni su come viene utilizzato il servizio e quali sono le caratteristiche dei suoi fruitori. Tutta la navigazione sul web è basata sul protocollo HTTP che è stateless (ovvero ogni richiesta dovrebbe venire eseguita indipendentemente dalle richieste precedenti e da quelle successive). Tuttavia da diversi anni le web application che comunemente siamo abituati ad utilizzare collegano le varie richieste fatte in sequenza da un utente utilizzando cookie o parametri, ed implementano così il concetto di sessione. Inoltre in una richiesta HTTP fatta da un browser internet sono solitamente presenti campi molto interessanti, come l'indirizzo IP del visitatore ed il *referer* (che riporta l'indirizzo HTTP della pagina visualizzata precedentemente).

Tutte queste informazioni vengono analizzate dalle applicazioni di web analytics che generalmente utilizzano due *sorgenti di dati* (i log dei web server e/o degli script implementati in Javascript client-side):

- *I log dei web server*: il web server scrive in un file tutte le richieste HTTP che riceve ed i parametri ad esse associate (data e ora, indirizzo IP del richiedente, url richiesto, referer, cookie);
- *Javascript client side*: degli appositi script Javascript vengono aggiunti alle pagine erogate, vengono eseguiti dal browser dell'utente ed inviano le suddette informazioni ad un server.

Ciascuna di queste fonti di informazione ha i suoi pregi e i suoi difetti: in alcuni casi può essere meglio utilizzare i log del web server ed in altri i Javascript client-side, in altri casi ancora può essere opportuno utilizzare entrambe le fonti.

In ogni caso rimangono dei margini di incertezza sulle informazioni raccolte, che dipendono dalla grande varietà dei possibili utenti del sito. Le cache del browser o del proxy possono ad esempio nascondere una richiesta ai log del web server e la configurazione del browser può impedire l'esecuzione di Javascript o la memorizzazione di cookie.

Cercheremo di analizzare in dettaglio queste problematiche nel prossimo articolo, per ora vediamo quali dati si possono estrarre:

- *page views*: è il dato classico, il più antico e rimane un dato importante da analizzare;

- *visit:* è una serie di pagine consecutive viste dallo stesso visitatore. Se il visitatore non vede una nuova pagina in un intervallo di tempo specificato, la prossima pagina che egli vedrà sarà considerata come l'inizio di una nuova visita. L'affidabilità di questo dato è fondamentale per la va-

lidità di alcuni dei dati elencati di seguito: (visit ratio, paths, entry page, durata di una visita);

> È importante cercare di distinguere quello che succede veramente sul nostro sito da quello che noi supponiamo che succeda

- *visit ratio:* in media quante pagine vengono viste in una visita? Se questo numero è basso significa probabilmente che l'utente non è interessato a tutto quello che il sito offre. Bisogna cercare quindi di rendere più accattivanti le varie proposte. Tuttavia un rapporto alto non significa necessariamente che l'utente sia contento, potrebbe al contrario significare che il visitatore vaga per il sito non trovando quello a cui è interessato;

- *unique visitors:* premesso che questo è probabilmente uno dei dati più difficili da calcolare, esso vorrebbe rappresentare il numero di persone fisiche che visitano il sito. Solitamente viene misurato settando un cookie permanente nei browser dei visitatori (tuttavia questo cookie può essere rifiutato o cancellato successivamente). Va comunque considerato che un visitatore può collegarsi con diversi browser, con diversi computer, da casa, dall'ufficio ecc;

- *paths:* quali percorsi vengono seguiti dai nostri visitatori? Questo è uno dei dati più interessanti per capire le tipologie di utenti. Esistono vari modi anche grafici di rappresentarli.
  Ad esempio la **Figura 1** mostra graficamente i diversi percorsi seguiti per arrivare ad una determinata pagina, mentre la **Figura 2** mostra dove vanno i visitatori dopo aver visto una particolare pagina.

- *entry page/exit page*: sapere qual è la prima pagina che vedono in una visita i nostri utenti è molto interessante. Si tratta dell'home page o piuttosto gli utenti si sono creati bookmark ad altre pagine e le raggiungono direttamente? È bene ricordare, infatti, che non necessariamente l'home page viene vista da tutti gli utenti. Anche i navigatori che provengono da una query sui motori di ricerca possono arrivare direttamente sulla pagina contenente le informazioni alle quali sono interessati. Anche la pagina di uscita è importante: sapere dove i visitatori abbandonano il sito può evidenziare dei problemi. Non di rado ad esempio noterete che la *exit page* è la pagina in cui si richiede la registrazione: un tasso di abbandono a questo punto è fisiologico ma capire quanti visitatori si perdono per voler consentire la fruizione di un servizio ai soli utenti registrati consente di prendere delle decisioni consapevoli;

- *referrer:* da dove provengono i nostri utenti? Questo dato è spesso importante per poter segmentare la clientela o per individuare possibili siti partner;

- *Keywords*: un certo numero di utenti raggiunge il nostro sito cercando delle parole chiave sui motori di ricerca. I software di web analytics sono generalmente in grado di fare una classifica dei termini più ricercati (estraendoli dal referrer delle richieste). Questa informazione è importante per capire cosa cercano i visitatori che raggiungono il nostro sito. Può anche essere rivelatrice del buono/cattivo posizionamento del nostro sito rispetto a determinate keyword e può spingerci a riorganizzare il contenuto del sito, in modo che sia indicizzato meglio dai motori di ricerca, o aiutarci nella scelta dei termini da acquistare, nel caso decidessimo di fare una campagna sui motori di ricerca.

- *web campaign*: quanti utenti sono stati portati sul nostro sito dal banner che abbiamo messo su Virgilio e quanti da quello che abbiamo messo su Repubblica? Questo dato è fondamentale per valutare i risultati di una web campaign. Spesso è anche utile incrociare questo dato con il numero di pagine viste (o con gli acquisti effettuati, se vendiamo online). In questo modo possiamo determinare l'affinità dei visitatori del sito sorgente con le nostre attività e pianificare di conseguenza le prossime campagne. Solitamente per poter collegare un visitatore ad una campagna si aggiungono dei parametri identificativi ai link che vengono seguiti cliccando sul banner (es. *Http://www.mysite.it?from=yahoo*);

- *page group*: raggruppare le pagine del nostro sito secondo criteri diversi e valutare quali sono le aree di maggiore interesse è fondamentale. La flessibilità con la quale è possibile definire questi gruppi è una caratteristica importante dei software di web analytics. Specificare una per una le pagine che fanno parte di un gruppo può essere molto faticoso, ma solitamente è possibile definire delle espressioni regolari sia sui nomi delle pagine che sul valore dei parametri passati. In alcuni software una pagina può appartenere ad un solo gruppo e questo può essere limitante;

- *durata di una visita*: se è breve può essere un dato positivo, e significa probabilmente che il visitatore trova rapidamente quello che cerca. Questo dato è sicuramente solo indicativo, viene infatti calcolato misurando il tempo passato tra le varie richieste, ma non è solitamente possibile sapere per quanto tempo il visitatore si è fermato sull'ultima pagina, perché quando egli abbandona il sito non viene inviata alcuna informazione.

Conclusioni

Misurare gli accessi ad un sito web è interessante per molti motivi. Sapere come i vostri visitatori hanno trovato il sito e come lo hanno esplorato è importante sia che vogliate aumentare il numero di visitatori o che vogliate conoscere meglio quelli che già avete. Cercare di capire cosa cercano i visitatori e se e come trovano quello che cercano è utilissimo per decidere quali servizi ed informazioni aggiungere o rimuovere dal vostro sito. Tuttavia può essere molto complicato sia ottenere dei dati affidabili che saperli interpretare. Nel prossimo articolo ci focalizzeremo sulla scelta del modo più opportuno per misurare gli accessi evidenziando i limiti di incertezza dei vari approcci.

> Rilevare gli accessi può costituire un solido riscontro agli investimenti necessari alla realizzazione e alla gestione di un servizio web

Bibliografia

[1] Eric T. Peterson – "*Web Site Measurement Hacks*", O'Reilly , 2005

Riferimenti

[2] *WebTrends* – www.netiq.com
[3] *NetTracker* – www.sane.com
[4] *Summary* – www.summary.net
[5] *Urchin* – www.urchin.com
[6] *Pilot Hit List* – www.pilotsoftware.com
[7] *Analog* – http://www.analog.cx/
[8] *WebAlizer* http://www.mrunix.net/webalizer/
[9] *Pathalizer* http://pathalizer.sourceforge.net/
[10] http://www.clickz.com/experts/crm/actionable_analysis/
[11] http://www.webanalyticsdemystified.com/
[12] http://www.coffeesuntechnology.com/
[13] http://www.hurolinan.com/

Programmazione **Template-Oriented**

La programmazione orientata ai template enfatizza il riuso non solo dei componenti, ma anche delle loro modalità di utilizzo

di Lorenzo Vandoni

In questo articolo descriverò un approccio del tutto personale allo sviluppo del software, che ho battezzato Programmazione Template-Oriented. L'idea alla base di questo approccio è nata molti anni fa, quando ho partecipato allo sviluppo di dBsee, un prodotto CASE che ha avuto anche un discreto successo commerciale.

È passato molto tempo, l'idea iniziale è lentamente maturata, ed ha acquisito nel tempo anche alcune basi teoriche. Si tratta per ora solo di un'idea, ancora lontana da una realizzazione concreta, ma spero che a qualcuno possa interessare. Andiamo però con ordine, analizzando anzitutto le motivazioni alla base di questo approccio, ovvero alcuni limiti delle tecnologie attuali.

La programmazione object-oriented

La programmazione object-oriented (OOP) ha permesso di realizzare librerie di classi e framework strutturati decisamente meglio rispetto alle librerie di funzioni un tempo disponibili con linguaggi come C, Pascal, Basic e Clipper. Il livello di astrazione è più alto, e i meccanismi di ereditarietà e incapsulamento offrono ai progettisti la possibilità di creare framework anche molto complessi.

Dal punto di vista dell'utente/programmatore, ovvero di colui che ha la necessità di usare questi framework all'interno delle proprie applicazioni, le modalità di utilizzo di queste librerie non sono però cambiate molto, rispetto ad alcune decine di anni fa.

Gli strumenti più utilizzati, soprattutto nelle prime fasi di apprendimento, rimangono tuttora i manuali, l'help on line, e soprattutto gli esempi forniti a corredo della libreria. Molto frequentemente, il codice scritto per utilizzare una determinata libreria viene creato adattando opportunamente questi esempi. La disponibilità, la quantità e la qualità di documentazione ed esempi costituisce ancora oggi uno dei fattori più importanti per decidere di acquisire una determinata libreria. Il tempo richiesto per l'apprendimento, inoltre, è ancora una voce di costo non banale, e spesso capita di scegliere di utilizzare una libreria

***Lorenzo Vandoni** è laureato in Informatica, ed è uno specialista di progettazione e sviluppo con tecniche e linguaggi object-oriented. Attualmente è direttore del Laboratorio di Ricerca e Sviluppo di Emisfera e coordina un progetto di ricerca sulle Reti Wireless finanziato nell'ambito del Sesto Programma Quadro (FP6) della Comunità Europea. Può essere contattato tramite e-mail all'indirizzo vandoni@infomedia.it.*

solo perché ci è già nota, rinunciando magari ad alternative di migliore qualità, ma mai precedentemente utilizzate.

Il limite di questa tecnologia, dal punto di vista del riuso del codice, è quindi la poca attenzione rivolta agli stessi meccanismi di riuso, ovvero alle modalità con cui un componente riusabile possa essere selezionato, compreso, utilizzato e quindi inserito nel contesto di un'applicazione.

Sviluppi recenti nell'ingegneria del software

Alcuni tra i più recenti sviluppi nell'ambito dell'ingegneria del software hanno messo in luce, più o meno direttamente, questo problema, e hanno proposto diverse possibili soluzioni. Facendo una rapida carrellata, possiamo considerare:

- i Design Pattern, che hanno mostrato che è possibile definire soluzioni generiche per problemi ricorrenti. Questi pattern sono piuttosto generici, e mostrano come assemblare delle classi per risolvere un problema;
- La Programmazione Aspect-Oriented (AOP), che ha introdotto l'idea di programmare separatamente gli aspetti "orizzontali" di un'applicazione, come la gestione degli errori o della persistenza;
- Model Driven Architecture (MDA) e Executable UML (xUML), che hanno proposto di generare automaticamente il codice applicativo a partire da descrizioni di alto livello;
- Extreme Programming (XP), che ha posto l'enfasi sul fatto, non del tutto lapalissiano, che l'unico modo per scrivere più velocemente il codice è quello di scriverne meno.

Un aspetto comune a tutti questi approcci è quello di avere posto in evidenza alcuni limiti nelle attuali tecniche di progettazione e programmazione ad oggetti, suggerendo però soluzioni che non vanno considerate come alternative alla classica OOP, ma come elementi aggiuntivi, da utilizzare per la risoluzione di particolari problemi.

Anche la programmazione Template-Oriented va nella stessa direzione. Prima di introdurla, però, dobbiamo fare un passo indietro nel tempo.

I generatori di codice

Gli anni '90 hanno visto una certa proliferiazione di strumenti di sviluppo che si qualificavano come strumenti CASE di basso livello. L'adozione di questi strumenti prevedeva tipicamente l'utilizzo di strumenti visuali per specificare i vari componenti dell'applicazione, come ad esempio la struttura della base dati, il design delle maschere e dei report.

Il codice dell'applicazione veniva successivamente generato in modo del tutto automatico, includendo anche molte funzionalità complesse, e ottenendo significativi risparmi sui tempi di sviluppo e soprattutto di manutenzione.

> Gli strumenti più utilizzati sono tuttora i manuali, l'help on line e gli esempi forniti a corredo

Questi strumenti permettevano di implementare applicazioni in modo molto simile a quanto oggi può essere fatto, ad esempio, per la realizzazione dei programmi di installazione, utilizzando tool come InstallShield. Il problema principale dei generatori di codice, anche se allora non ci si rendeva conto di questo aspetto, era l'assoluta mancanza di attenzione verso gli standard.

Ogni strumento viveva in un mondo a sé, del tutto separato dagli altri, e permetteva di realizzare applicazioni sostanzialmente chiuse.

Per questo ed altri motivi questo approccio non ha avuto il successo che, forse, meritava. Visto che però oggi, grazie soprattutto a *MDA* e *xUML*, si torna a parlare di generazione di codice, può essere utile riconsiderare alcune lezioni che la storia legata all'utilizzo di questi strumenti ci dovrebbe avere insegnato, e in particolare:

- questi sistemi hanno funzionato solo se legati a un particolare dominio applicativo; non è mai stato costruito un generatore di codice del tutto generico;
- la struttura delle applicazioni generate è piuttosto rigida; questo, se da una parte può essere un beneficio in termini di omogeneità e standardizzazione del codice, dall'altra può risultare un problema in quanto rende più difficile soddisfare esigenze particolari;
- in ogni caso, è impossibile evitare del tutto interventi manuali sul codice generato, perché il generatore di codice non può prevedere tutte le situazioni possibili; intervenire sul codice significa anzitutto doverlo comprendere, e quindi prevedere un meccanismo che preservi le modifiche effettuate nel caso di successive generazioni di codice.

Credo che queste limitazioni non siano legate a particolari caratteristiche degli strumenti presenti negli anni '90, ma piuttosto intrinseche in un qualsiasi approccio che preveda la generazione di codice a partire da descrizioni di più alto livello.

La programmazione Template-Oriented

L'idea alla base della programmazione Template-Oriented è quella di fornire un supporto concreto all'utilizzo di librerie e framework per costruire applicazioni.

Ogni algoritmo che possa essere implementato tramite una libreria viene rappresentato usando uno scheletro di codice, denominato template, che può essere completato interattivamente dal programmatore usando un apposito tool, ottenendo la generazione del codice applicativo. La programmazione Template-Oriented prevede quindi due figure professionali: lo sviluppatore di sistema, che crea o conosce le librerie, e che si occupa di codificarne l'utilizzo all'interno dei template, e lo sviluppatore applicativo, che sfrutta i template per generare il codice applicativo.

Questo tipo di programmazione si può affiancare alle tecniche tradizionali, come OOP, e a quelle più recenti, senza ambire a sostituirle. Vediamo ora più in dettaglio come si potrebbe realizzare un sistema a supporto di questa idea.

> MDA e xUML hanno proposto di generare automaticamente il codice applicativo a partire da descrizioni di alto livello

Un sistema Template-Oriented

Un template, come si è detto, è una porzione di codice parametrica, di dimensione variabile, che codifica il modo con cui deve essere utilizzata una determinata libreria per implementare un certo tipo di funzione. Ad esempio, potremmo avere un template che codifica la realizzazione di una maschera gestionale di tipo testata-dettaglio in ASP.NET.

Il template contiene al suo interno elementi di tipo parametrico, scritti secondo la sintassi di un particolare Template Language, che identificano le informazioni che l'utente deve specificare per completare lo stesso template, in modo che questo possa essere convertito in codice compilabile e funzionante.

L'inserimento di queste informazioni viene effettuato utilizzando un apposito tool, in grado di leggere il template, riconoscere

gli elementi parametrici, e permettere all'utente di completarli. Oltre a un tool di tipo generico, utilizzabile per tutti i template, è possibile prevedere alcuni tool specifici, progettati per semplificare l'utilizzo di alcuni template particolari. Ad esempio, la posizione di un elemento grafico sullo schermo potrebbe essere definita trascinando l'elemento stesso alla posizione desiderata, e non scrivendo manualmente le coordinate corrispondenti.

Questi tool specifici dovrebbero somigliare a quelli comunemente presenti nei moderni ambienti di sviluppo visuale, ed essere utilizzati soprattutto per la gestione dei template relativi all'interfaccia grafica dell'applicazione.

Una volta completati tutti gli elementi di tipo parametrico, entrerà in gioco un generatore di codice, in grado di produrre i sorgenti dell'applicazione a partire dal template e dalle informazioni inserite dall'utente.

> Gli anni '90 hanno visto la proliferiazione di strumenti CASE di basso livello

Alcune caratteristiche aggiuntive

La disponibilità di uno o più tool che supportino l'utente nell'utilizzo dei template, e di un generatore di codice, costituiscono le caratteristiche minime di un sistema Template-Oriented.

Perché questo possa essere però realmente utile, occorre prevedere alcune caratteristiche aggiuntive, ovvero:

- un *Repository* che contenga tutte le informazioni inserite dall'utente e specifiche per una determinata applicazione. Questo repository risulta utile soprattutto per evitare di inserire più volte le stesse informazioni in template differenti. Per esempio, se due template richiedono l'accesso ad una tabella di database dove sono registrati i clienti dell'azienda, la stringa di connessione per l'accesso a questa tabella, il nome della stessa, e i nomi dei suoi campi potranno essere registrati nel repository e recuperati successivamente;
- un sistema che consenta di mantenere una visione globale dell'applicazione, che permetta di selezionare i vari template da utilizzare per realizzare i vari componenti del programma, e quindi di assemblare il codice generato.
  Questo sistema potrebbe somigliare molto ad uno dei moderni ambienti di sviluppo, o addirittura essere costituito da un add-in di uno di questi;
- un Template Editor rivolto agli sviluppatori di sistema.

Un sistema di questo tipo permetterebbe un reale riuso delle conoscenze e delle competenze, permettendo di utilizzare al meglio framework e librerie.

Potrebbe essere utilizzato sia come strumento di uso interno, in un'organizzazione che preveda una distinzione tra programmatori di sistema e applicativi, sia come supporto alla vendita o alla distribuzione di librerie commerciali.

Conclusioni

Come detto, per ora l'idea è poco più che abbozzata, e tale rimarrà fino a quando non riuscirò a trovare un meccanismo per riuscire a dedicarci più tempo e a coinvolgere altri.

I prossimi passi dovrebbero riguardare una più precisa definizione dei componenti fondamentali del sistema, primo tra tutti il *Template Language*, con particolare attenzione agli standard esistenti ed emergenti. In ogni caso, al di là del fatto di riuscire o meno ad arrivare ad un risultato concreto, questo potrà servire a ragionare attivamente sulle tecnologie che utilizziamo ogni giorno, invece di limitarsi a subirle.

Controllo Remoto in Visual Basic .NET

In questa ultima puntata, creeremo una classe per gestire da remoto il registro di sistema

Quinta puntata

di Stefano Corti

Dopo avere esaminato il codice che implementa tutte le funzionalità per trasferire file di qualsiasi estensione tra le due macchine, occupiamoci della progettazione di una classe che espone proprietà e metodi per gestire da remoto il registro di sistema.

In verità i sistemi operativi WindowsXP/2000/NT consentono nativamente tale operazione, ma riteniamo opportuno implementare anche questo aspetto operativo nel nostro progetto, al fine di familiarizzare e comprendere in dettaglio le classi del .NET framework che si occupano della gestione del registry. In questa trattazione diamo però per scontato che il lettore, così come l'eventuale amministratore che opera sulla macchina Controller, conosca approfonditamente il registro e che sappia come gestirlo correttamente, avvertendo che un'errata operazione su tale sistema può compromettere definitivamente la stabilità dell'intera macchina, costringendo ad una completa reinstallazione del sistema operativo.

Come si può osservare nella **Figura 1**, l'amministratore deve prima di tutto decidere su quale Hive intende agire, semplicemente selezionando il pulsante radio corrispondente alla Hive interessata.

Stefano Corti *si occupa di programmazione PHP e JSP lato server, della piattaforma .NET e di integrazione di sistemi legacy con le nuove realtà del web, soprattutto in ambito gestionale, bancario e finanziario. È attualmente alle dipendenze di un primario gruppo bancario italiano.*

È quindi possibile operare su LocalMachine, CurrentUser, ClassesRoot e Users. Premendo il pulsante "Imposta Chiave Hive Registro su cui lavorare", viene memorizzata la chiave base sulla quale verranno eseguite le varie interrogazioni ed eventuali modifiche.

Vediamo come scrivere il codice per ottenere queste funzioni. Nel **Listato 1** si può vedere il codice che gestisce, lato Controller, la GUI cui abbiamo appena fatto cenno. In questo caso il comando inviato attraverso il nostro socket di flusso è SETREGISTRYBASEKEY>, seguito dal nome vero e proprio della Hive del registro remoto. Naturalmente abbiamo anche previsto, mediante un semplice valore flag, l'apposito messaggio di errore nel caso che l'utente prema il pulsante senza avere precedentemente selezionato alcun radio button. Vediamo ora come tutto questo viene gestito lato server. Nel **Listato 2** possiamo vedere la porzione all'interno della struttura *Select Case* che si occupa di interpretare e gestire

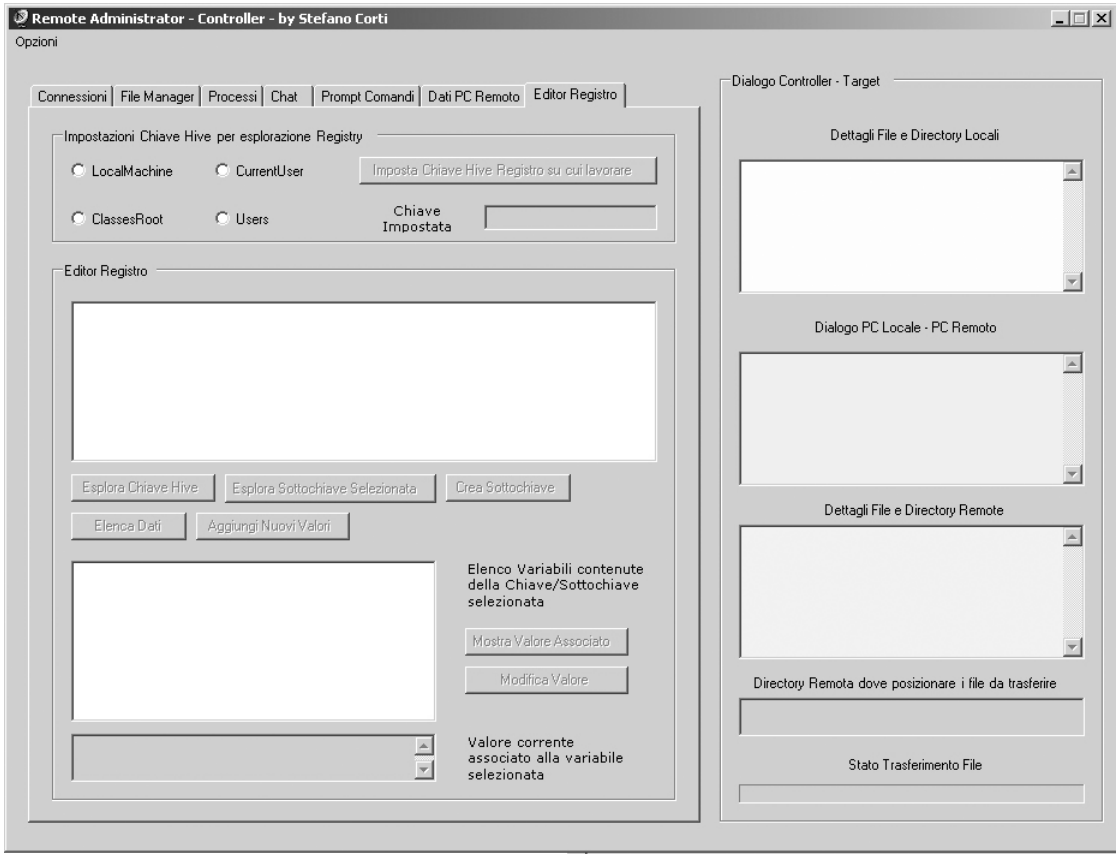


Figura 1 Controllo del registro remoto

tutti i comandi preposti alla gestione del registro. Precisiamo che abbiamo già provveduto a creare un oggetto *objReg* di tipo *RemoteRegistry* per il quale verranno invocati i relativi metodi pubblici esposti dall'omonima classe *RemoteRegistry* che ci accingiamo a realizzare. Per impostare la chiave Hive su cui lavorare, assegniamo quindi il valore di *job(1)* alla proprietà *SelectedBaseKey* dell'oggetto *objReg*.

Nel **Listato 3** possiamo vedere il codice che implementa le tre semplici proprietà pubbliche della nostra classe. In pratica il valore di *job(1)* viene passato ad una variabile privata all'interno della classe.

Questa variabile servirà esclusivamente all'interno della classe medesima per mantenere in memoria la chiave Hive del registro sulla quale si sta lavorando. Nella GUI del controller troviamo quindi un secondo riquadro che contiene l'editor del registro vero e proprio. Il primo pulsante permette di eseguire un'esplorazione completa della Hive selezionata.

> Occupiamoci della progettazione di una classe che espone proprietà e metodi per gestire da remoto il registro di sistema

Nel **Listato 4** è possibile osservare il codice che implementa il metodo pubblico *exploreBaseRegistry()* che permette al server Target

Listato 1 Metodo che imposta la Hive di partenza

```
Private Sub impoBaseKeyBtn_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles impoBaseKeyBtn.Click
   Dim messGo As String = "SETREGISTRYBASEKEY>"
   Dim ba_ke As String
   Dim rFlag As Integer

   If RadioRLocalMachine.Checked Then
      ba_ke = "LocalMachine"
      rFlag = 1
   End If
   If RadioRClassesRoot.Checked Then
      ba_ke = "ClassesRoot"
      rFlag = 1
   End If
   If RadioRCurrentUser.Checked Then
      ba_ke = "CurrentUser"
      rFlag = 1
   End If
   If RadioRUsers.Checked Then
      ba_ke = "Users"
      rFlag = 1
   End If

   If (rFlag) Then
      Try
         writer.Write(messGo & ba_ke)
         inMessages.Clear()
         inMessages.Text &= vbCrLf & "Comando Inviato: " & _vbCrLf & messGo
      Catch exception As SocketException
         inMessages.Text &= "Errore di comunicazione con il computer remoto: " & _exception.Message & vbCrLf
      Catch ex As Exception
         inMessages.Text &= "Errore: " & ex.Message & vbCrLf
      End Try
   Else
         MessageBox.Show("Dovete scegliere una classe base", "Errore Registro", MessageBoxButtons.OK, MessageBoxIcon.Stop)
   End If
End Sub
```

di restituire al Controller una stringa contenente tutte le chiavi all'interno della Hive selezionata. La stringa ottenuta può essere "splittata" mediante un carattere separatore e i vari elementi possono essere utilizzati dal Controller per popolare il *ListBox1* visibile in **Figura 1**. Osserviamo l'utilizzo del metodo pubblico *GetSubKeyNames()* membro della classe *RegistryKey*, che consente di recuperare una matrice di stringhe contenente i nomi delle sottochiavi. Tale matrice viene quindi iterata nel successivo ciclo *For Each Next* per costruire la stringa *rReturnKeyNames* il cui valore viene ritornato al ciclo principale del server. Proseguendo nella progettazione della GUI del Controller, vediamo che il *ListBox1* viene popolato con tutte le chiavi contenute all'interno della Hive selezionata. A questo punto l'amministratore può selezionare una sottochiave, eseguire un'esplorazione della nuova sottochiave selezionata, oppure creare una sottochiave del tutto nuova. Nel **Listato 5** è visibile il codice lato Controller che consente di intercettare la sottochiave selezionata nel *ListBox1* ed eseguire una nuova esplorazione. Il metodo pubblico della classe *RemoteRegistry,* che consente una successiva esplorazione relativa ad una sottochiave selezionata nel *ListBox1*, è visibile nel **Listato 6** e viene chiamato *exploreTreeRegistry()*. Si tratta di un metodo abbastanza complesso che ora analizzeremo in dettaglio. Prima di tutto alla variabile *rK* viene passato per valo-

re il contenuto di *job(1)*. La variabile privata *rKey* di tipo *RegistryKey*, poiché modificata mediante il metodo *exploreBaseRegistry()*, contiene il valore della chiave Hive corrente. Viene quindi invocato il metodo sovraccarico *OpenSubKey()* che consente di recuperare una sottochiave specificata e di determinare l'accesso in scrittura alla chiave stessa, impostando su *True* il secondo parametro di overload di tipo *Boolean*. Successivamente viene recuperato l'array di valori utilizzando il metodo *GetSubKeyNames()*, già analizzato. All'interno del ciclo *For Each* viene quindi creata la stringa *rKeyToExplore* contenente il percorso relativo di ciascuna sottochiave, con tanti livelli quanto più l'esplorazione è stata eseguita in profondità. Per delimitare ciascun percorso, utilizziamo il consueto carattere separatore, mentre il valore della Hive di partenza si trova sempre memorizzato nella variabile privata *rKey*. Il controllo sulla matrice *oSK*, mediante la proprietà *Length,* viene effettuato allo scopo di determinare se l'array stesso contiene più di un elemento, ovvero se il numero totale di indici è maggiore di zero.

> Un'errata operazione sul registry può compromettere definitivamente la stabilità dell'intera macchina

In caso contrario significa che l'amministratore è arrivato al livello più basso dell'esplorazione e quindi l'ultima sottochiave esplorata non può che contenere coppie di nomi e valori. Tali elementi vengono recuperati tramite i metodi *GetValueNames()* e *GetValue()*. Il primo metodo consente di recuperare un array di stringhe contenente tutti i nomi dei valori associati alla chiave, mentre il secondo consente di recuperare il valore specificato. Entrambi i metodi sono

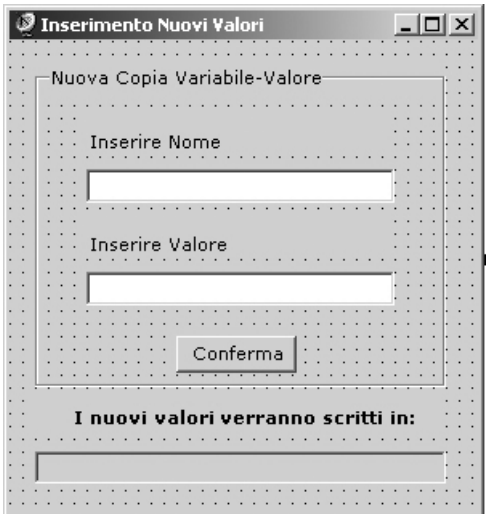


Figura 2 Form3 per l'inserimento di nome/valore all'interno di una chiave/sottochiave

membri della classe *RegistryKey*. Vediamo ora come creare una nuova sottochiave all'interno della struttura del Registry. Il metodo in questione è visibile nel **Listato 7**, mentre il **Listato 8** contiene il codice che gestisce l'intera operazione dal lato Controller. Prima di tutto l'amministratore deve selezionare all'interno del *ListBox1* una chiave (o una sottochiave) all'interno della quale generare una nuova ulteriore sottochiave. Alla variabile stringa *kToC* viene passato il valore ritornato dalla funzione *InputBox* che determina la comparsa di una finestra di prompt con relativo campo di testo, nel quale l'utente dovrà inserire il nome della sottochiave da generare (che sarà in definitiva una ulteriore sottochiave della sottochiave – o chiave – selezionata del *ListBox1*). La stringa finale, contenuta nella variabile *messGo*, sarà dunque costituita dall'intero persorso relativo (senza radice hive) copiato direttamente dalla stringa selezionata nel *ListBox1* più un carattere separatore seguito infine dal nome della nuova sottochiave (valore stringa di *KtoC*). Il metodo publico *createSubRegistryKey()* in esecuzione su Target, si occupa prima di tutto di segmentare la stringa inviata attraverso il

Listato 2 Struttura Select Case per il controllo del registro (continua...)

```
Case Is = "SETREGISTRYBASEKEY"
   Try
      sentText.Clear()
      sentText.Text = "IMPOSTATA CHIAVE BASE REGISTRO: " & job(1) & vbCrLf
      objReg.SelectedBaseKey = job(1)
      writer.Write("OKREGBASEKEY>" & objReg.SelectedBaseKey)
   Catch exception As SocketException
      sentText.Clear()
      sentText.Text = "Errore di Comunicazione: " & exception.Message & vbCrLf
   Catch er As Exception
      sentText.Clear()
      sentText.Text = "Errore: " & er.Message & vbCrLf
   End Try

Case Is = "MODIFYREGKEYVALUE"
   sentText.Clear()
   sentText.Text = "Richiesta Modifica Registro."
   Try
      writer.Write("CONFIRMMODIFYREGKEYVALUE>" & objReg.showOnlyKeyValue(job(1)))
   Catch exception As SocketException
      sentText.Clear()
      sentText.Text = "Errore di Comunicazione: " & exception.Message & vbCrLf
   Catch er As Exception
      sentText.Clear()
      sentText.Text = "Errore: " & er.Message & vbCrLf
   End Try

Case Is = "SETNEWKEYVALUE"
   sentText.Clear()
   sentText.Text = "Richiesta modifica Registro in corso..."
   Try
      writer.Write("OKMODIFYREGKEYVALUE>" & objReg.modReg(job(1)))
   Catch exception As SocketException
      sentText.Clear()
      sentText.Text = "Errore di Comunicazione: " & exception.Message & vbCrLf
   Catch er As Exception
      sentText.Clear()
      sentText.Text = "Errore: " & er.Message & vbCrLf
   End Try
Case Is = "SETNEWNAMEANDVALUE"
   sentText.Clear()
   sentText.Text = "Richiesta Inserimento Nuovi Valori in Registro in corso..."
   Try
      writer.Write("RESULTCREATEREGKEYVALUE>" & objReg.creaReg(job(1)))
   Catch exception As SocketException
      sentText.Clear()
      sentText.Text = "Errore di Comunicazione: " & exception.Message & vbCrLf
   Catch er As Exception
      sentText.Clear()
      sentText.Text = "Errore: " & er.Message & vbCrLf
   End Try
Case Is = "GETBASESUBKEYS"
   Try
      sentText.Clear()
      sentText.Text = "Trasmissione sottochiavi base registro" & vbCrLf
      writer.Write("REPLYBASESUBKEYS>" & objReg.exploreBaseRegistry())
   Catch exception As SocketException
      sentText.Clear()
      sentText.Text = "Errore di Comunicazione: " & exception.Message & vbCrLf
   Catch er As Exception
      sentText.Clear()
      sentText.Text = "Errore: " & er.Message & vbCrLf
   End Try

Case Is = "GETSELECTEDSUBKEYS"
```

Listato 2 (...fine)

```
   sentText.Clear()
   sentText.Text = "Richieste SottoChiavi del Registry di Sistema"
   Try
     writer.Write(objReg.exploreTreeRegistry(job(1)))
   Catch exception As SocketException
     sentText.Clear()
     sentText.Text = "Errore di Comunicazione: " & exception.Message & vbCrLf
   Catch sre As Exception
     sentText.Clear()
     sentText.Text = "Errore: " & sre.Message & vbCrLf
   End Try

Case Is = "CREATESUBKEY"
   sentText.Clear()
   sentText.Text = "Creazione nuova Chiave/Sottochiave in corso" & vbCrLf
   If (objReg.createSubRegistryKey(job(1))) Then
   Try
     writer.Write("OKCREATESUBKEY>Nuova Chiave/Sottochiave creata correttamente")
   Catch exception As SocketException
     MessageBox.Show("Errore di Comunicazione: " & exception.Message, _
     "Errore di Comunicazione", MessageBoxButtons.OK, MessageBoxIcon.Error)
   Catch e As Exception
     MessageBox.Show("Errore Generico: " & e.Message, _
     "Errore di Comunicazione", MessageBoxButtons.OK, MessageBoxIcon.Error)
   End Try
   Else
   Try
     writer.Write("OKCREATESUBKEY>Errore: creazione nuova Chiave/Sottochiave non eseguita")
   Catch exception As SocketException
     MessageBox.Show("Errore di Comunicazione: " & exception.Message, _
     "Errore di Comunicazione", MessageBoxButtons.OK, MessageBoxIcon.Error)
   Catch e As Exception
     MessageBox.Show("Errore Generico: " & e.Message, _
     "Errore di Comunicazione", MessageBoxButtons.OK, MessageBoxIcon.Error)
   End Try
   End If

Case Is = "SHOWKEYNAMESANDVALUES"
   sentText.Clear()
   Dim rChain As String = objReg.showKeyNamesAndValues(job(1))
   sentText.Text = "Trasmissione copie nomi/valori chiave registro" & vbCrLf
   Try
     writer.Write(rChain)
   Catch exception As SocketException
     MessageBox.Show("Errore di Comunicazione: " & exception.Message, _
     "Errore di Comunicazione", MessageBoxButtons.OK, MessageBoxIcon.Error)
   Catch e As Exception
     MessageBox.Show("Errore Generico: " & e.Message, _
     "Errore di Comunicazione", MessageBoxButtons.OK, MessageBoxIcon.Error)
   End Try

Case Is = "SHOWONLYKEYVALUE"
   sentText.Clear()
   sentText.Text = "Interrogazione contenuto valore chiave registry in corso..."
   Dim vChain As String = "REPLYONLYKEYVALUE>" & objReg.showOnlyKeyValue(job(1))
   Try
     writer.Write(vChain)
   Catch exception As SocketException
     MessageBox.Show("Errore di Comunicazione: " & exception.Message, _
     "Errore di Comunicazione", MessageBoxButtons.OK, MessageBoxIcon.Error)
   Catch e As Exception
     MessageBox.Show("Errore Generico: " & e.Message, _
     "Errore di Comunicazione", MessageBoxButtons.OK, MessageBoxIcon.Error)
   End Try
```

Listato 3 Proprietà pubbliche della classe RemoteRegistry

```
Public Property SelectedBaseKey() As String
  Get
    Return rBaseKey
  End Get
  Set(ByVal Value As String)
    rBaseKey = Value
    rCurrentKey = Nothing
  End Set
End Property

Public Property CurrentKey() As String
  Get
    Return rCurrentKey
  End Get
  Set(ByVal Value As String)
    rCurrentKey = Value
  End Set
End Property

Public Property CurrentValue() As String
  Get
    Return rCurrentValue
  End Get
  Set(ByVal Value As String)
    rCurrentValue = Value
  End Set
End Property
```

socket di rete, ottenendo quindi tutto il percorso relativo gerarchico ed infine genera la nuova sottochiave. Tutto questo viene eseguito dalla seguente linea di codice.

```
rKey.OpenSubKey(kCCC(0), True).CreateSubKey(kCCC(1))
```

> In una particolare chiave o sottochiave possono essere contenute diverse variabili

Il valore di *kCCC(0)* contiene il percorso relativo completo all'interno della struttura del registro, che viene passato al metodo sovraccarico *OpenSubKey()*, mentre *kCCC(1)* è il valore che effettivamente viene passato al metodo *CreateSubKey()* che si occupa della generazione della nuova sottochiave. Come sempre *rKey* contiene un valore di tipo *RegistryKey* che rappresenta la Hive di partenza.

In una particolare chiave o sottochiave possono essere contenute diverse variabili (o nomi) ed ogni nome può contenere dati di diverso tipo.

I valori consentiti nel registro di sistema possono essere di tipo DWORD, binario e stringa. I valori di tipo stringa possono essere rappresentati dalle seguenti categorie:

- *sz*: i dati sono rappresentati come valore di tipo stringa Unicode con terminazione null
- *multi_sz*: i dati sono rappresentati come matrice di stringhe Unicode con terminazione null
- *expanded_sz*: i dati sono rappresentati come stringa Unicode con terminazione null e riferimenti espansi alle variabili di ambiente

Ricordiamo che la chiave/sottochiave aperta con il valore da impostare deve essere stata definita con l'accesso in scrittura e non può essere di sola lettura.

Una volta determinato l'accesso in scrittura a una chiave è possibile modificare i dati associati a qualsiasi valore della chiave stessa. Inoltre è prevista anche la possibilità di aggiungere nuove coppie nome/valore all'interno di una particolare chiave o sottochiave. In questo contesto creiamo una nuova Form che verrà inizializzata e resa visibile quando l'amministratore premerà il relativo pulsante sulla GUI del controller. La Form è visibile in **Figura 2**. Abbiamo previsto due campi di testo nei quali l'utente può inserire la variabile ed il valore associato.

Dal **Listato 9** è possibile evincere che in questo contesto è direttamente la nuova Form che si occupa di iniettare nel socket di flusso i valori digitati dall'utente unitamente al comando relativo, in quando nella Form principale dell'applicativo Controller (Form1) gli oggetti *writer* e *reader* – rispettivamente di tipo *BinaryWriter* e *BinaryReader* – sono dichiarati pubblici e statici, quin-

di accessibili anche dalla classe Form3, invocabili direttamente specificando la classe di appartenenza, senza bisogno di istanziare nuovi oggetti.

Sul lato server il metodo che abbiamo implementato si chiama *creaReg()* ed utilizza la seguente linea di codice per creare i nuovi valori.

```
rKey.OpenSubKey(CCCC(0), True).SetValue(CCCC(1), CCCC(2))
```

Il metodo *SetValue()* prevede appunto due parametri e genera così un nuovo nome di variabile ed il relativo valore all'interno della chiave/sottochiave selezionata aperta per la scrittura dal metodo *OpenSubKey()*, più volte menzionato. L'amministratore che opera sul programma Controller ha anche la possibilità di modificare il valore di una o più variabili già presenti nella chiave/sottochiave selezionata. Per prima cosa devono essere elencate tutte le coppie nome-valore all'interno della sottochiave scelta.

Tali elementi, recuperati mediante i metodi già descritti, vengono visualizzati nel *ListBox2* nella parte inferiore dello schermo (vedi **Figura 1**). L'utente può quindi selezionare una particolare variabile in questo ListBox e visualizzare i dati in essa contenuti oppure può anche decidere di modificare i dati medesimi. Questo processo è un poco più complesso rispetto ai casi fin qui illustrati, poiché prevede più sessioni di comunicazione basate sul nostro protocollo tra il server Target e la macchina Controller. Prima di tutto alla pressione del pulsante "Modifica Valore" viene inviato il comando MODIFYREGKEYVALUE> seguito dal-

Listato 4 Metodo che esplora le Hive del Registry

```
Public Function exploreBaseRegistry() As String
  rReturnKeyNames = Nothing
  rCurrentKey = Nothing
  If rBaseKey = "CurrentUser" Then
    rKey = Registry.CurrentUser
  ElseIf rBaseKey = "LocalMachine" Then
    rKey = Registry.LocalMachine
  ElseIf rBaseKey = "ClassesRoot" Then
    rKey = Registry.ClassesRoot
  ElseIf rBaseKey = "Users" Then
    rKey = Registry.Users
  Else
    MessageBox.Show("Errore nella Selezione _
    Chiave Base", "Errore Registry", _
    MessageBoxButtons.OK, MessageBoxIcon.Error)
  End If
  rListKeyNames = rKey.GetSubKeyNames()
  For Each rArrayKeyNames In rListKeyNames
    rReturnKeyNames &= rArrayKeyNames & "*"
  Next
  Return rReturnKeyNames
End Function
```

Listato 5 Gestione del pulsante Esplora Hive Selezionata

```
Private Sub subKeyExploBtn_Click(ByVal sender As System.Object, ByVal e As
                                 System.EventArgs) Handles subKeyExploBtn.Click
  Dim selRR As String = ListBox1.SelectedItem
  If ((RadioRUsers.Checked Or RadioRLocalMachine.Checked Or _
  RadioRClassesRoot.Checked Or RadioRCurrentUser.Checked) And _
  (baseKeyField.Text <> Nothing Or baseKeyField.Text <> "")) And _
  (selRR <> Nothing Or selRR <> "") Then
    Dim messGo As String = "GETSELECTEDSUBKEYS>" & selRR
    Try
      writer.Write(messGo)
      inMessages.Clear()
      inMessages.Text &= vbCrLf & "Comando Inviato: " & _
      vbCrLf & messGo
    Catch exception As SocketException
      inMessages.Text &= "Errore di comunicazione con il computer remoto: " & _
      exception.Message & vbCrLf
    Catch ex As Exception
      inMessages.Text &= "Errore: " & ex.Message & vbCrLf
    End Try
  Else
      MessageBox.Show("Dovete prima scegliere e impostare una classe base" & _
      vbCrLf & "poi selezionare una sottochiave dalla lista", "Errore Registro",
  MessageBoxButtons.OK, MessageBoxIcon.Stop)
  End If
End Sub
```

Listato 6 Metodo che esplora gerarchicamente la struttura del Registry

```
Public Function exploreTreeRegistry(ByVal rK As String) As String
  rKeyToExplore = Nothing
  kToExplore = Nothing

  kToExplore = rK

  Dim oSK As String()
  Try
    rCurrentKey = kToExplore
    oSK = rKey.OpenSubKey(kToExplore, True).GetSubKeyNames()
    Dim listASK As String
    If oSK.Length > 0 Then
      For Each listASK In oSK
        If (rCurrentKey <> Nothing) Then
          rKeyToExplore &= kToExplore & "¥" & listASK & "*"
        Else
          rKeyToExplore &= kToExplore & "¥" & listASK & "*"
        End If
      Next
      Return "REPLYSUBKEYS>" & rKeyToExplore
    Else
      Dim rMatr As String()
      Dim rT As String
      Dim rFS As String
      rMatr = rKey.OpenSubKey(rCurrentKey, True).GetValueNames()
      For Each rT In rMatr
        rFS &= "Nome Chiave: " & rT & vbCrLf
        Try
          rFS &= "Valore: " & rKey.OpenSubKey(rCurrentKey, True).GetValue(rT) & "*"
        Catch ecast As System.InvalidCastException
          rFS &= "..........." & "*"
        Catch erege As Exception
          rFS &= "..........." & "*"
        End Try
      Next
      Return "REPLYKEYVALUES>" & rFS
    End If
  Catch genex As Exception
    MessageBox.Show("Errore nell'interrogazione del registro: " & genex.Message, MessageBoxButtons.OK,
                                                                                  MessageBoxIcon.Exclamation)
  End Try
End Function
```

Listato 7 Metodo per la creazione di nuove chiavi/sottochiavi

```
Public Function createSubRegistryKey(ByVal sToSplit As String) As Boolean
  Dim o_delimStr As String = "*"
  Dim o_delimiter As Char() = o_delimStr.ToCharArray()
  Dim kCCC() As String = sToSplit.Split(o_delimiter, 2)
  Try
    rKey.OpenSubKey(kCCC(0), True).CreateSubKey(kCCC(1))
    Return True
  Catch eCCC As Exception
    Return False
  End Try
End Function
```

l'elemento selezionato nel *ListBox2*.

Se osserviamo nuovamente il **Listato 2** possiamo notare che il server risponde con il comando di conferma CONFIRMMODIFYREGKEYVALUE> seguito dal valore contenuto nella variabile passata come argomento del metodo, quindi l'elemento selezionato nel *ListBox2*. Il va-

Listato 8 Gestione del pulsante Crea Sottochiave

```
Private Sub creaSubRKeyBtn_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles
creaSubRKeyBtn.Click
  Dim selCC As String = Nothing
  selCC = ListBox1.SelectedItem
  If (selCC <> Nothing Or selCC <> "") Then
    Dim kToC As String = Nothing
    kToC = InputBox("Immettere il nome della sottochiave da creare" & vbCrLf & _
    "all'interno di " & selCC, "Creazione Chiave/Sottochiave")
    Dim l_kToC = kToC.Length
    If (kToC = Nothing Or kToC = "" Or kToC = " " Or l_kToC < 2)
    Then MessageBox.Show("Dovete immettere un nome di chiave valido", _
      "Errore Chiave/Sottochiave", MessageBoxButtons.OK, MessageBoxIcon.Warning)
    Else
      Dim messGo As String = "CREATESUBKEY>" & selCC & "*" & kToC
      Try
        writer.Write(messGo)
        inMessages.Clear()
        inMessages.Text &= vbCrLf & "Comando Inviato: " & vbCrLf & messGo
      Catch exception As SocketException
        inMessages.Text &= "Errore di comunicazione con il computer remoto:   " _
        & exception.Message & vbCrLf
      Catch ex As Exception
        inMessages.Text &= "Errore: " & ex.Message & vbCrLf
      End Try
    End If
  Else
    MessageBox.Show("Dovete prima selezionare una chiave relativa dall'elenco", _
    "Errore nella creazione di nuova Chiave/Sottochiave", MessageBoxButtons.OK, MessageBoxIcon.Warning)
  End If
End Sub
```

lore viene ricavato invocando il metodo pubblico *showOnlyKeyValue()* passando per valore il contenuto di *job(1)*.

In questo metodo è importante impostare, prima dell'istruzione *Return*, la variabile privata *rCurrentValue* uguale al contenuto di *nVVVa,* ossia il valore di *job(1)* che in definitiva è il nome della variabile selezionata del *ListBox2*, che contiene i dati da archiviare o modificare.

Tale variabile privata verrà infatti utilizzata nel metodo *modReg()*, che ci accingiamo ad esaminare.

Quando il controller riceve il comando CONFIRMMODIFYREGKEYVALUE> chiama immediatamente la procedura function *modifyKeyProcedure()* passando come argomento il valore ritornato dal metodo *showOnlyKeyValue()* e trasmesso dal server attraverso il consueto socket TCP. Tale funzione è visibile nel **Listato 10**. A questo punto il Controller, in modo del tutto automatico, genera un nuovo comando SETNEWKEYVALUE> seguito dal nuovo dato da attribuire alla variabile.

> I valori consentiti nel registro di sistema possono essere di tipo DWORD, binario e stringa

Tale valore è ottenuto mediante un *InputBox* che mostra il consueto prompt di inserimento.

Nel **Listato 2** possiamo infine osservare che quando il server riceve questo coman-

Listato 9 Codice della classe Form3

```
Public Class Form3
  Inherits System.Windows.Forms.Form

  Private Sub Button1_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs)
                                                          Handles Button1.Click
    If (TextBox1.Text <> "" And TextBox1.Text <> Nothing And _
    TextBox2.Text <> "" And TextBox2.Text <> Nothing) Then
      Dim f3_messGo As String = Nothing
      f3_messGo = "SETNEWNAMEANDVALUE>" & showRegPath.Text & "*" & _
      TextBox1.Text & "*" & TextBox2.Text
      Try
        Form1.writer.Write(f3_messGo)
      Catch e3 As Exception
        MessageBox.Show("Errore: " & e3.Message, "Errore di Comunicazione", _
        MessageBoxButtons.OK, MessageBoxIcon.Exclamation)
      End Try
      Me.Close()
    End If
  End Sub
End Class
```

Listato 10-11 Modifica nome/valore all'interno di una chiave/sottochiave

```
Private Function modifyKeyProcedure(ByVal currKVal As String) As String
  Dim InB As String = Nothing
  Dim messGC As String = "SETNEWKEYVALUE>"
  InB = InputBox("Valore Corrente: " & currKVal & vbCrLf & "Inserire il nuovo valore.", _
  "Conferma Modifica Valori Registry Remoto")
  If (InB = Nothing Or InB = "") Then
    MessageBox.Show("Errore: Il valore da inserire non pu  essere nullo", _
    "Errore Modifica Registry Remoto", MessageBoxButtons.OK, MessageBoxIcon.Stop)
  Else
    Return messGC & InB
  End If
End Function



Public Function modReg(ByVal vTTTs As String) As String
  Dim retVal As String = Nothing
  If (rCurrentKey <> Nothing) Then
    If (rCurrentValue <> Nothing) Then
      Try
        rKey.OpenSubKey(rCurrentKey, True).SetValue(rCurrentValue, vTTTs)
      Catch eMMM As Exception
        retVal = "Errore durante la scrittura nel registro: " & eMMM.Message
      End Try
      retVal = "Scrittura nel Registry eseguita correttamente."
    Else
      retVal &= "Errore: non   stato selezionato un nome di variabile."
    End If
  Else
    retVal &= "Errore: non   stata selezionata una chiave/sottochiave."
  End If
  Return retVal
End Function
```

do, provvede immediatamente ad invocare il metodo *modReg()* visibile nel **Listato 11**.

In questo metodo troviamo i metodi della FCL già presentati.

Riteniamo opportuno però richiamare l'attenzione sulle variabili private *rCurrentKey* e *rCurrentValue* il cui valore viene modificato dalla proprietà *CurrentKey* e *CurrentValue* della classe *RemoteRegistry*.

Come il nome suggerisce esplicitamente, queste variabili contengono la chiave/sottochiave attualmente in memoria, unitamente al valore corrente associato.

Alla variabile *vTTTs* viene quindi passato per valore il contenuto di *job(1)* che esprime il nuovo dato da associare alla relativa variabile all'interno della chiave/sottochiave su cui si sta lavorando.

Ecco quindi che il metodo della FCL *SetValue()* provvede a scrivere il nuovo dato da archiviare (*vTTTS*) associato al valore espresso dalla variabile privata *rCurrentValue*.

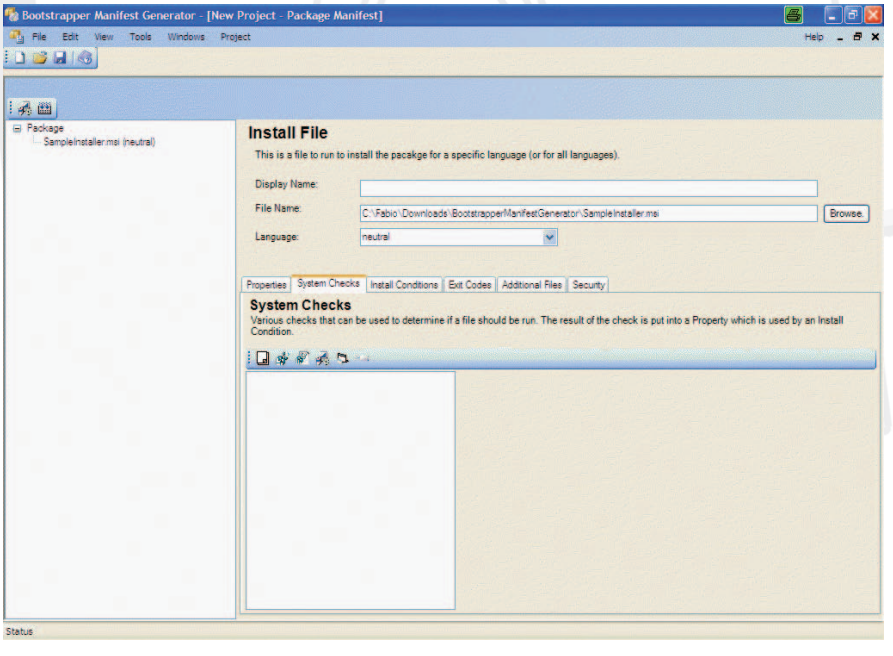
Bootstrapper Manifest Generator

di Fabio Perrone

Uno strumento gratuito per la creazione dei package d'installazione

Per chiunque scriva applicazioni desktop arriva prima o poi il momento della distribuzione dell'applicazione stessa e di tutte le componenti richieste per il suo corretto funzionamento; chi ha già avuto un minimo di esperienza in questo campo sa benissimo che questa operazione può essere particolarmente noiosa e "pericolosa": se non vengono installate tutte le componenti, niente potrà impedire alla nostra applicazione di rispondere con un bell'errore al momento del caricamento del componente mancante. Con l'arrivo di Visual Studio 2005 il problema si è notevolmente semplificato grazie al "Visual Studio 2005 Bootstrapper" che fornisce un metodo piuttosto immediato per individuare ed eventualmente scaricare dalla rete (o installare da un supporto quale un CD-ROM) un'applicazione e tutti i suoi componenti. Una delle caratteristiche principali del Bootstrapper di VS 2005 è la sua estensibilità, in quanto è possibile permettere l'installazione di qualsiasi componente tramite una coppia di file XML (chiamati "manifest" XML del Bootstrapper) che contengono tutti i dati necessari per eseguire l'installazione. Questi due file XML sono il Product Manifest, che contiene dati comuni alle differenti versioni localizzabili dell'applicazione, e il Package Manifest, che viene usato per dati specifici per ogni lingua, quali i messaggi di errore. Poiché la creazione di questi due file potrebbe essere abbastanza complessa, ecco che viene in aiuto il tool protagonista di questa recensione.

Una volta creato il file d'installazione .msi con VS 2005, lo si apre all'interno del Bootstrapper Manifest Generator (BMG). Immediatamente vengono creati i seguenti tab: Properties, System Checks, Install Conditions, Exit Codes, Additional Files e Security. Il tab "Properties" si utilizza per passare al file di setup eventuali argomenti a riga di comando, per gestire un eventuale *reboot* del sistema dopo l'installazione, per fornire un tempo stimato per l'installazione (l'utilità di quest'opzione è provvedere una progress-bar che segue l'andamento dell'installazione; il problema principale è che non sempre si conosce la reale velocità del PC su cui verrà installata l'applicazione,



quindi non può essere altro che una stima), la dimensione dell'installazione. D'importanza fondamentale è il tab "System Checks", che permette di impostare delle condizioni per capire se un determinato componente deve essere installato. Il risultato di un System Check viene salvato in una proprietà che verrà poi utilizzata dalle "Install Conditions". Qui è possibile controllare la presenza di un determinato file, l'esistenza di una chiave di registro, un controllo personalizzato scritto in qualsiasi linguaggio (in poche parole un eseguibile esterno che dovrebbe essere scritto in C++ unmanaged, in quanto il .NET Framework potrebbe non essere installato sulla macchina target) oppure individuare se un prodotto è già installato tramite il Product Code fornito da un'installazione effettuata con Windows Installer. Mentre i System Check vengono effettuati per capire se sul PC target sia necessario o meno installare un componente, le Install Conditions pongono dei limiti entro i quali la procedura di installazione deve essere eseguita, pena il suo fallimento, quali la versione del sistema operativo, la necessità di effettuare l'installazione con i diritti da Amministratore del sistema e così via. L'utilizzo principale degli Exit Codes è invece quello di capire se l'installazione è andata a buon fine e, in caso positivo, eventualmente effettuare un reboot del sistema.

Se dovesse sopraggiungere qualche problema è possibile fornire all'utente un messaggio di errore personalizzato. Il tab "Additional Files" permette di inserire file aggiuntivi nell'installazione, mentre "Security" permette di determinare se devono essere eseguite delle convalide prima dell'installazione del prodotto, per evitare che l'installazione sia stata manomessa allo scopo di installare sul PC dell'utente del malware indesiderato.

Una volta eseguite tutte le scelte, si compie una procedura di compilazione producendo un file manifest che si potrà poi utilizzare con MSBuild o con un progetto di setup fatto con Visual Studio 2005.

Per poter sfruttare completamente questo tool è necessario avere una buona conoscenza delle procedure d'installazione d'applicazioni .NET , nonché della sintassi che viene prodotta da Visual Studio 2005 per creare i file "manifest" XML menzionati in precedenza. L'unico svantaggio è che si abbia la leggera impressione che il deployment di un'applicazione sia sempre un problema da affrontare per ultimo, quando in realtà la difficoltà e le complicazioni che possono sorgere durante la distribuzione di un'applicazione particolarmente complessa dovrebbero porre quest'attività in primo piano. Questa considerazione scaturisce dal fatto che la documentazione del Bootstrapper corredata da relativi esempi è ancora, al momento della scrittura di quest'articolo, molto scarna.

Inoltre, la scarsa documentazione del prodotto stesso influenza non poco la capacità di essere produttivi immediatamente; tuttavia, una volta che sia ben chiaro lo scopo che ci si prefigge, il suo valore è senza dubbio elevato. Da notare, infine, che nella versione da noi provata il programma talvolta ha mostrato alcune inesattezze o errori imprevisti (anche se sempre ben gestiti), quindi la speranza è che venga rilasciata una nuova versione e che il progetto non sia giunto alla fine. Un'ultima nota: l'autore del BMG è David Guyer che fa parte del team di test di VB.NET, quindi si può stare tranquilli sulla qualità del software (o almeno si spera!).

Prodotto
Bootstrapper Manifest Generator

Url di riferimento
http://www.gotdotnet.com/workspaces/workspace.aspx?id=ddb4f08c-7d7c-4f44-a009-ea19fc812545

Stato Release
Stabile, ma in continua manutenzione

Semplicità d'uso ✦✦✦✦✧
La distribuzione di applicazione è un argomento complesso, e il programma ne richiede una certa conoscenza di base

Utilità ✦✦✦✦✦
Strumento fondamentale per la creazione di file "manifest" XML senza dover conoscere la sintassi a memoria.

Qualità prodotto ✦✦✦✦✧
Non sempre affidabile, nei test sul prodotto eseguiti ha talvolta mostrato qualche problema.

Qualità documentazione ✦✦✦✧✧
L'help dell'applicazione fornisce solo alcune indicazioni di base, per il resto è necessario provvedere autonomamente.

DayPilot

di Raffaele Di Natale

Un componente ASP.NET in C# per la gestione di un calendario giornaliero Outlook-like

Chi utilizza giornalmente Microsoft Outlook® ed ha familiarità con il suo calendario giornaliero apprezzerà senz'altro la possibilità di utilizzare questo componente open source all'interno di una propria web application. Infatti, il componente calendario standard, messo a disposizione in Visual Studio .NET, integrandosi con DayPilot, ci permette di gestire gli eventi programmati come se fossimo proprio in ambiente Outlook. DayPilot, fin dove è possibile, imita il comportamento di Outlook nella visualizzazione giornaliera.

Prerequisiti

DayPilot è un componente ASP.NET sviluppato in C# che si integra perfettamente in ambiente VisualStudio .Net 2003.

L'applicazione dimostrativa realizzata per questo articolo è stata eseguita su Windows XP SP2 con Internet Information Services [IIS] e .NET Framework 2.0. DayPilot è testato su Internet Explorer 5.0, 5.5 e 6.0 e FireFox 1.0 e 1.5.

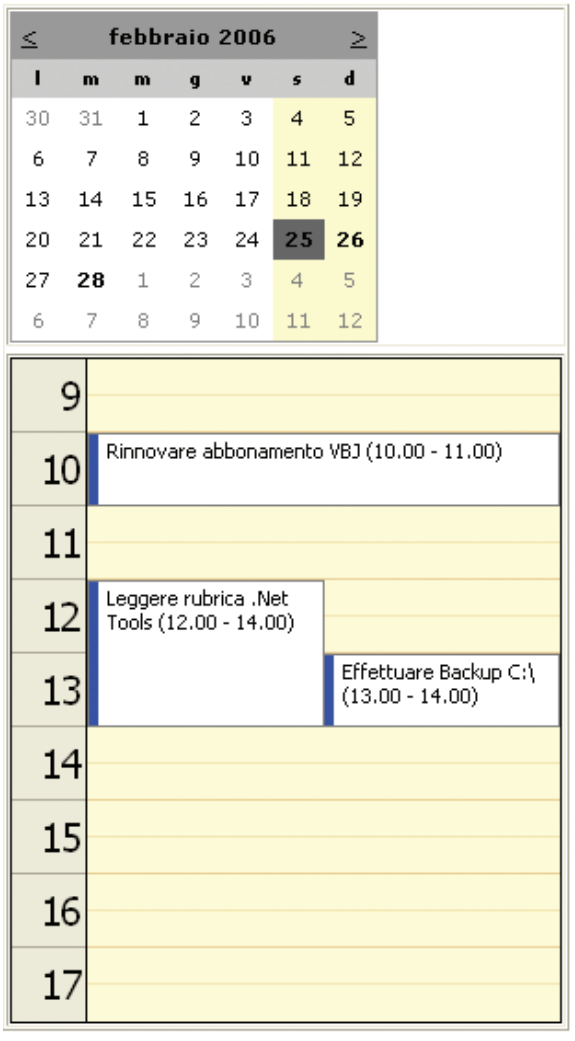
Installazione

Dal sito [1] è possibile scaricare sia i file contenenti il codice sorgente che i binari. Non esiste una vera e propria installazione del prodotto, in quanto è sufficiente aggiungere un riferimento all'assembly *DayPilot.dll* per poterlo utilizzare all'interno della nostra web application.

Se lo vogliamo avere a disposizione anche in altre applicazioni, e ritrovarlo fra gli altri componenti standard di Visual Studio .NET, basta cliccare sul menu "strumenti/aggiungi rimuovi elementi casella degli strumenti" e selezionare sempre DayPilot.dll.

In questo modo DayPilot apparirà tra gli altri componenti della casella degli strumenti, pronto per essere trascinato all'interno di una WebForm.

Day Pilot per VISUALBASIC JOURNAL



Descrizione delle funzionalità principali

DayPilot non è Outlook e quindi non ci aspettiamo di editare la descrizione degli eventi, né di effettuare il drag'n'drop degli stessi. Fatta questa premessa, c'è da dire che DayPilot si comporta in maniera soddisfacente e risolve impeccabilmente i problemi tipici di sovrapposizione di eventi o di eventi estesi su più giorni. Le principali caratteristiche di questo componente sono:

- Grafica perfettamente somigliante a quella utilizzata in Outlook
- Gli eventi possono essere importati da una sorgente dati esterna oltre ad essere definiti da codice

- Possibilità di impostare le ore lavorative e di nascondere quelle che non lo sono
- Possibilità di gestire il doppio click mediante JavaScript sia su un evento esistente che su un periodo libero

Mediante le proprietà del componente, oppure direttamente da codice, sono possibili ulteriori personalizzazioni del componente in maniera del tutto simile a quanto accade per il calendario giornaliero di Outlook: aumentare o diminuire la larghezza della pagina, variare l'altezza e la larghezza dell'area assegnata ad una singola ora, definire l'ora iniziale e l'ora finale del calendario, utilizzare un formato su 12 o 24 ore.

Utilizzare DayPilot in un'applicazione

Prima di integrare questo componente all'interno di una web application esistente o in fase di sviluppo, si consiglia di scaricare il codice della demo all'indirizzo [3]. Una versione online della demo può essere visionata direttamente all'indirizzo [4]. Dopo aver configurato il componente per essere utilizzato nell'IDE ed averlo trascinato dalla casella degli strumenti alla webform di destinazione, si può passare a personalizzare il calendario giornaliero direttamente da codice o dalla finestra delle proprietà del componente stesso.

L'aspetto saliente riguarda senz'altro l'inserimento degli eventi. Per iniziare è necessario osservare che DayPilot utilizza una *DataTable* o un *DataSet* opportunamente formattatti a partire dai quali prelevare le informazioni relative agli eventi che giornalmente devono essere visualizzati. In generale una *DataTable* deve prevedere almeno quattro colonne che potrebbero essere così definite:

- *Id*: identificativo univoco dell'evento
- *Start*: giorno e ora di inizio dell'evento
- *End*: giorno e ora di fine dell'evento
- *Name*: descrizione dell'evento

Questa operazione va effettuata utilizzando una *property* getData che restituirà *DataTable* popolata dagli inserimenti desiderati. Più dettagliatamente scriveremo:

```
protected DataTable getData
{
get
{
DataTable myDataTable;
myDataTable = new DataTable();
myDataTable.Columns.Add("start", typeof(DateTime));
myDataTable.Columns.Add("end", typeof(DateTime));
myDataTable.Columns.Add("name", typeof(string));
myDataTable.Columns.Add("id", typeof(string));

DataRow myDataRow;
myDataRow = myDataTable.NewRow();
myDataRow ["id"] = 0;
myDataRow ["start"] = Convert.ToDateTime("10:00").AddDays(1);
myDataRow ["end"] = Convert.ToDateTime("11:00").AddDays(1);
myDataRow ["name"] = "Rinnovare abbonamento VBJ";
myDataTable.Rows.Add(myDataRow);
return dt;
}
}
```

Il codice precedente, oltre ad instanziare una *DataTable* con le caratteristiche richieste, inserirà una riga contenente l'evento "Rinnovare abbonamento VBJ" che ha inizio alle ore 10:00 e fine alle 11:00 della data odierna di default. Bisognerà inoltre valorizzare alcuni campi del componente DayPilot in questo modo:

- *DataSource*: getData;
- *PkColumnName*: id
- *BeginColumnName*: start
- *EndColumnName*: end
- *NameColumnName*: name

Collegamento ad una sorgente dati

Un'operazione del tutto simile alla precedente può essere realizzata importando da una sorgente dati esterna i dati relativi agli eventi da visualizzare. Nel caso più semplice è possibile modificare la property getData integrando il codice relativo alla connessione al database esterno. Supponiamo di aver precedentemente preparato un database, in formato Access, denominato "Event.mdb" e formato da una sola tabella "Event" i cui campi sono: id (chiave primaria), start (Data), end (Data) e name (Testo). Pertanto:

```
protected DataTable getData
{
get
{
string connectionString = "Driver={Microsoft Access Driver
(*.mdb)};" + "DBQ=C:\\InetPub\\Daypilot\\Event.mdb;UID=;PWD=";
OdbcConnection myOdbcConnection = new OdbcConnection
                                        (connectionString);
OdbcCommand myOdbcCommand = myOdbcCommand.CreateCommand();
myOdbcCommand.CommandText = "SELECT * " + "FROM Event";
OdbcDataAdapter myDataAdapter = new OdbcDataAdapter();
myDataAdapter.SelectCommand = myOdbcCommand;
DataSet myDataSet = new DataSet();
myOdbcConnection.Open();
myDataAdapter.Fill(myDataSet, "Event");
myOdbcConnection.Close();
DataTable myDataTable;
myDataTable = new DataTable();
myDataTable = myDataSet.Tables["Event"];
myOdbcConnection.Close();
return myDataTable;
}
}
```

Come nel caso precedente è necessario impostare i campi *DataSource* con getData, *PkColumnName* con id, *BeginColumnName* con start, *EndColumnName* con end e *NameColumnName* con name. Ovviamente i parametri del componente vanno adattati ai nomi reali delle colonne della tabella del database. In **Figura 1** è visibile il risultato di questa integrazione in cui emerge, tra l'altro, la possibilità di sovrapporre gli eventi in perfetto stile Outlook.

Ulteriori personalizzazioni

DayPilot consente inoltre di configurare anche altri aspetti importanti quali per esempio:

- *TimeFormat*, per il formato 12/24 ore;
- *NonBusinessHour*, che consente di visualizzare o meno le ore non lavorative (se possibile);
- *JavaScriptEventAction*, che definisce il codice JavaScript da eseguire in seguito al doppio click su un evento visualizzato (dipende dal paramentro *EventClickHandling*);
- *JavaScriptFreeAction*, che definisce il codice JavaScript da eseguire in seguito al doppio cli-

ck su un periodo libero (dipende dal parametro *FreeTimeClickHandling*);

Conclusioni

DayPilot usa una grafica rassicurante per tutti coloro che usano già Outook per organizzare i propri eventi: è questa la forza, ma anche il limite del componente. Quante volte cliccheremo sull'evento per provare a modificarne il contenuto? O tenteremo di spostare un evento in un periodo diverso? DayPilot nasce con uno scopo ben preciso e cioè quello di visualizzare eventi già impostati, ma la bontà del progetto software porta a pensare che nelle prossime *release* l'autore inserirà un supporto AJAX che consentirà di eseguire anche queste operazioni.

Riferimenti

[1] http://www.daypilot.org
[2] http://www.daypilot.org/installation.html
[3] http://www.daypilot.org/download.html
[4] http://www.daypilot.org/live-demo.html

Prodotto
DayPilot

Url di riferimento
http://www.daypilot.org

Stato Release
v. 1.1

Semplicità d'uso ✦✦✦✦✧
Non trattandosi di un'applicazione, ma di un componente, qualche problema iniziale di integrazione può essere accettabile. La versione dimostrativa online e il relativo codice sorgente sono senz'altro di grande aiuto in questo senso.

Utilità ✦✦✦✦✦
Chi ha in mente di sviluppare una web application in cui si gestiscono eventi può trarre notevoli vantaggi dall'utilizzo di questo componente open source.

Qualità prodotto ✦✦✦✦✧
Si tratta di un componente ben strutturato e versatile il cui comportamento è risultato davvero soddisfacente nei test effettuati.

Qualità documentazione ✦✦✦✦✧
La documentazione è disponibile solo in inglese; è chiara, ma forse un po' troppo concisa. L'autore sembra puntare tutto sulla versione dimostrativa

Campagna abbonamenti 2006

	1 ANNO	2 ANNI
Computer Programming	€ 50,00 ☐	€ 90,00 ☐
DEV	€ 50,00 ☐	€ 90,00 ☐
VBJ	€ 44,00 ☐	€ 79,00 ☐
Login	€ 44,00 ☐	€ 79,00 ☐
CP+DEV	€ 90,00 ☐	€ 162,00 ☐
CP+VBJ	€ 85,00 ☐	€ 153,00 ☐
CP+Login	€ 85,00 ☐	€ 153,00 ☐
DEV+VBJ	€ 85,00 ☐	€ 153,00 ☐
DEV+Login	€ 85,00 ☐	€ 153,00 ☐
VBJ+Login	€ 79,00 ☐	€ 142,00 ☐
CP+DEV+VBJ	€ 129,00 ☐	€ 210,00 ☐
CP+DEV+Login	€ 129,00 ☐	€ 210,00 ☐
CP+VBJ+Login	€ 124,00 ☐	€ 209,00 ☐
DEV+VBJ+Login	€ 124,00 ☐	€ 209,00 ☐
CP+DEV+VBJ+Login	€ 150,00 ☐	€ 250,00 ☐

	1 ANNO
CP web	€ 42,00 ☐
DEV web	€ 42,00 ☐
VBJ web	€ 42,00 ☐
Login web	€ 42,00 ☐
2 web a scelta	€ 80,00 ☐
3 web a scelta	€ 90,00 ☐
4 web	€ 110,00 ☐

	1 ANNO
CP (carta+web)	€ 80,00 ☐
DEV (carta+web)	€ 80,00 ☐
VBJ (carta+web)	€ 74,00 ☐
Login (carta+web)	€ 74,00 ☐
CP+DEV (carta+web)	€ 150,00 ☐
CP+VBJ (carta+web)	€ 145,00 ☐
CP+Login (carta+web)	€ 145,00 ☐
DEV+VBJ (carta+web)	€ 145,00 ☐
DEV+Login (carta+web)	€ 145,00 ☐
VBJ+Login (carta+web)	€ 143,00 ☐
CP+DEV+VBJ (carta+web)	€ 190,00 ☐
CP+DEV+Login (carta+web)	€ 190,00 ☐
CP+VBJ+Login (carta+web)	€ 187,00 ☐
DEV+VBJ+Login (carta+web)	€ 187,00 ☐
CP+DEV+VBJ+Login (carta+web)	€ 200,00 ☐

1 ANNO
+ € 18,00 per CP
+ € 18,00 per DEV
+ € 10,00 per VBJ
+ € 10,00 per Login

2 ANNI
+ € 36,00 per CP
+ € 36,00 per DEV
+ € 20,00 per VBJ
+ € 20,00 per Login

I prezzi degli abbonamenti per l'estero sono maggiorati di spese di spedizione.

Gli abbonamenti decorrono dal primo numero raggiungibile. Per attivazioni retroattive contattaci al numero 0587/736460 o invia un'e-mail all'indirizzo abbonamenti@gruppoinfomedia.it.

INDIRIZZO DI SPEDIZIONE

CODICE CLIENTE _____

Nome/Società

Indirizzo

CAP Città Prov.

Telefono

Fax

E-Mail

PRIVACY
Si autorizza al trattamento dei dati personali ai sensi delle vigenti norme sulla privacy

FIRMA _____

MODALITA' DI PAGAMENTO

☐ Allego fotocopia del Bonifico Bancario effettuato sul **C/C 000010005424** CAB 71120 ABI 6370 Cin "M" - Cassa di Risparmio di Volterra - Agenzia di Ponsacco
☐ Allego fotocopia della ricevuta del versamento sul C/C Postale **N.14291561** intestato a "Gruppo Editoriale Infomedia S.r.l. - Ponsacco"
☐ Allego assegno bancario intestato a "Gruppo Editoriale Infomedia S.r.l." NON TRASFERIBILE
☐ Contrassegno (+ € 7,00 contributo spese postali)
☐ Autorizzo l'addebito dell'importo di € _____ sulla mia CARTASI'/VISA

N. _____ _____ _____ _____

Scad. _____ _____ mm/aa

Nome del Titolare _____

Data di nascita _____

Firma del Titolare _____

☐ Collegati al sito www.infomedia.it dove potrai effettuare il pagamento con carta di credito in modalità sicura (banca SELLA)

Garanzia di riservatezza - Gruppo Editoriale Infomedia garantisce la massima riservatezza dei dati da lei forniti e la possibilità di richiederne gratuitamente la rettifica o la cancellazione scrivendo a: Responsabile Dati - Gruppo Editoriale Infomedia Srl - Via Valdera P. 116 - 56038 Ponsacco (PI). Le informazioni custodite nel nostro archivio elettronico verranno trattate in conformità alla legge 196/03 sulla tutela dati personali.

L'IVA sul prezzo dell'abbonamento è assolta dall'Editore e non sussiste l'obbligo di emissione della fattura, ai sensi del D.M. 9 aprile 1993, art.1, comma 5; pertanto, ai fini contabili, farà fede la sola ricevuta di pagamento; perciò la fattura verrà emessa solo se esplicitamente richiesta al momento dell'ordine.

GRUPPO EDITORIALE INFOMEDIA S.R.L.
Via Valdera P. 116 - 56038 Ponsacco (Pi) - Tel. 0587/736460 - Fax 0587/732232
abbonamenti@gruppoinfomedia.it

SI, mi voglio iscrivere gratuitamente alla newsletter Infomedia che mi tiene aggiornato sui prossimi argomenti delle mie riviste di programmazione e mi informa delle offerte, le nuove uscite e le prossime pubblicazioni;
questo è il mio indirizzo di posta elettronica _____@_____
So che posso annullare la mia iscrizione in qualsiasi momento alla pagina http://www.infomedia.it/newsletter.htm

euro 7,50

36 racconti
per stuzzicare
la tua *abilità*
matematica

I rompicapo del Doktor Morb

pagg. 80
ISBN 8881500132
€ 7.50

WEB: http://book.infomedia.it
e-mail:book@infomedia.it
Tel. 0587/736460

GRUPPO EDITORIALE
INFOMEDIA
THE SOFTWARE SIDE OF YOUR MIND

NOVITÀ

euro 5,00

Un testo pratico e semplice per l'apprendimento di C#, il nuovo e rivoluzionario linguaggio di programmazione, concepito da Microsoft, per l'ambiente .NET. Con spiegazioni abilmente concepite, suggerimenti nati dalla profonda esperienza accumulata negli anni, ed esempi semplici ed efficaci, Baccan prende in esame gli aspetti salienti di C#, sia della versione 1.1 che della **nuovissima versione 2.0.**

corso di C#

Questo testo è l'evoluzione, riveduta e potenziata, del corso a puntate tenuto per rivista DEV di Infomedia. Sono state riscritte per intero delle parti, per renderle attuali, è stato fatto un lavoro di ricontrollo di tutti gli esempi e soprattutto è stata inserita la spiegazione di tutti i nuovi costrutti di C# 2.0.

pagg. 96
ISBN 8881500167
€ 5.00

GRUPPO EDITORIALE INFOMEDIA
THE SOFTWARE SIDE OF YOUR MIND

Tel. 0587/736460
e-mail:book@infomedia.it
WEB: http://book.infomedia.it