Net.Data: riferimenti all'ambiente di linguaggio

Codice documento N/A

Nota

Assicurarsi di consultare le indicazioni contenute in Appendice B, “Informazioni particolari” a pagina 79 prima di utilizzare le presenti informazioni e il prodotto ad esse relativo.

Quarta edizione (agosto 1998)

Questa pubblicazione potrebbe contenere imprecisioni tecniche o errori tipografici. Le correzioni relative saranno incluse nelle nuove edizioni della pubblicazione. L'IBM si riserva il diritto di apportare miglioramenti o modifiche al prodotto descritto nel manuale in qualsiasi momento e senza preavviso.

E' possibile che questo manuale contenga riferimenti o informazioni su prodotti (macchina o programmi) non ancora annunciati. Tali riferimenti o informazioni non possono significare in alcun modo che l'IBM Semea intende annunciare tali prodotti, programmi o servizi.

Richieste di ulteriori copie di questo prodotto o informazioni tecniche sullo stesso vanno indirizzate ad un rivenditore autorizzato o ad un rappresentante commerciale IBM.

E' possibile inviare eventuali commenti all'indirizzo riportato di seguito:

Selfin S.P.A.
Translation Assurance
Via F. Giordani, 7
80122 - Napoli

Tutti i commenti e i suggerimenti inviati potranno essere utilizzati liberamente dall'IBM e dalla Selfin e diventeranno esclusiva delle stesse.

Indice

Prefazione

Grazie per aver scelto Net.Data Versione 2, gli strumenti di sviluppo IBM per la creazione di pagine Web dinamiche. Net.Data consente di sviluppare rapidamente pagine Web a contenuto dinamico incorporando dati provenienti da diverse origini e utilizzando le capacità dei linguaggi di programmazione già conosciuti.

Net.Data Versione 2 fornisce prestazioni estremamente migliorate insieme a nuove funzioni di creazione e organizzazione delle soluzioni gestionali di Internet.

Informazioni su Net.Data

IBM Net.Data consente di creare pagine Web dinamiche tramite i dati provenienti da DBMS (database management system) relazionali e non, compreso DB2, IMS e database abilitati a ODBC e utilizzando applicazioni scritte in linguaggi di programmazione quali Java, JavaScript, Perl, C, C++ e REXX.

Net.Data può essere considerato come un processore macro eseguito come middleware su un server Web. E' possibile scrivere programmi applicativi Net.Data, chiamati macro, che Net.Data interpreta al fine di creare pagine Web dinamiche con contenuto personalizzato basato sulle immissioni dell'utente, sullo stato corrente dei database, sulla logica di gestione esistente e su altri fattori strutturati nella macro.

Una richiesta, in formato URL (uniform resource locator), viene trasmessa da un browser, come Netscape o Internet Explorer, su un server Web che la inoltra a Net.Data per l'esecuzione. Net.Data individua ed esegue la macro e crea una pagina Web personalizzata in base alle funzioni scritte. Queste funzioni possono:

- Inserire la logica di gestione all'interno di script di tipo Perl, di applicazioni C e C++ o di programmi REXX
- Accedere a database come DB2

Net.Data supporta interfacce standard come HTTP (HyperText Transfer Protocol) e CGI (Common Gateway Interface). HTTP viene utilizzato tra il browser e il server Web e CGI è utilizzato tra il server Web e Net.Data. Ciò consente di selezionare il browser preferito o il server web da utilizzare con Net.Data. Net.Data supporta anche FastCGI e le principali API del server Web su più sistemi operativi.

Informazioni sul manuale

Il presente manuale tratta gli ambienti di linguaggio Net.Data utilizzati per richiamare programmi o funzioni o origini dati come i database DB2, Oracle o Sybase dal file macro di Net.Data. Descrive ogni ambiente di linguaggio fornito da Net.Data e l'interfaccia di ambiente di linguaggio da utilizzare per strutturarne uno proprio.

Il presente manuale può fare riferimento a prodotti o programmi annunciati, ma non ancora disponibili.

Ulteriori informazioni comprendenti le macro di esempio di Net.Data, programmi dimostrativi e l'ultima versione del manuale è disponibile sui seguenti siti World Wide Web:

- http://www.software.ibm.com/data/net.data
- http://www.as400.ibm.com/netdata

A chi si rivolge il manuale

Gli utenti che creano macro di Net.Data possono utilizzare queste informazioni per acquisire conoscenze sulle capacità degli ambienti di linguaggio che Net.Data fornisce. Il manuale contiene anche informazioni indirizzate a quegli utenti che desiderano scrivere propri ambienti di linguaggio per Net.Data.

Per una migliore comprensione dei concetti trattati in questo manuale, è necessaria una certa familiarità con il linguaggio di programmazione C e con le informazioni contenute in *Net.Data Administration and Programming Guide* e in *Net.Data Reference*.

Esempi contenuti nel presente manuale

Gli esempi contenuti in questo documento sono stati estremamente semplificati per illustrare concetti specifici e non considerano tutti i possibili casi. Alcuni di essi sono solo frammenti che non funzionano isolatamente.

Informazioni sugli ambienti di linguaggio di Net.Data

Net.Data è strutturato in modo da consentire un nuovo linguaggio di programmazione e interfacce di database in una modalità *plug (di inserimento)*. Queste interfacce vengono chiamate ambienti di linguaggio e vi si accede come DLL o come librerie condivise. Gli ambienti di linguaggio forniscono l'accesso alle applicazioni e ai database che supportano le pagine Web dinamiche. Richiamando gli ambienti di linguaggio con le chiamate di funzione e con le istruzioni SQL, è possibile accedere alle funzioni e ai programmi di utilità che questi ambienti di linguaggio forniscono per essere utilizzati con le applicazioni gestionali. Ad esempio, è possibile accedere direttamente al database ODBC, utilizzare l'ambiente di linguaggio Perl per richiamare gli script Perl o richiamare l'ambiente di linguaggio delle applet di Java per eseguire applet Java.

Il file di inizializzazione Net.Data associa ogni nome di ambiente di linguaggio ad una DLL o a una libreria condivisa. Ogni ambiente di linguaggio deve supportare una serie standard di interfacce definite da Net.Data. Net.Data carica la DLL o la libreria condivisa specificata nel file di inizializzazione la prima volta che una funzione richiama un blocco FUNCTION specificando l'ambiente di linguaggio rilevato.

Net.Data analizza la macro Net.Data, conserva le variabili Net.Data, comunica con gli ambienti di linguaggio e formatta l'output in base alle specifiche dei blocchi REPORT e MESSAGE. L'ambiente di linguaggio supporta le interfacce definite per Net.Data, rende i parametri Net.Data accessibili al processore del linguaggio in modalità dipendenti dal linguaggio, richiama l'interpreter del linguaggio e riceve la variabile dall'interpreter del linguaggio in modalità dipendenti dal linguaggio.

Figura 1 a pagina viii illustra l'interazione di Net.Data con gli ambienti di linguaggio.

La gestione degli ambienti di linguaggio in un'applicazione Net.Data coinvolge due tipi di attività.

- L'uso di ambienti di linguaggio forniti da Net.Data per lo sviluppo di un'applicazione Net.Data.
- Lo sviluppo di un nuovo linguaggio o di ambienti di database per altri utenti da utilizzare quando vengono sviluppate applicazioni Net.Data.

Questo manuale è di supporto ad entrambe queste attività:

- “Ambienti di linguaggio forniti da Net-Data” a pagina 1 descrive gli ambienti di linguaggio forniti da Net.Data da utilizzare con le applicazioni Net.Data.
- “Ambienti di linguaggio non-IBM” a pagina 35 indica come creare nuovi ambienti di linguaggio e di database.

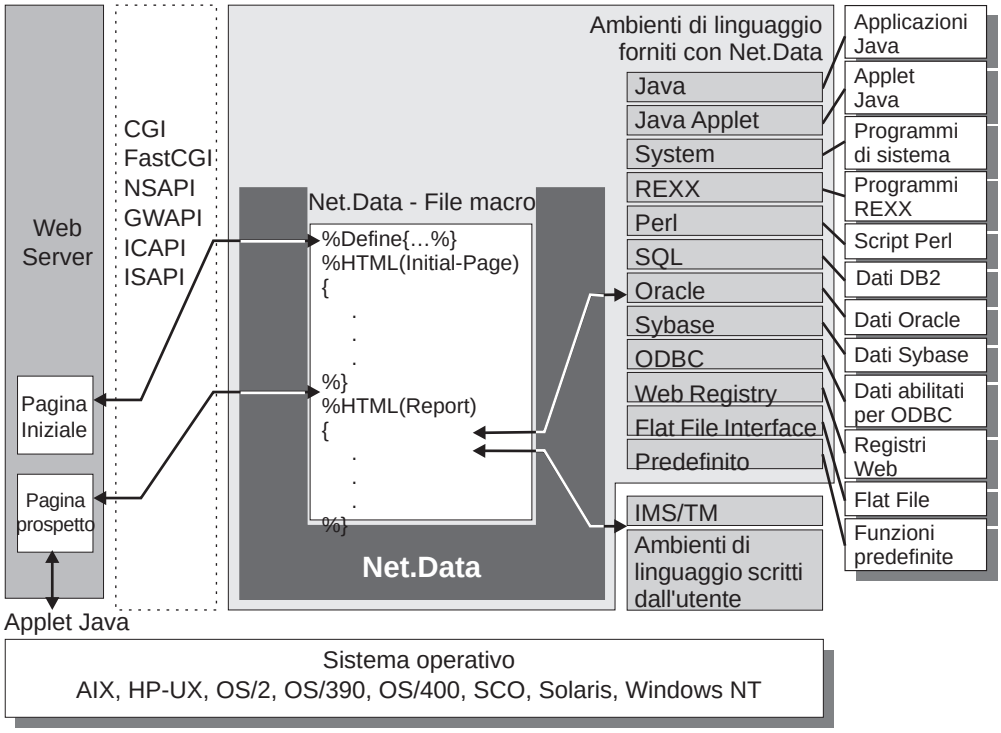

Figura 1. Net.Data e ambienti di linguaggio

Ambienti di linguaggio forniti da Net-Data

Net.Data fornisce diversi ambienti di linguaggio che consentono di trasferire informazioni tra le origini dei dati. Ad esempio, l'ambiente di linguaggio per SQL consente di trasferire interrogazioni SQL nativo ad un database. Allo stesso modo, l'ambiente di linguaggio REXX consente di richiamare programmi REXX.

Questa sezione descrive ogni ambiente di linguaggio e come configurare gli ambienti di linguaggio nel file di inizializzazione di Net.Data.

Panoramica sugli ambienti di linguaggio forniti da Net.Data

Net.Data fornisce alcuni ambienti di linguaggio, sebbene alcuni sistemi operativi non supportino tutti gli ambienti. Tabella 1 a pagina 4 elenca gli ambienti di linguaggio forniti dall'IBM. Per verificare se un ambiente di linguaggio è supportato dal proprio sistema operativo, consultare l'appendice di riferimento al sistema operativo *Riferimento di Net.Data*. Consultare il file README di Net.Data o la Directory del programma per i dettagli relativi alle istruzioni dell'ambiente di linguaggio che il sistema operativo utilizza.

Tabella 1. Ambienti di linguaggio di Net.Data

Ambiente di linguaggio	Istruzione Environment	Descrizione
Interfaccia di file non codificato	DTW_FILE	L'interfaccia FFI (flat file interface) fornisce le funzioni che supportano i file di testo che sono origini di dati.
IMS Web	HWS_LE	L'ambiente di linguaggio IMS Web consente di inoltrare una transazione IMS utilizzando IMS Web e di ricevere l'emissione della transazione sul server Web.
Applet Java	DTW_APPLET	L'ambiente di linguaggio delle applet Java consente di utilizzare le applet di Java nelle applicazioni di Net.Data. Per generare una tag di applet, è necessario fornire gli attributi della tag di applet insieme ad un elenco di parametri.
Applicazione Java	DTW_JAVAPPS	Net.Data supporta le applicazioni Java esistenti con l'ambiente di linguaggio Java.
ODBC	DTW_ODBC	L'ambiente di linguaggio ODBC esegue le istruzioni SQL tramite un'interfaccia ODBC per l'accesso a più sistemi di gestione database.
Oracle	DTW_ORA	L'ambiente di linguaggio Oracle fornisce l'accesso diretto ai dati Oracle.
Perl	DTW_PERL	L'ambiente di linguaggio Perl è in grado di interpretare gli script Perl interni specificati in un blocco FUNCTION di Net.Data o consente di elaborare script Perl esterni memorizzati in file separati.
REXX	DTW_REXX	L'ambiente di linguaggio REXX è in grado di interpretare i programmi REXX interni specificati in un blocco FUNCTION di Net.Data o consente di elaborare programmi REXX esterni memorizzati in file separati.
SQL	DTW_SQL	L'ambiente di linguaggio SQL viene utilizzato per eseguire le istruzioni SQL tramite DB2. I risultati dell'istruzione SQL possono essere restituiti in una variabile di tabella.
Sybase	DTW_SYB	L'ambiente di linguaggio Sybase fornisce l'accesso diretto ai dati Sybase.
System	DTW_SYSTEM	L'ambiente di linguaggio System supporta le chiamate ai programmi esterni identificati in un'istruzione EXEC in un blocco FUNCTION. L'ambiente di linguaggio System interpreta l'istruzione EXEC trasferendo il nome programma e i parametri al sistema operativo per l'elaborazione.
Registro Web	DTW_WEBREG	L'ambiente di linguaggio Registro Web fornisce le funzioni per la memorizzazione continua dei dati relativi all'applicazione.

Ogni ambiente di linguaggio richiede un'istruzione ENVIRONMENT nel file di inizializzazione ed una libreria condivisa o un file DLL nella directory `/lib` o `/dll` del server. Consultare il capitolo sulla configurazione in *Guida alla gestione e alla programmazione di Net.Data* per ulteriori informazioni.

Suggerimento: Eseguire una copia di backup del file di inizializzazione prima di eseguire modifiche.

Uso degli ambienti di linguaggio forniti da Net.Data

Le sezioni che seguono descrivono gli ambienti di linguaggio forniti da Net.Data e le procedure di impostazione e d'uso.

Ambiente di linguaggio dell'interfaccia di file non codificato

Se si decide di utilizzare i file non codificati (o file di testo non formattati) come origine dei dati, utilizzare FFI (flat file interface - interfaccia file non codificato) e le funzioni associate per aprire, chiudere, leggere, scrivere e cancellare i file sul server Web. E' necessario specificare un percorso per la variabile FFI_PATH nel file di inizializzazione.

Il supporto del linguaggio di file utilizza le funzioni FFI per leggere o scrivere i file sul server Web alla richiesta del client Web tramite il browser. FFI visualizza il file come file di registrazione, ogni record equivalente ad una riga in una variabile di tabella della macro di Net.Data ed ogni valore in un record equivalente ad un valore di campo in una variabile di tabella della macro di Net.Data. FFI legge i record da un file nelle righe di una tabella macro di Net.Data e scrive le righe da una tabella nei record.

Considerazioni sulla sicurezza

E' possibile specificare le funzioni dei file FFI che possono accedere tramite l'istruzione FFI_PATH nel file di inizializzazione di Net.Data. FFI ricerca solo i percorsi riportati nell'istruzione e quindi i file di altre directory sono protetti. Di seguito viene riportato un esempio di istruzione:

```
FFI_PATH
C:\public;.\;E:\WWW;E:\guest;A:
```

I percorsi riportati in FFI_PATH vengono ricercati dal primo all'ultimo. Net.Data utilizza la prima copia che trova. Se FFI_PATH non è presente nel file di inizializzazione, FFI tenta di trovare il file nella directory corrente. Il file di inizializzazione di Net.Data viene fornito senza FFI_PATH.

Consigli:

- Scegliere quali directory sono appropriate per essere utilizzate in operazioni di file non codificato. Queste directory vanno aggiunte a FFI_PATH per limitare ad esse la ricerca.
- Fare attenzione nel consentire ad altri utenti di eseguire DTWF_REMOVE o altre operazioni di esportazione nella macro per evitare che vengano eliminati o alterati i file con estensione `.dll` e `.cmd` che possono essere presenti nella directory corrente.
- Eseguire le procedure appropriate per salvaguardare i file del sistema tenendo sotto controllo le macro aggiunte al sistema.
- Non specificare un percorso in FFI_PATH che consenta ad utenti FTP sconosciuti di scrivere sul percorso. Potrebbe succedere che qualcuno possa inserire una macro di Net.Data sul sistema che consenta azioni non concesse precedentemente.
- Non aggiungere il percorso del file di inizializzazione di Net.Data a FFI_PATH.

Suggerimento per l'autorizzazione: Assicurarsi che il server Web abbia diritto di accesso ai file utilizzati dalle funzioni integrate FFI. Consultare la sezione sulla specifica dei diritti di accesso del server Web ai file di Net.Data nel capitolo relativo alla configurazione contenuto in *Guida alla gestione e alla programmazione di Net.Data* per ulteriori informazioni.

Le funzioni FFI integrate

Questa sezione descrive i suggerimenti sull'uso e le questioni da considerare quando vengono utilizzate le funzioni FFI integrate

Considerazioni generali

- E' possibile importare file di testo non formattati, ma la sintassi della macro di Net.Data viene interpretata da Net.Data e le tag HTML che possono essere presenti nel testo vengono utilizzate per formattare il testo tramite il browser.
- I parametri FFI sono sensibili al maiuscolo-minuscolo solo se lo è il sistema operativo.

Directory corrente

- La directory corrente per Net.Data dipende dalla configurazione del server Web. Se si sta utilizzando CGI, la directory corrente è la directory da cui è in esecuzione Net.Data, che normalmente è `\www\cgi-bin`. Se si sta utilizzando un'API del server Web, la directory corrente può variare. Per scrivere la directory corrente, Net.Data (o l'ID utente associato con il sottoprocesso o il processo che esegue Net.Data) deve avere l'accesso alla scrittura. Se l'instradamento predefinito della richiesta del server o la mappatura della risorsa viene modificata, anche la directory corrente può essere modificata.
- Il modo consigliato per specificare la directory corrente è quello di utilizzare i percorsi assoluti per l'istruzione FFI_PATH e per il parametro FILENAME, specialmente se si sta utilizzando un'API del server Web. Tutte le directory e le sottodirectory riportate nel percorso devono essere definite nell'istruzione del percorso FFI_PATH nel file di inizializzazione o Net.Data non riuscirà a trovare il file. Ad esempio, il percorso che segue richiede che la directory e le sottodirectory `/u/user/mydir/` vengano riportate in FFI_PATH.

  ```
  filename="/u/user/mydir/myfile.txt"
  ```

  Se viene specificato solo il nome file, come nell'esempio che segue.

  ```
  filename="myfile.txt"
  ```

  Net.Data concatena tutti i nomi di directory nel percorso per FFI_PATH e ricerca prima il file. Se non è possibile rilevare il file, Net.Data lo ricerca nella directory corrente. Se il file non è presente nella directory corrente, nella directory corrente viene creato un file con il nome file specificato. Se Net.Data non dispone dell'accesso alla scrittura per la directory corrente, si verifica un errore. Non utilizzare la seguente sintassi:

  ```
  filename="/myfile.txt"
  ```

parametro DELIMITER

- Il delimitatore è un indicatore o separatore che FFI utilizza quando il file viene diviso in parti (come le colonne di una riga) secondo la trasformazione richiesta.

- Per le operazioni di lettura, il delimitatore separa il contenuto del file in righe e colonne di una tabella. Per le operazioni di scrittura, il delimitatore indica la fine di un valore in una riga e in una colonna di una tabella.

  Net.Data trasferisce il delimitatore a FFI come una stringa macro di Net.Data e non comprende un carattere null alla fine dei caratteri a meno che non venga esplicitamente riportato nel parametro DELIMITER. Per utilizzare un carattere null nel delimitatore, specificare nel parametro DELIMITER una barra retroversa ed uno zero in doppi apici, “/0”, al posto di una stringa vuota utilizzando due doppi apici, “””. Se viene specificato ASCIITEXT, Net.Data utilizza il carattere nuova riga come delimitatore ed ignora qualsiasi delimitatore richiesto.

- Modifiche non desiderate a un file possono verificarsi se viene utilizzato un delimitatore per le operazioni di scrittura e uno di lettura. In questo caso, Net.Data scrive il file con il nuovo delimitatore.

- La lunghezza massima di un delimitatore è 256 caratteri.

FFI_PATH

- I percorsi contenuti in FFI_PATH devono contenere caratteri validi stampabili. FFI non consente percorsi che includano un punto interrogativo (?) o doppi apici (“”).

- Le sottodirectory dei percorsi riportate in FILENAME non vengono ricercate a meno che non venga esplicitamente specificato in FFI_PATH. Specificare tutte le directory e le sottodirectory contenute in FFI_PATH utilizzate con il parametro *filename* nel file della macro. I seguenti esempi visualizzano le istruzioni di percorso raccomandate:

  Esempio 1: Specifica un percorso assoluto che riporta tutte le directory e le sottodirectory

  ```
  filename="/u/usr/mydir/myfile.txt"
  ```

  Net.Data ricerca i percorsi consentiti in FFI_PATH; se il percorso assoluto assegnato al parametro FILENAME non è corretto o non è disponibile, Net.Data ricerca la directory corrente e se il nome file non viene trovato emette un messaggio di avvertenza.

  Esempio 2: Specifica un nome file nella directory corrente

  ```
  filename="myfile".txt
  ```

  Net.Data crea nuovi file nella directory corrente. Se Net.Data non dispone dell'accesso a creare file nella directory, Net.Data (o l'ID utente associato al sottoprocesso o al processo su cui Net.Data è in esecuzione) emette un'avvertenza.

Funzione DTWF_SEARCH

- La tabella riportata per DTWF_SEARCH contiene tre colonne. Le prime due colonne contengono il numero colonna e il numero riga in cui è stata rilevata la corrispondenza; l'ultima colonna contiene il valore di colonna contenente i caratteri specificati nel parametro *SearchFor*. Ad esempio, se la quarta riga contiene i caratteri corrispondenti nella colonna tre, la tabella riportata contiene una riga con il numero `4` nella prima colonna per indicare la riga del file da cui proviene; essa presenta il numero `3` nella seconda colonna ad indicare quale colonna del file contiene una corrispondenza; e presenta il valore di colonna completo nella terza colonna.

- Il parametro *SearchFor* non può includere il contenuto del parametro delimitatore.

Parametri STARTROW e ROWS

- Per le funzioni DTWF_DELETE, DTWF_INSERT, DTWF_UPDATE e DTWF_WRITE, se viene specificato un valore *StartRow* maggiore dell'ultima riga, *StartRow* viene modificato per indicare l'ultima riga e viene riportato un errore.
- Per le funzioni DTWF_READ e DTWF_SEARCH, il valore *Rows* viene restituito come numero di righe della tabella.

Parametro TABLE

- La lunghezza massima di una riga in una tabella FFI è di 16383 caratteri. Questo limite include un carattere null per ogni colonna nella tabella della macro di Net.Data.

Parametro TRANSFORM

- Questo parametro indica come il file viene suddiviso in parti in base alle righe e alle colonne di una tabella della macro di Net.Data. Ad esempio, ASCIITEXT transform significa che ogni riga del file corrisponde ad una riga di una tabella della macro di Net.Data e questa tabella contiene una sola colonna. DELIMITED transform significa che i caratteri contenuti nella riga vengono analizzati per rilevare DELIMITER e DELIMITER diventa il contenuto della colonna successiva.
- Un carattere di nuova riga del file indica la fine di una riga di una tabella della macro di Net.Data per ASCIITEXT e DELIMITED transform.

Blocco file

- I file non vengono bloccati a meno che non vengano aperti con DTWF_OPEN. Se un file non viene bloccato, esso può essere modificato tra il momento della lettura e quello dell'aggiornamento. Ciò può risultare nella perdita delle precedenti modifiche. Utilizzando DTWF_OPEN il file viene aperto durante l'esecuzione della macro utilizzando il meccanismo di blocco del sistema di file.

DTWF_APPEND

- Il contenuto corrente di un file influisce sui risultati dell'operazione DTWF_APPEND e, in modo particolare, sul contenuto dell'ultima colonna dell'ultima riga. Se una nuova riga segue il valore dell'ultima colonna dell'ultima riga del file, i dati accodati vengono posizionati in una nuova riga. In caso contrario, i dati accodati diventano parte dell'ultima riga del file.

Ambiente di linguaggio IMS Web

L'ambiente di linguaggio IMS Web è parte di una soluzione end-to-end completa per l'esecuzione delle transazioni IMS in ambiente World Wide Web. L'ambiente di linguaggio IMS Web fornisce:

- Una macro di Net.Data con:
  - L'HTML utilizzato per immettere i dati di immissione transazione
  - Un blocco FUNCTION di Net:Data che richiama l'ambiente di linguaggio IMS Web

  - L'HTML che visualizza l'emissione della transazione
- Una DLL di transazione o libreria condivisa richiamata dall'ambiente di linguaggio IMS Web

Limitazione: L'ambiente di linguaggio IMS Web di Net.Data è supportato solo quando Net.Data viene eseguito come applicazione CGI. Non è supportato da Net.Data con ICAPI.

Il programma IMS Web Studio genera un codice per la DLL e per la macro ed un file MAK per creare la DLL o la libreria condivisa, dall'origine MFS (Message Format Service) per la transazione. Una volta creato un modulo eseguibile della DLL, la DLL e il file della macro vengono spostati sul server Web su cui è in esecuzione Net.Data. La transazione è pronta per essere eseguita in un ambiente Web.

Per utilizzare l'ambiente di linguaggio IMS Web:

1. Installare il componente Runtime di IMS Web sul server Web su cui è in esecuzione Net.Data. Per informazioni relative al componente Runtime di IMS Web consultare *IMS Web User's Guide* su:

   `http://www.software.ibm.com/data/ims/about/imsweb/document/index.html`

2. Creare il file di transazione DLL.
   a. Generare i file della macro C++, MAK e Net.Data per la transazione con il programma IMS Web Studio.
   b. Se si esegue Net.Data su un sistema operativo diverso da quello su cui viene eseguito IMS Web Studio, spostare i file DLL di origine su una macchina IMS Web per il sistema operativo di destinazione. Ad esempio, se si esegue IMS Web Studio su Windows NT e la piattaforma di destinazione è AIX o OS/390, spostare l'origine per la DLL di transazione su una macchina IMS Web in ambiente AIX or OS/390, rispettivamente.
   c. Creare il modulo eseguibile della DLL della transazione utilizzando il file MAK generato.
3. Copiare il file DLL della transazione generato (DTW*proj*.dll) e il file della macro di Net.Data macro (DTW*proj*.d2w) sul server Web.
   a. Posizionare la macro in una directory da cui Net.Data richiama la macro. Consultare l'istruzione MACRO_PATH nel capitolo sulla configurazione del manuale *Guida alla gestione e alla programmazione di Net.Data.*
   b. Posizionare la DLL della transazione o la libreria condivisa in una directory da cui il server Web richiama le DLL o le librerie condivise.
4. Utilizzare il collegamento nel file di esempio generato dal programma IMS Web Studio, DTW*proj*.htm, per modificare un file HTML nella struttura ad albero HTML del server Web. E' possibile utilizzare poi li collegamento per richiamare Net.Data e visualizzare il modulo HTML di immissione per la transazione su un browser Web. Completare l'immissione della transazione e selezionare il pulsante INOLTRO sul modulo per eseguire la transazione e ricevere l'emissione sul browser Web.

IMS Web utilizza il collegamento OTMA (Open Transaction Manager Access) IMS TCP/IP per le comunicazioni tra il server Web e gli ambienti IMS. Consultare l'home page IMS Web per ulteriori informazioni:

`http://www.software.ibm.com/data/ims/about/imsweb`

Ambiente di linguaggio delle applet Java

L'ambiente di linguaggio delle applet Java consente di generare tag HTML per le applet di Java nelle applicazioni di Net.Data. Quando viene richiamato l'ambiente di linguaggio applet di Java, specificare il nome dell'applet e trasferire i parametri necessari all'applet. L'ambiente di linguaggio elabora la macro, genera le tag dell'applet HTML e poi esegue l'applet sul browser Web.

Inoltre Net.Data fornisce una serie di interfacce da utilizzare per accedere ai parametri della tabella. Queste interfacce sono contenute nella classe DTW_Applet.class.

Le sezioni che seguono descrivono come utilizzare l'ambiente di linguaggio applet di Java per eseguire le applet di Java.

Creazioni di applet Java

Prima di utilizzare l'ambiente di linguaggio applet di Java di Net.Data, è necessario stabilire le applet da utilizzare e scriverle, se necessario. Consultare la documentazione Java per ulteriori informazioni.

Creazione delle tag dell'applet

Specificare una chiamata alla variabile di linguaggio dell'applet con una chiamata di funzione Net.Data. Non è necessaria alcuna dichiarazione per la chiamata di funzione. Utilizzare la sintassi che segue per la chiamata di funzione:

```
@DTWA_AppletName(parm1, parm2, ..., parmN)
```

- DTWA_ identifica la chiamata di funzione presso l'ambiente di linguaggio dell'applet.
- NomeApplet è il nome dell'applet per la quale vengono generate le tag.
- parm1 - parmN sono i parametri usati per generare le tag PARAM.

Per scrivere un file della macro che genera le tag dell'applet:

1. Definire i parametri richiesti dall'applet nella sezione DEFINE del file della macro. Questi parametri includono gli attributi della tag dell'applet, le variabili Net.Data ed i parametri di tabella Net.Data necessari all'applet. Ad esempio:

   ```
   %define{
   DATABASE = "celdial"                  <=variabile Net.Data: nome database
   MyGraph.codebase = "/netdata-java"    <=Parametro applet obbligatorio
   MyGraph.height = "200"                <=Parametro applet obbligatorio
   MyGraph.width = "400"                 <=Parametro applet obbligatorio
   MyTitle = "Titolo"                    <=Variabile Net.Data: nome pagina Web
   MyTable = %TABLE(all)                 <=Tabella per i risultati dell'interrogazione
   %}
   ```

2. Facoltativo: Specificare un'interrogazione al database per generare un risultato impostato come immissione per un'applet. Ciò risulta utile quando si sta utilizzando un'applet che genera una scheda o una tabella. Ad esempio:

   ```
   %FUNCTION(DTW_SQL) mySQL(OUT table){
   select name, age from ibmuser.guests
   %}
   ```

3. Specificare la chiamata di funzione nella macro di Net.Data per richiamare l'ambiente di linguaggio applet di Java e richiamare l'applet. La chiamata di fun-

zione specifica il nome dell'applet e i parametri da trasferire all'ambiente di linguaggio. Questi parametri includono gli attributi della tag dell'applet, le variabili Net.Data e la tabella Net.Data o i parametri di colonna necessari all'applet.

Ad esempio:

```
%HTML(report){<=Inizion blocco HTML
@mySQL(MyTable) <=Chiamata alla funzione SQL
mySQL
@DTWA_MyGraph( MyTitle, DTW_COLUMN(ages) MyTable) <=Chiamata funzione applet
%}
```

4. Richiamare Net.Data ed eseguire il file della macro. Consultare *Guida alla gestione e alla programmazione di Net.Data* per i dettagli su come richiamare Net.Data.

Attributi della tag dell'applet

E' possibile specificare gli attributi delle tags dell'applet in un punto qualsiasi della macro di Net.Data. Net.Data sostituisce tutte le variabili di formato *AppletName.attribute* nella tag dell'applet come attributi. La sintassi per la definizione di un attributo di una tag dell'applet è mostrato di seguito:

```
%define AppletName.attribute = "value"
```

Tali attributi sono richiesti per tutte le applet:

- *codebase:* Ubicazione dell'applet, identificata dalla URL.
- *height:* Altezza dell'applet in pixel.
- *width:* Larghezza dell'applet in pixel.

Ad esempio, se l'applet è denominata MyGraph, è possibile definire gli attributi richiesti come mostrato di seguito:

```
%DEFINE{
MyGraph.codebase = "/netdata-java/"
MyGraph.height = "200"
MyGraph.width = "400"
%}
```

Non è necessario che l'assegnazione sia presente in una sezione DEFINE. E' possibile impostare il valore con la funzione DTW_ASSIGN. Se non viene definita alcuna variabile per *NomeApplet.code*, Net.Data aggiunge un parametro *code* predefinito per la tag dell'applet. Il valore del parametro *code* è *NomeApplet.class*, in cui *NomeApplet* è il nome dell'applet.

Parametri della tag dell'applet

Nella chiamata di funzione è necessario definire un elenco di parametri da trasferire all'ambiente di linguaggio dell'applet di Java. E' possibile trasferire i parametri che includono:

- Variabili Net.Data (comprese le variabili LIST)
- Tabelle Net.Data
- Colunne delle tabelle Net.Data

Quando viene trasferito un parametro, Net.Data crea una tag PARAM dell'applet di Java nell'emissione HTML con il nome e il valore assegnato al parametro. Non è possibile trasferire string literals o risultati di chiamate di funzione.

Parametri di variabile di Net.Data: E' possibile utilizzare le variabili di Net.Data come parametri. Se viene definita una variabile nel blocco DEFINE della macro e il valore della variabile viene trasferito nella chiamata di funzione DTWA_*NomeApplet*, Net.Data genera una tag PARAM che ha lo stesso nome e lo stesso valore della variabile. Ad esempio, nel caso della seguente istruzione macro:

```
%define{

...

MyTitle = "This is my Title"
%}

%HTML (report){
@DTWA_MyGraph( MyTitle, ...)
%}
```

Net.Data fornisce la seguente tag PARAM dell'applet:

```
<param name = 'MyTitle' value = "This is my Title" >
```

Parametri di tabella di Net.Data:

Net.Data genera automaticamente una tag PARAM con il nome DTW_NUMBER_OF_TABLES ogni volta che viene chiamato l'ambiente di linguaggio applet di Java, specificando se la chiamata di funzione ha trasferito variabili di tabella. Il valore è il numero delle variabili di tabella che Net.Data utilizza nella funzione. Se non vengono specificate variabili di tabella nella chiamata di funzione, viene generata la tag che segue:

```
<param name = "DTW_NUMBER_OF_TABLES" value = "0" >
```

E' possibile trasferire una o più variabili di tabella come parametri sulla chiamata di funzione. Se viene specificata una variabile di tabella di Net.Data su una chiamata di funzione DTWA_*NomeApplet*, Net.Data genera le seguenti tag PARAM:

Tag parametro nome tabella:

Questa tag indica i nomi delle tabelle da trasferire. La tag presenta la seguente sintassi:

```
<param name = 'DTW_TABLE_i_NAME' value = "tname" >
```

In cui `i` è il numero della tabella basata sulla chiamata di funzione e *tname* è il nome della tabella.

Tag del parametro di specifica della riga e della colonna:

Le tag PARAM vengono generate per specificare il numero di righe e di colonne per una tabella. La tag presenta la seguente sintassi:

```
<param name = 'DTW_tname_NUMBER_OF_ROWS' value = "righe" >
<param name = 'DTW_tname_NUMBER_OF_COLUMNS' value = "col" >
```

In cui il nome della tabella è *tname*, *righe* è il numero di righe della tabella e *col* è il numero di colonne della tabella. Questa coppia di tag

viene generata per ogni tabella univoca specificata nella chiamata di funzione.

Tag del parametro valore della colonna:

Questa tag PARAM indica il nome per una particolare colonna. La tag presenta la seguente sintassi:

```
<param name = 'DTW_tname_COLUMN_NAME_j' value = "cname" >
```

In cui il nome della tabella è *tname*, *j* è il numero della colonna e *cname* è il nome della colonna.

Tag del parametro valore della riga:

Questa tag PARAM indica i valori per una particolare riga/colonna. La tag presenta la seguente sintassi:

```
<param name = 'DTW_tname_cname_VALUE_k' value = "val" >
```

In cui il nome tabella è *tname*, *cname* è il nome colonna, *k* è il numero riga e *val* è il valore che corrisponde al valore della riga e della colonna corrispondenti.

Parametri di colonna di tabella: E' possibile trasferire una colonna di una tabella come un parametro su una chiamata di funzione per generare le tag per una colonna specifica. Net.Data genera le tag dell'applet corrispondenti solo per la colonna specificata. Un parametro della colonna della tabella utilizza la seguente sintassi:

```
@DTWA_AppletName(DTW_COLUMN( x )Table)
```

In cui *x* è il nome o il numero della colonna nella tabella.

I parametri della colonna della tabella utilizzano le stesse tag dell'applet definite per i parametri della tabella.

Testo alternativo per la tag dell'applet su browser non abilitati a Java

La variabile DTW_APPLET_ALTTEXT indica il testo da visualizzare su browser che non supportano Java o su cui il supporto Java non è attivato. Ad esempio, la seguente definizione di variabile:

```
%define DTW_APPLET_ALTTEXT = "<P>Il browser non è abilitato per Java."
```

produce i seguenti tag HTML e testo:

```
<P>Il browser non è abilitato per Java.<BR>
```

Se la variabile non è definita, non verrà visualizzato alcun testo alternativo.

Esempio di applet Java

L'esempio che segue illustra un file della macro Net.Data che richiama l'ambiente di linguaggio applet di Java e la tag dell'applet che l'ambiente di linguaggio genera.

Il file della macro di Net.Data contiene le seguenti chiamate di funzione per l'ambiente di linguaggio dell'applet di Java:

```
%define{
DATABASE = "celdial"
DTW_APPLET_ALTTEXT = "<P>Il browser non è abilitato per Java."
DTW_DEFAULT_REPORT = "no"
MyGraph.codebase = "/netdata-java/"
MyGraph.height = "200"
MyGraph.width = "400"
MyTitle = "This is my Title"
%}
%FUNCTION(DTW_SQL) mySQL(OUT table){
select name, age from ibmuser.guests
%}
%HTML (report){
@mySQL(MyTable)
@DTWA_MyGraph( MyTitle, DTW_COLUMN(ages) MyTable )
%}
```

Le righe della macro di Net.Data nella sezione DEFINE specificano la prima riga della tag dell'applet:

```
MyGraph.codebase = "/netdata-java/"
MyGraph.height = "200"
MyGraph.width = "400"
```

L'ambiente di linguaggio genera una tag di applet con i seguenti attributi:

```
<applet code = 'MyGraph.class'
codebase = '/netdata-java/' width = '400'
height = '200' >
```

Net.Data restituisce i risultati dell'interrogazione SQL dalla sezione SQL del file della macro di Net.Data nella tabella dei risultati `MyTable`. Questa tabella viene specificata nella sezione DEFINE:

```
MyTable = %TABLE(all)
```

La chiamata all'applet viene specificata nella sezione HTML:

```
@DTWA_MyGraph( MyTitle, DTW_COLUMN(ages) MyTable )
```

In base ai parametri contenuti nella chiamata di funzione, Net.Data genera la tag completa dell'applet contenente le informazioni sulla tabella dei risultati, come il numero delle colonne, il numero delle righe restituite e le righe dei risultati. Net.Data genera una tag di parametro per ogni cella nella tabella di risultati come riportato nel seguente esempio:

```
param name = 'DTW_MyTable_ages_VALUE_1' value = "35">
```

Il nome parametro, *DTW_MyTable_ages_VALUE_1*, specifica la cella della tabella (riga 1, colonna `ages`) nella tabella `MyTable` il cui valore è 4. La parola chiave, DTW_COLUMN nella chiamata di funzione all'applet, specifica che è importante solo la colonna ages della tabella che risulta, MyTable, di seguito riportata:

```
@DTWA_MyGraph( MyTitle, DTW_COLUMN(ages) MyTable )
```

L'emissione che segue mostra la tag di applet completa che Net.Data genera per l'esempio:

```
<applet code = 'MyGraph.class'
codebase = '/netdata-java/' width = '400' height = '200' >
<param name = 'MyTitle' value = "This is my Title" >
<param name = 'DTW_NUMBER_OF_TABLES' value = "1" >
<param name = 'DTW_TABLE_1_NAME' value = "MyTable" >
<param name = 'DTW_MyTable_NUMBER_OF_ROWS' value = "5" >
<param name = 'DTW_MyTable_NUMBER_OF_COLUMNS' value = "1" >
<param name = 'DTW_MyTable_COLUMN_NAME_1' value = "ages" >
<param name = 'DTW_MyTable_ages_VALUE_1' value = "35">
<param name = 'DTW_MyTable_ages_VALUE_2' value = "32">
<param name = 'DTW_MyTable_ages_VALUE_3' value = "31" >
<param name = 'DTW_MyTable_ages_VALUE_4' value = "28" >
<param name = 'DTW_MyTable_ages_VALUE_5' value = "40" >
<P>Il browser non è abilitato per Java.<BR>
</applet>
```

Uso dell'interfaccia applet Java di Net.Data

Net.Data fornisce una serie di interfacce in una classe denominata DTW_Applet.class, che può essere utilizzate con le applet Java per elaborare tag PARAM generate per variabili di tabella. E' possibile creare un'applet che estenda tale interfaccia per richiamare le routine dall'applet.

Net.Data fornisce queste interfacce:

- **int GetNumberOfTables()** restituisce il numero di tabelle rilevate nella tag dell'applet.
- **String [] GetTableNames()** restituisce un elenco dei nomi tabella rilevati nella tag dell'applet.
- **int GetNumberOfColumns(String table_name)** restituisce il numero di colonne nella tabella table_name.
- **int GetNumberOfRows(String table_name)** restituisce il numero di righe nella tabella table_name.
- **String[] GetColumnNames(String table_name)** restituisce i nomi delle colonne nella tabella table_name.
- **String[][] GetTable(String table_name)** restituisce una serie bidimensionale di stringhe che contengono i valori delle righe e delle colonne della tabella.

Per accedere alle interfacce, utilizzare la parola chiave EXTENDS nel codice applet per la sottoclasse dell'applet dalla classe DTW_APPLET, come riportato nel seguente esempio:

```
import java.io.*;
import java.applet.Applet;

public class myDriver extends DTW_Applet
    {
    public void init()
        {
        super.init();

        if (GetNumberOfTables() > 0)
            {
            String [] tables = GetTableNames();
            printTables(tables);
            }
        }

    private void printTables(String[] tables)
        {
        String nome_tabella;

        for (int i = 0; i < tables.length; i++)
            {
            table_name = tables[i];
            printTable(table_name);
            }
        }

    private void printTable(String nome_tabella)
        {
        int nrows = GetNumberOfRows(nome_tabella);
        int ncols = GetNumberOfColumns(nome_tabella);

        System.out.println("Table: " + table_name + " has " + ncols + " columns and
                            " + nrows + " rows.");

        String [] col_names = GetColumnNames(nome_tabella);

        System.out.println("-----------------------------------------------------");

            for (int i = 0; i < ncols; i++)
            System.out.print("   " + col_names[i] + "   ");
        System.out.println("\n-----------------------------------------------------");

        String [][] mytable = GetTable(nome_tabella);

        for (int j = 0; j < nrows; j++)
            {
            for (int i = 0; i < ncols; i++)
                System.out.print("   " + mytable[i][j] + "   ");

            System.out.println("\n");
            }
        }
    }
```

Ambiente di linguaggio dell'applicazione Java

Net.Data supporta le applicazioni Java esistenti con l'ambiente di linguaggio Java. Con il supporto per le applet e i metodi (o per le applicazioni) Java, è possibile accedere a DB2 tramite l'API JDBC* (Java Database Connectivity) API.

I dettagli su JDBC sono disponibili sui seguenti siti Web:

- IBM Software dispone di JDK 1.1 o successivo, obbligatorio per utilizzare JDBC con Net.Data:

  `http://www.software.ibm.com/data/db2/jdbc/`

- JavaSoft contiene altri driver JDBC, documentazione su API JDBC e gli ultimi aggiornamenti di JDBC:

  `http://splash.javasoft.com/jdbc/`

L'ambiente di linguaggio Java fornisce un'interfaccia del tipo RPC (Remote Procedure Call). E' possibile emettere le chiamate di funzione Java dal file della macro di Net.Data con stringhe Net.Data come parametri e la funzione di Java richiamata può restituire una stringa. E' necessario utilizzare Net.Data Live Connection quando viene utilizzato l'ambiente di linguaggio Java (consultare il capitolo sulle prestazioni in *Guida alla gestione e alla programmazione di Net.Data* per ulteriori informazioni su Live Connection). Per utilizzare l'ambiente di linguaggio Java è necessario completare i seguenti passi. Queste procedure vengono descritte in dettaglio nelle sezioni che seguono.

1. Scrivere le funzioni Java.
2. Creare un cliette Net.Data per tutte le funzioni Java (i cliette di Net.Data avviano Java Virtual Machine in cui viene eseguita la funzione Java).
3. Definire un'istruzione cliette nel file di configurazione di Live Connection.
4. Avviare Connection Manager.
5. Eseguire il file della macro di Net.Data che richiama l'ambiente di linguaggio Java.

Ogni volta che vengono introdotte nuove funzioni Java è necessario ricreare il cliette Java.

Struttura file dell'ambiente di linguaggio Java

Net.Data crea diverse directory durante l'installazione di Net.Data. Queste directory comprendono i file necessari a creare le funzioni Java, definire il cliette ed eseguire la macro con il linguaggio di ambiente Java:

- Una funzione di esempio Java chiamata `UserFunctions.java`.
- Un file di esempio chiamato `makeClas`. Quando viene eseguito, questo file crea una classe cliette di Net.Data per la funzione Java.
- Un file di esempio chiamato `launchjv` utilizzato dal cliette di Net.Data per avviare Java Virtual Machine ed eseguire la funzione Java.

Tabella 2 a pagina 20 descrive la directory e i nomi file per i file del sistema operativo.

Tabella 2. File utilizzati per creare le funzioni Java

Sistema operativo	Nome file	Directory
OS/2	UserFunctions.java	javaapps
	launchjv.com	connect
Windows NT	UserFunctions.java	javaclas
	makeClas.bat	javaclas
	launchjv.bat	connect
UNIX	UserFunctions.java	javaapps
	launchjv	javaapps

Creazione di una funzione Java

Modificare il file di esempio della funzione Java `UserFunctions.java` o creare un nuovo file basato sul seguente esempio, chiamato `myfile.java`:

```
====================myfile.java====================
import mypackage.*    <=contiene le funzioni
 public String myfctcall(...parametri dei file macro...)
{
 return ( mypackage.mymethod(...parametri...));
                      <=chiamata di livello superiore alle funzioni
 }

public String lowlevelcall(...parametri...)
{
 string result;
  .......codice che utilizza molte delle funzioni del pacchetto...
  return(result)
}
```

Definizione cliette dell'ambiente di linguaggio Java

Modificare il file di esempio, `makeClas.bat` o creare un nuovo file .bat per generare una classe cliette Net.Data, chiamata `dtw_samp.class`, per tutte le funzioni Java. L'esempio che segue mostra come il file di batch, `CreateServer`, elabora tre funzioni Java:

```
rem Batch file to create dtw_samp for Net.Data
java CreateServer dtw_samp.java UserFunctions.java myfile.java
javac dtw_samp.java
```

Il file di batch elabora i seguenti file, insieme al file di stub fornito da Net.Data chiamato `Stub.java` per creare `dtw_samp.class`.

- `dtw_samp.java`
- `UserFunctions.java`
- `myfile.java`

Scrivere un'applicazione JDBC o un'applet è molto simile alla scrittura di un'applicazione C che utilizza DB2 CLI o ODBC per accedere a un database. La principale differenza tra le applicazioni e le applet è che un'applicazione può richiedere software particolare per comunicare con DB2, ad esempio, DB2 Client Application

Enabler. L'applet dipende da un browser Web abilitato per Java e non richiede alcun codice DB2 installato sul client.

Il sistema richiede una configurazione prima di utilizzare JDBC. Queste considerazioni vengono trattate sul sito Web relativo al supporto applet e applicazione di DB2 JDBC:

`http://www.software.ibm.com/data/db2/jdbc/db2java.html`

Configurazione di Net.Data per l'ambiente di linguaggio Java

Per utilizzare l'ambiente di linguaggio Java è necessario configurare Net.Data. Utilizzare i seguenti passi per completare questa configurazione:

1. Creare un file batch per avviare l'applicazione Java poichè Net.Data non riesce ad avviare direttamente un'applicazione Java. Net.Data utilizza questo file per avviare Java Virtual Machine, che esegue la funzione Java. Il file di batch deve contenere l'istruzione java-classpath per assicurare che possano essere trovati i pacchetti Java obbligatori (i pacchetti standard e specifici per l'applicazione). Ad esempio, il file di batch, `launchjv.bat`, contiene il seguente java-classpath:

   ```
   java -classpath %CLASSPATH%;C:\DB2WWW\Javaclas dtw_samp %1 %2 %3 %4 %5 %6
   ```

2. Definire un cliette per gestire il linguaggio di ambiente Java nel file di configurazione di Live Connection, `dtwcm.cnf`. Specificare i numeri della porta univoca per il cliette e il nome file di batch correlato con la variabile di configurazione EXEC_NAME. Nell'esempio che segue, il nome del cliette Java viene definito come DTW_JAVAPPS e la variabile di configurazione EXEC_NAME viene impostata sul nome del file di batch, `launchjv.bat`:

   ```
   CLIETTE DTW_JAVAPPS{
   MIN_PROCESS=1                <= Obbligatorio: questo valore deve essere 1
                                 poichè il cliette JAVAPPS è per più sottoprocessi.
   MAX_PROCESS=1                <= Obbligatorio: questo valore deve essere 1
                                 poichè il cliette JAVAPPS è per più sottoprocessi.
   START_PRIVATE_PORT=5100      <= Deve essere un numero porta univoco
   START_PUBLIC_PORT=5300       <= Deve essere un numero porta univoco
   EXEC_NAME=launchjv.bat       <= Nome del file di batch che include
                                 l'istruzione classpath
   }
   ```

   Quando viene avviato Connection Manager di Net.Data viene anche avviato il cliette di Java specificato nel file di configurazione. Il cliette diventa disponibile per elaborare le richieste di linguaggio Java dalle applicazioni macro di Net.Data.

3. Aggiornare l'istruzione di percorso DTW_JAVAPPS ENVIRONMENT nel file inizializzazione di Net.Data `db2www.ini`, aggiungendo ogni nome cliette all'istruzione. Ad esempio:

   ```
   ENVIRONMENT DTW_JAVAPPS  ( OUT RETURN_CODE ) CLIETTE "DTW_JAVAPPS"
   ```

Creazione ed esecuzione del file della macro

Una volta creata la funzione Java, definita la classe cliette e configurato Net.Data, è possibile eseguire il file della macro contenente i riferimenti per la funzione Java.

1. Creare un file della macro che richiami le funzioni Java. Ad esempio, la chiamata di funzione *myfctcall* richiama la funzione di esempio fornita con Net.Data, utilizzando il cliette DTW_JAVAPPS.

```
%function (DTW_JAVAPPS) myfctcall( ....parametri
dal file della macro ....)

%{ per richiamare l'esempio fornito con Net.Data %}
%function (DTW_JAVAPPS) reverse_linel(str);

%HTML_REPORT{
Viene visualizzata la stringa "Hello World" al contrario.
@reverse_line("Hello World")
Viene riportato il risultato della chiamata di funzione.
@myfctcall( ... ....)
%}
```

2. Avviare Connection Manager. Consultare il capitolo sulle prestazioni in *Guida alla gestione e alla programmazione di Net.Data* per ulteriori informazioni su Connection Manager.
3. Avviare la macro e richiamare Net.Data utilizzando un collegamento HTML, un modulo HTML o un'istruzione URL. Richiamare ad esempio il file della macro di Net.Data macro, `mymacro.d2w`, con la seguente istruzione URL:

   ```
   http://myserver/cgi-bin/dt2www/mymacro.d2w/report
   ```

Ambiente di linguaggio ODBC

L'ambiente di linguaggio ODBC (Open Database Connectivity) esegue le istruzioni SQL tramite un'interfaccia ODBC. ODBC è basato sulla specifica X/Open SQL CAE, che consente ad una singola applicazione di accedere a più sistemi di gestione di database.

Per utilizzare l'ambiente di linguaggio ODBC, è necessario disporre di un driver ODBC e di un driver manager. La documentazione relativa al driver ODBC descrive come installare e configurare l'ambiente ODBC.

L'invio di istruzioni SQL in un ambiente ODBC è simile ad altre funzioni di Net.Data. L'esempio che segue è una macro di Net.Data che invia più istruzioni SQL al database che è l'origine dei dati ODBC. I sistemi operativi che utilizzano la variabile DATABASE devono specificare lo stesso database come origine dei dati nel file ODBC.INI.

```
%DEFINE {
   DATABASE="qesq1"
   SHOWSQL="YES"
   table="int_null"
   LOGIN="netdata1"
   PASSWORD="ibmdb2"
%}

%function(dtw_odbc) sq1() {
create table int_null (int1 int, int2 int)
%}

%function(dtw_odbc) sql2() {
insert into $(table) (int1) values (111)
%}

%function(DTW_odbc) sql3() {
insert into $(table) (int2) values (222)
%}

%function(dtw_odbc) sql4() {
select * from $(table)
%}

%function(dtw_odbc) sql5() {
drop table $(table)
%}

%HTML(REPORT) {
@sql1()
@sql2()
@sql3()
@sql4()
%}
```

Ambiente di linguaggio Oracle

L'ambiente di linguaggio Oracle fornisce l'accesso nativo ai dati Oracle. E' possibile accedere alle tabelle Oracle da un sistema Net.Data eseguito in modalità CGI, FastCGI, NSAPI, ISAPI o GWAPI. Questo ambiente di linguaggio supporta Oracle 7.2, 7.3 e 8.0.

Limitazioni:

- Le procedure memorizzate non sono supportate tramite il presente ambiente di linguaggio.
- La variabile DATABASE non viene utilizzata per accedere ai database Oracle.
- La variabile LOGIN deve contenere il nome istanza database di Oracle. Ad esempio *ora73* è il nome istanza definito nella seguente variabile LOGIN:

  ```
  LOGON=admin@ora73
  ```

Per accedere ad Oracle da Net.Data

1. Verificare che l'istruzione ENVIRONMENT nel file di inizializzazione di Net.Data sia corretta per l'ambiente di linguaggio Oracle. Consultare il capitolo sulla

configurazione in *Guida alla gestione alla programmazione di Net.Data* per procedure ed esempi.

2. Assicurarsi che i componenti appropriati di Oracle siano installati e operativi nel modo seguente:

   a. Installare SQL*Net sulla macchina in cui è installato Net.Data, se non è già installato. Per ulteriori informazioni, aprire la seguente URL:

      `http://www.oracle.com/products/networking/html/stnd_sqlnet.html`

   b. Verificare che la funzione Oracle *tnsping* può essere utilizzata con la stessa autorizzazione di sicurezza utilizzata dal server Web. Per questa verifica, collegarsi con il proprio ID utente di server Web ed immettere:

      `tnsping`
      `nome-istanza-oracle`

      In cui *nome-istanza-oracle* è il nome del sistema Oracle a cui accede la macro di Net.Data.

      E' possibile che non si riesca a verificare la funzione *tnsping* su Windows NT se il server Web viene eseguito con l'autorizzazione del sistema. In tal caso, ignorare questo passo.

   c. Verificare che sia possibile accedere alle tabelle Oracle con la stessa autorizzazione di sicurezza che il server Web utilizza. Per eseguire questa verifica, immettere un'istruzione SQL SELECT, utilizzando lo strumento della riga comandi SQL*Plus, per accedere a una tabella Oracle con un'istruzione SQL SELECT con l'autorizzazione del server Web. Ad esempio:

      `SELECT * FROM` *`nometabella`*

      E' possibile che non si riesca a verificare l'accesso alla tabella su Windows NT se il server Web viene eseguito con l'autorizzazione del sistema. In tal caso, ignorare questo passo.

   Risoluzione del problema: non proseguire con l'elaborazione se queste procedure hanno esito negativo. Se qualcuno di questi passi non riesce, verificare la configurazione Oracle.

3. Assicurarsi che le variabili di ambiente Oracle siano impostate correttamente nel processo del server Web.

   - Per ambiente AIX, inserire le righe che seguono nel file `/etc/environment`:

     `ORACLE_SID=`*`nome-istanza-oracle`*
     `ORACLE_HOME=`*`directory-libreria-runtime-oracle`*

   - Per Windows NT, utilizzare nel pannello di controllo le proprietà di sistema per aggiungere le seguenti variabili di ambiente:

     `ORACLE_SID=`*`nome-istanza-oracle`*
     `ORACLE_HOME=`*`directory-libreria-runtime-oracle`*

   Suggerimento: E' possibile richiedere altre righe per altre variabili di ambiente Oracle, a seconda delle funzioni di Oracle da utilizzare, come il supporto lingua nazionale e la sincronizzazione a due fasi. Consultare la documentazione di gestione Oracle per ulteriori informazioni su queste variabili di ambiente.

4. Verificare il collegamento ad Oracle da Net.Data. Nel file della macro di Net.Data, specificare i valori appropriati contenuti nelle variabili LOGIN e PASSWORD. *Non* definire la variabile DATABASE quando si accede ai

database di Oracle. Quello che segue è un esempio di istruzione connect in un file della macro:

```
%DEFINE LOGIN=user_ID@remote-nome-istanza-oracle
%DEFINE PASSWORD=password
```

Istanze Oracle locale:

Se si accede solo all'istanza Oracle locale, non specificare il nome istanza oracle come parte dell'ID utente di collegamento, come riportato nell'esempio che segue:

```
%DEFINE LOGIN=user_ID
%DEFINE PASSWORD=password
```

Live Connection:

Se viene utilizzato Live Connection, è possibile specificare LOGIN e PASSWORD nel file di configurazione di Live Connection, sebbene non sia consigliato per motivi di sicurezza. Ad esempio:

```
LOGIN=user_ID
PASSWORD=password
```

Suggerimento: Non specificare la variabile di DATABASE per Oracle.

5. Verificare la configurazione eseguendo uno script shell CGI per assicurarsi che sia possibile accedere all'istanza Oracle dal server Web, come in questo esempio:

```
#! /bin/sh
echo "content-type; text/html
echo
echo "< html>< pre>"
set
echo "</pre>< p>< pre>"
tnsping nome-istanza-oracle
echo
```

In alternativa, è possibile eseguire *tnsping* direttamente da una macro di Net.Data, come nell'esempio che segue:

```
%DEFINE testora = %exec "tnsping nome-istanza-oracle"
%HTML (report){
< P>About to test Oracle access with tnsping.
< hr>
$(testora)
< hr>
< P>The Oracle test is complete.
%}
```

Risoluzione del problema:

Se la verifica ha esito negativo, verificare che tutti i passi precedenti abbiano esito positivo controllando i seguenti elementi:

- Verificare la configurazione Oracle.
- Verificare che la sintassi della variabile di ambiente Oracle sia corretta e che non manchi nessuna variabile.
- Verificare il collegamento Oracle, assicurandosi di avere immesso l'ID utente e la password corretti.

Se la verifica ha ancora esito negativo, rivolgersi al servizio assistenza IBM.

Esempio:

Una volta completati i passi per la verifica dell'accesso, è possibile eseguire delle chiamate all'ambiente di linguaggio Oracle con le funzioni contenute nel file della macro, come di seguito indicato:

```
%FUNCTION(DTW_ORA) STL1() {
insert into $(nometabella) (int1,int2) values (111,NULL)
%}
```

Ambiente di linguaggio Perl

L'ambiente di linguaggio Perl è in grado di interpretare gli script Perl in linea specificati in un blocco FUNCTION sulla macro di Net.Data o consente di elaborare script Perl esterni memorizzati in file separati sul server. Le chiamate agli script Perl esterni vengono identificati in un blocco FUNCTION da un'istruzione EXEC, come ad esempio:

```
%EXEC{ nome-script-perl [optional parameters] %}
```

L'ambiente di linguaggio Perl non è in grado di trasferire o richiamare direttamente varabili di Net.Data e quindi queste vengono rese disponibili agli script Perl con queste modalità:

- Net.Data trasferisce i parametri di immissione allo script Perl come variabili di ambiente. Lo script Perl può richiamare i parametri leggendo la serie Perl di associazione.
- Lo script Perl trasferisce i parametri di emissione di nuovo all'ambiente di linguaggio scrivendo su una linea definita il cui nome viene trasferito da Net.Data nella variabile di ambiente, DTWPIPE. Utilizzare la sintassi dell'istruzione DEFINE per scrivere i dati per la linea definita:

  ```
  name = value
  ```

  Per più elementi di dati, separare ogni elemento con una nuova riga o uno spazio.

  Se una variabile presenta lo stesso nome di un parametro di emissione ed utilizza la sintassi precedente, il nuovo valore sostituisce quello corrente. Se un nome variabile non corrisponde ad un parametro di emissione, Net.Data lo ignora.

L'esempio che segue indica come Net.Data trasferisce le variabili da un file della macro.

```
%FUNCTION(DTW_PERL) today() RETURNS(result) {
  $date =  date ;
  chop $date;
  open(DTW, "> $ENV{DTWPIPE}") `` die "Could not open: $!";
  print DTW "result = \"$date\"\n";
%}
%HTML(INPUT) {
  @today()
%}
```

Se lo script Perl è contenuto in un file esterno chiamato `today.prl`, la stessa funzione può essere scritta nel modo indicato dall'esempio che segue:

```
%FUNCTION(DTW_PERL) today() RETURNS(result) {
  %EXEC { today.prl %}
%}
```

Un programma di ambiente di linguaggio Perl accede ai valori di un parametro di tabella tramite il nome di Net.Data. Le intestazioni della colonna per la tabella T sono T_N_*i* e i valori dei campi sono T_V_*i*_*j*. I numeri delle righe e delle colonne nella tabella T sono rispettivamente T_ROWS e T_COLS.

I blocchi REPORT e MESSAGE sono consentiti come in qualsiasi sezione FUNCTION. Essi vengono elaborati da Net.Data e non dall'ambiente di linguaggio. Un programma Perl può, comunque, produrre testo per il flusso standard di emissione e gestire direttamente il modulo HTML di emissione.

Suggerimento per l'autorizzazione: Assicurarsi che il server Web abbia diritto di accesso a tutti i file eseguibili esterni a cui fa riferimento questo ambiente di linguaggio, compreso la versione corretta dell'interprete Perl. Per ulteriori informazioni, consultare la sezione sulla specifica dei diritti di accesso del server Web ai file di Net.Data nel capitolo relativo alla configurazione contenuto in *Guida alla gestione e alla programmazione di Net.Data*.

Ambiente di linguaggio REXX

L'ambiente di linguaggio REXX è in grado di interpretare i programmi REXX in linea specificati in un blocco FUNCTION sulla macro di Net.Data o consente di elaborare programmi REXX esterni memorizzati in file separati. Le chiamate ai programmi REXX esterni vengono identificati in un blocco FUNCTION da un'istruzione EXEC, come ad esempio:

```
%EXEC{ REXX-nome-file-programma [optional parameters] %}
```

L'ambiente di linguaggio REXX utilizza l'API RexxStart() per indicare che l'interpreter REXX deve eseguire il file specificato e poi trasferisce i parametri che seguono il nome file al programma come se fossero stati immessi in una riga comandi. Sul programma REXX, tutti i parametri vengono ricevuti come ARG[1].

Suggerimento per l'autorizzazione: Assicurarsi che il server Web abbia diritto di accesso a tutti i file eseguibili esterni a cui fanno riferimento gli ambienti di linguaggio. Per ulteriori informazioni, consultare la sezione sulla specifica dei diritti di accesso del server Web ai file di Net.Data nel capitolo relativo alla configurazione contenuto in *Guida alla gestione e alla programmazione di Net.Data*.

Sotituzione di variabile:

La sostituzione della variabile viene eseguita solo sulla sezione delle istruzioni eseguibili del blocco FUNCTION. I parametri, tuttavia, sono accessibili al programma REXX se il programma viene definito internamente in un blocco FUNCTION o esternamente in un file separato. L'ambiente di linguaggio REXX utilizza la funzione RexxVariablePool() dei processori di linguaggio REXX per condividere le variabili di Net.Data con il programma REXX. Ciò consente al programma REXX di gestire direttamente le variabili di Net.Data identificate nell'elenco dei parametri.

Un programma REXX accede ai valori di un parametro di tabella come variabili stem REXX. In un programma REXX, le intestazioni della colonna per la tabella T

sono T_N.*i* e i valori dei campi sono T_V.*i.j*. I numeri delle righe e delle colonne nella tabella T sono rispettivamente T_ROWS e T_COLS.

Miglioramento delle prestazioni per AIX:

Se vi sono diverse chiamate all'ambiente di linguaggio REXX sul sistema AIX, è opportuno impostare RXQUEUE_OWNER_PID su 0. Le macro che eseguono diverse chiamate all'ambiente di linguaggio REXX possono avviare diversi processi, esaurendo le risorse di sistema.

E' possibile impostare la variabile di ambiente in tre modi:

- Nel file di macro utilizzando la funzione DTW_SETENV:

  ```
  @DTW_rSETENV("RXQUEUE_OWNER_PID", "0")
  ```

- Nel file di ambiente di sistema AIX, inserendo la seguente istruzione:

  ```
  /etc/environment: RXQUEUE_OWNER_PID = 0
  ```

  Tale metodo influenza le funzioni REXX su tutta la macchina.

- Nel file di ambiente del server web HTTP; ad esempio, per Domino Go Webserver, inserire la seguente istruzione:

  ```
  InheritEnv RXQUEUE_OWNER_PID = 0
  ```

  Tale metodo influenza le funzioni REXX per il server web.

Ambiente di linguaggio SQL

L'ambiente di linguaggio SQL viene utilizzato per eseguire le istruzioni SQL tramite DB2. I risultati dell'istruzione SQL possono essere restituiti nella tabella predefinita di Net.Data o in una tabella specificata dall'utente.

Net.Data supporta tutte le istruzioni SQL autorizzate dall'utente. E' possibile collegarsi ad un database per ogni sezione HTML quando viene richiamato Net.Data come applicazione CGI ed è necessario specificare il nome database con la variabile DATABASE (tranne che in ambiente OS/390). Se il database DB2 è ubicato sulla stessa macchina del server Web, non è necessaria nessun'altra impostazione. In caso contrario, a seconda del sistema operativo utilizzato, è possibile accedere ai database remoti utilizzando CAE (Client Application Enabler) o DDCS (Database Connection Services) per richiamare tutti i supporti per le transazioni DB2. E' anche possibile utilizzare DataJoiner per accedere ad altri database. L'uso di DataJoiner consente di utilizzare una sincronizzazione a due fasi con i database che la supportano.

Ambiente di linguaggio Sybase

L'ambiente di linguaggio Sybase fornisce l'accesso nativo ai dati Sybase. E' possibile accedere alle tabelle Sybase da un sistema Net.Data eseguito in modalità CGI, FastCGI, NSAPI, ISAPI o GWAPI.

Limitazioni:

- L'ambiente di linguaggio Sybase non supporta oggetti estesi, come immagini o oggetti audio. Le procedure memorizzate sono supportate solo per le procedure che non contengono istruzioni SELECT.

- L'ambiente di linguaggio Sybase richiede Live Connection per utilizzare FastCGI.

Per accedere a Sybase da Net.Data

1. Verificare che l'istruzione ENVIRONMENT nel file di inizializzazione di Net.Data sia corretta per l'ambiente di linguaggio Sybase. Consultare il capitolo sulla configurazione in *Guida alla gestione alla programmazione di Net.Data* per procedure ed esempi.
2. Assicurarsi che i componenti appropriati di Sybase siano installati e operativi nel modo seguente:
   a. Installare Open Client di Sybase sulla macchina in cui è installato Net.Data, se non è già installato. Per ulteriori informazioni, consultare la documentazione relativa a Sybase Open Client.
   b. Verificare che sia possibile utilizzare la funzione *ping* di Sybase con la stessa autorizzazione di sicurezza utilizzata dal server Web. Per questa verifica, collegarsi con il proprio ID utente di server Web ed immettere:

      ```
      ping
      nome-istanza-sybase
      ```

      In cui *nome-istanza-sybase* è il nome del sistema Sybase a cui accede la macro di Net.Data.

      E' possibile che non si riesca a verificare la funzione *ping* su Windows NT se il server Web viene eseguito con l'autorizzazione del sistema. In tal caso, ignorare questo passo.
   c. Verificare che sia possibile accedere alle tabelle Sybase con la stessa autorizzazione di sicurezza che il server Web utilizza. Per eseguire questa verifica, immettere un'istruzione SQL SELECT, utilizzando lo strumento della riga comandi ISQL, per accedere a una tabella Sybase con l'autorizzazione del server Web. Ad esempio:

      ```
      SELECT * FROM nometabella
      ```

      E' possibile che non si riesca a verificare l'accesso alla tabella su Windows NT se il server Web viene eseguito con l'autorizzazione del sistema. In tal caso, ignorare questo passo.

   Risoluzione del problema: non proseguire con l'elaborazione se queste procedure hanno esito negativo. Se qualcuno di questi passi non riesce, verificare la configurazione Sybase.
3. Assicurarsi che le variabili di ambiente Sybase siano impostate correttamente nel processo del server Web.
   - Per ambiente AIX, inserire le righe che seguono nel file `/etc/environment`:

     ```
     DSQUERY=nome-istanza-sybase
     SYBASE=sybase-runtime-library-directory
     ```
   - Per Windows NT, utilizzare nel pannello di controllo le proprietà di sistema per aggiungere le seguenti variabili di ambiente:

     ```
     DSQUERY=nome-istanza-sybase
     SYBASE=sybase-runtime-library-directory
     ```

   Suggerimento: E' possibile richiedere altre righe per altre variabili di ambiente Sybase, a seconda delle funzioni di Sybase da utilizzare, come il supporto lingua nazionale e la sincronizzazione a due fasi. Consultare la documen-

tazione di gestione Sybase per ulteriori informazioni su queste variabili di ambiente.

4. Verificare il collegamento a Sybase da Net.Data. Nel file della macro di Net.Data, specificare i valori appropriati contenuti nelle variabili LOGIN, PASSWORD e DATABASE. Quello che segue è un esempio di istruzione connect in un file della macro:

   ```
   %DEFINE DATABASE=nome-database
   %DEFINE LOGIN=user_ID@remote-nome-istanza-sybase
   %DEFINE PASSWORD=password
   ```

   Live Connection: Se viene utilizzato Live Connection, è possibile specificare LOGIN e PASSWORD nel file di configurazione di Live Connection, sebbene non sia consigliato per motivi di sicurezza. Ad esempio:

   ```
   DATABASE=nome-database
   LOGIN=user_ID
   PASSWORD=password
   ```

5. Verificare la configurazione eseguendo uno script shell CGI per assicurarsi che sia possibile accedere all'istanza Sybase dal server Web, come in questo esempio:

   ```
   #! /bin/sh
   echo "content-type; text/html
   echo
   echo "< html>< pre>"
   set
   echo "</pre>< p>< pre>"
   isql -u user_ID -p password << EOFF
   SELECT * FROM nometabella
   EOFF
   echo
   ```

   Risoluzione del problema:

   Se la verifica ha esito negativo, verificare che tutti i passi precedenti abbiano esito positivo controllando i seguenti elementi:

   - Verificare la configurazione di Sybase.
   - Verificare che la sintassi della variabile di ambiente Sybase sia corretta e che non manchi nessuna variabile.
   - Verificare il collegamento Sybase, assicurandosi di avere immesso l'ID utente e la password corretti.

   Se la verifica ha ancora esito negativo, rivolgersi al servizio assistenza IBM.

Esempio:

Una volta completati i passi per la verifica dell'accesso, è possibile eseguire delle chiamate all'ambiente di linguaggio Sybase con le funzioni contenute nel file della macro, come di seguito indicato:

```
%function(DTW_SYB) STL1() {
insert into $(nometabella) (int1,int2) values (111,NULL)
%}
```

Ambiente di linguaggio System

L'ambiente di linguaggio System è un ambiente definito da Net.Data che supporta le chiamate ai programmi esterni identificati in un'istruzione EXEC in un blocco FUNCTION.

L'ambiente di linguaggio System interpreta l'istruzione EXEC trasferendo il nome programma e i parametri al sistema operativo per l'elaborazione utilizzando utilizzando la chiamata di funzione al sistema() di linguaggio C. Questo metodo non consente al programma esterno di trasferire direttamente le variabili a e da Net.Data, come nel caso dell'ambiente di linguaggio REXX e quindi Net.Data elabora le variabili utilizzando il seguente metodo:

- Net.Data trasferisce i parametri di immissione al programma esterno come variabili di ambiente e il programma esterno le richiama:
  - Uno script UNIX CSHELL fa riferimento a variabili di ambiente precedendo il nome della variabile di ambiente con un segno del dollaro ($), ad esempio `$x`.
  - Uno script di linguaggio Perl fa riferimento ad essi riferendosi all'ENV della serie di associazione, ad esempio `%ENV{'x'}`.
  - Un file di batch del DOS fa riferimento al nome variabile racchiuso tra segni percentuale (%), ad esempio `%x%`.
- La maggior parte dei sistemi operativi trasferisce i parametri di emissione di nuovo all'ambiente di linguaggio scrivendo su una linea definita il cui nome viene trasferito da Net.Data nella variabile di ambiente, DTWPIPE. Net.Data per OS/400 trasferisce i parametri di nuovo all'ambiente di linguaggio utilizzando variabili di ambiente. Utilizzare la sintassi dell'istruzione DEFINE per scrivere i dati per la linea definita:

  ```
  name = value
  ```

  Per più elementi di dati, separare ogni elemento con una nuova riga o uno spazio.

  Se una variabile presenta lo stesso nome di un parametro di emissione, il nuovo valore sostituisce quello corrente. Net.Data ignora i nomi variabile che non corrispondono ad alcuno dei parametri di emissione.

Un programma di ambiente di linguaggio system accede ai valori di un parametro di tabella tramite il nome di Net.Data. Le intestazioni della colonna per la tabella T sono T_N_*i* e i valori dei campi sono T_V_*i*_*j*. I numeri delle righe e delle colonne nella tabella T sono rispettivamente T_ROWS e T_COLS.

Suggerimento per l'autorizzazione: Assicurarsi che il server Web abbia diritto di accesso a tutti i file eseguibili esterni richiamati dall'ambiente di linguaggio System. Per ulteriori informazioni, consultare la sezione sulla specifica dei diritti di accesso del server Web ai file di Net.Data nel capitolo relativo alla configurazione contenuto in *Guida alla gestione e alla programmazione di Net.Data*.

Ambiente di linguaggio Registro Web

Il registro Web di Net.Data fornisce una memoria costante per i dati correlati alle applicazioni. Un registro Web può essere utilizzato per memorizzare le informazioni sulla configurazione ed altri dati a cui è possibile accedere in modo dinamico durante l'esecuzione tramite le applicazioni basate su Web. E' possibile accedere ai registri Web solo tramite le macro di Net.Data utilizzando Net.Data e il supporto di registro Web integrato e dai programmi CGI scritti per questo scopo. Il registro Web è disponibile su una sottoserie di sistemi operativi. Consultare l'appendice di riferimento al sistema operativo di Net.Data in *Riferimento di Net.Data*

Lo sviluppo della pagina Web standard richiede che le URL siano posizionate direttamente nell'origine HTML per la pagina. Ciò rende difficoltosa la modifica dei collegamenti. Anche la natura statica limita il tipo di collegamenti che è possibile facilmente posizionare su una pagina Web. L'uso di un registro Web per memorizzare i dati correlati all'applicazione, ad esempio le URL, può essere di supporto per la creazione di pagine HTML con collegamenti impostati in modo dinamico.

Le informazioni possono essere memorizzate e mantenute in un registro dagli sviluppatori di applicazioni e da responsabili Web che dispongono dell'accesso di scrittura al registro. Le applicazioni richiamano le informazioni da registri associati al momento dell'esecuzione. Ciò semplifica la progettazione di applicazioni flessibili e consente anche spostamenti di applicazioni e di server. E' possibile utilizzare le macro di Net.Data per creare pagine HTML utilizzando collegamenti impostati in modo dinamico.

Le informazioni vengono memorizzate in un registro Web nella forma di voci di registro. Ogni voce di registro consiste di una coppia di stringhe di carattere: una stringa RegistryVariable ed una stringa corrispondente RegistryData. Tutte le informazioni che possono essere rappresentate da una coppia di stringhe possono essere memorizzate come voce di registro. Net.Data utilizza la stringa di variabile come chiave di ricerca per assegnare e richiamare voci specifiche da un registro.

E' possibile consultare contenuti di esempio di un registro Web in Tabella 3.

Tabella 3. Registro Web di esempio

CompanyName	**WorldConnect**
Server	ftp.einet.net
JohnDoe/foreground	Green
CompanyURL/IBM Corp.	http://www.ibm.com
CompanyURL/Sun Microsystems Corp.	http://www.sun.com
CompanyURL/Digital Equipment Corp.	http://www.dec.com
JaneDoe/Home_page	http://jane.info.net

Di seguito sono riportate alcune situazioni in cui si consiglia di utilizzare un registro Web:

- E' possibile utilizzare un registro Web per memorizzare i nomi alternativi per i server e le URL, facilitando la riubicazione delle applicazioni e dei server.
- Gli sviluppatori di applicazioni possono fornire applicazioni con dati basate su Web, come le URL, predefinite nel registro. L'utente finale può cambiare i dati del registro per modificare le azioni dell'applicazione.

- E' possibile utilizzare un registro Web per eseguire ricerche URL basate sul nome prodotto, la lingua nazionale, il costruttore e così via.

Le voci indicizzate nel registro Web sono voci le cui stringhe RegistryVariable hanno una stringa di indice aggiuntiva a loro accodata, utilizzando la seguente sintassi:

```
RegistryVariable/Index
```

L'utente fornisce il valore della stringa di indice in un parametro separato per una funzione integrata progettata per gestire voci di indice. Più voci di registro indicizzate possono avere lo stesso valore della stringa RegistryVariable, ma possono mantenere la loro unicità con diversi valori di stringa di indice.

Tabella 4. Registro Web indicizzato di esempio

Smith/Company_URL	http://www.ibmlink.ibm.com
Smith/Home_page	http://www.advantis.com

Anche se le due voci indicizzate nell'esempio presentano lo stesso valore di stringa RegistryVariable `Smith`, la stringa di indice è differente per ogni caso. Essi vengono trattati come due voci distinte dalle funzioni del registro Web.

Configurazione degli ambienti di linguaggio di Net.Data

Prima di poter utilizzare gli ambienti di linguaggio con Net.Data, è necessario configurare il file di inizializzazione di Net.Data e si sta utilizzando Live Connection, il file di configurazione di Live Connection.

Quella che segue è una panoramica di queste attività. Consultare il capitolo sulla configurazione in *Guida alla gestione e alla programmazione di Net.Data* per informazioni dettagliate su come configurare gli ambienti di linguaggio di Net.Data. Rivedere inoltre le precedenti sezioni relative all'ambiente di linguaggio per le istruzioni particolareggiate sulla configurazione speciale.

- Verificare o aggiornare l'istruzione ENVIRONMENT nel file di inizializzazione di Net.Data, `db2www.ini`. Queste istruzioni sono descritte nel capitolo “Configurazione di Net.Data” in *Guida alla gestione e alla programmazione di Net.Data*.
- Definire i cliette per il database o gli ambienti di linguaggio delle applicazioni Java nel file di Live Connection, `dtwcm.cnf`. La definizione dei cliette è descritta nel capitolo sulla configurazione in *Guida alla gestione e alla programmazione di Net.Data*.

Ambienti di linguaggio non-IBM

Oltre a fornire ambienti di linguaggio, Net.Data consente di aggiungere ambienti di linguaggio e di database propri. Net.Data accede agli ambienti di linguaggio dell'utente come librerie di collegamento dinamico o condivise, separate dal file eseguibile di Net.Data. Ogni ambiente di linguaggio deve supportare una serie di interfacce definite da Net.Data. I capitolo che seguono indicano come creare un nuovo ambiente di linguaggio e descrivono l'interfaccia di programmazione dell'ambiente e gli ambienti.

“Creazione di un nuovo ambiente di linguaggio” a pagina 37

“Struttura dell'istruzione dell'ambiente di linguaggio” a pagina 46

“Programmi di utilità dell'interfaccia di programmazione dell'ambiente di linguaggio” a pagina 49

Creazione di un nuovo ambiente di linguaggio

Net.Data utilizza gli ambienti di linguaggio come interfacce come interfacce collegabili di linguaggio di programmazione e di database, a cui si ha accesso come file DLL o librerie condivise. Net.Data fornisce una serie di ambienti di linguaggio, ma quando questi non sono compatibili con le caratteristiche delle applicazioni, è possibile creare nuovi ambienti di linguaggio. Prima di decidere di creare un nuovo ambiente di linguaggio, stabilire se gli ambienti di linguaggio forniti dall'IBM con Net.Data sono compatibili con le proprie necessità.

Quando si decide di creare un nuovo ambiente di linguaggio, è necessario completare la seguente procedura:

- Definire le interfacce e le funzioni per l'ambiente di linguaggio. Fornire l'interfaccia dtw_execute() e tutte le interfacce previste devono corrispondere esattamente ai prototipi definiti nell'intestazione di linguaggio C dtwle.h.
- Creare un makefile o JCL per creare la DLL o la libreria condivisa.
- Creare una DLL o una libreria condivisa che integri la serie di routine di interfaccia di ambiente di linguaggio da fornire. Consultare la documentazione sui linguaggi C o C++ per il sistema operativo per creare i makefile, le DLL o le librerie condivise.
- Rendere tutte le interfacce disponibili all'esterno alla DLL o alla libreria condivisa in modo che Net.Data possa richiamarle.
- Stabilire un'istruzione di configurazione ENVIRONMENT e aggiungerla poi al file di inizializzazione di Net.Data. Consultare “Struttura dell'istruzione dell'ambiente di linguaggio” a pagina 46 per dettagli.
- Aggiungere le funzioni al file macro di Net.Data che utilizza il nuovo ambiente di linguaggio.

Questo capitolo descrive come strutturare l'ambiente di linguaggio.

Struttura di una DLL o di una libreria condivisa

Quando viene creato un ambiente di linguaggio, viene aggiornata la maschera fornita in Appendice A, “Maschera dell'ambiente di linguaggio” a pagina 65 per includere le funzioni di interfaccia dell'ambiente e le strutture di comunicazione utilizzate da Net.Data per comunicare con l'ambiente di linguaggio e per trasferire i parametri su e dall'ambiente di linguaggio.

Le sezioni che seguono descrivono i concetti e le strutture delle funzioni. I programmi di utilità forniti nell'interfaccia dell'ambiente di linguaggio vengono descritti in “Programmi di utilità dell'interfaccia di programmazione dell'ambiente di linguaggio” a pagina 49.

Quali interfacce di ambiente di linguaggio fornire

Quando viene creato un ambiente di linguaggio, è necessario stabilire quali interfacce fornire. Le scelte dipendono da quale tipo di ambiente creare. Ad esempio, se l'ambiente di linguaggio deve accedere ai dati di un database, è necessario operare scelte diverse da quello relative ad un linguaggio di creazione di script. La sezione che segue descrive le interfacce dell'ambiente di linguaggio di Net.Data.

dtw_execute() E' necessario fornire l'interfaccia dtw_execute() per trasferire i parametri di immissione dal file macro; è l'unica interfaccia richiesta per tutti gli ambienti di linguaggio. Net.Data trasferisce tutti i parametri di immissione a dtw_execute() tramite la struttura per comunicazioni dell'ambiente di linguaggio, dtw_lei .

dtw_initialize() Fornire l'interfaccia dtw_initialize() per assegnare o inizializzare i dati. Net.Data richiama questa interfaccia una volta sola per ogni richiamo della macro, prima che la prima funzione richiami l'ambiente di linguaggio. Se non esistono richiami di funzione per l'ambiente di linguaggio, Net.Data non richiama l'interfaccia dtw_initialize().

dtw_cleanup()

Fornire l'interfaccia dtw_cleanup() quando viene fornita un'interfaccia dtw_initialize() e si desidera attivare la gestione errori quando la macro termina in modo anomalo. Net.Data richiama questa interfaccia una volta sola per ogni richiamo della macro.

dtw_getNextRow() Fornire l'interfaccia dtw_getNextRow() come parte di un ambiente di linguaggio di database o di ambiente di linguaggio che può elaborare i dati una riga per volta. Questa interfaccia viene richiamata solo se Net.Data è in esecuzione su sistemi operativi OS/400 o OS/390.

Elaborazione di parametri di immissione

Gli ambienti di linguaggio di Net.Data utilizzano l'interfaccia dtw_execute() per ricevere ed elaborare parametri. L'interfaccia dtw_execute() gestisce la struttura dtw_lei, utilizzata da Net.Data per comunicare con l'ambiente di linguaggio. Utilizzare i seguenti consigli per l'elaborazione dei parametri di immissione quando si scrive l'ambiente di linguaggio.

- Specificare tutti i parametri impliciti contenuti nel file di inizializzazione. Net.Data trasferisce i parametri qui specificati su tutte le chiamate di funzione all'ambiente di linguaggio dopo che sono stati trasferiti tutti i parametri specificati dal creatore della macro sul blocco FUNCTION in esecuzione.
- Ricevere i parametri di immissione sull'interfaccia dtw_execute() come parte della struttura dtw_lei. Chi ha scritto la macro determina l'ordine in cui Net.Data trasferisce i parametri quando sono stati specificati nella definizione del blocco FUNCTION della macro di Net.Data.

La routine processInputParms() della maschera del programma, in Appendice A, “Maschera dell'ambiente di linguaggio” a pagina 65 riporta un metodo di elaborazione dei parametri di immissione.

Elaborazione delle richieste dell'utente

Il modo in cui l'ambiente di linguaggio elabora una richiesta dell'utente dipende da come l'ambiente di linguaggio riceve la richiesta. Net.Data fornisce diversi modi di comunicare una richiesta all'ambiente di linguaggio:

- Tramite il nome funzione specificato su un blocco FUNCTION. Su tutte le chiamate di funzione, Net.Data trasferisce il nome funzione all'ambiente di linguaggio nel campo nome_funzione della struttura dtw_lei.
- Tramite l'elenco parametri del blocco FUNCTION. E' possibile specificare che un parametro contenuto nell'elenco dei parametri può indicare una richiesta utente. Su ogni chiamata di funzione, Net.Data trasferisce i parametri per l'ambiente di linguaggio nel campo parm_data_array della struttura dtw_lei.
- Tramite la sezione di istruzioni eseguibili di un blocco FUNCTION. Su tutte le chiamate di funzione, Net.Data trasferisce le istruzioni eseguibili nel blocco FUNCTION all'ambiente di linguaggio nel campo exec_statement della struttura dtw_lei.

Elaborazione di parametri di emissione

Il metodo da utilizzare per elaborare i parametri di emissione dipende interamente dall'ambiente di linguaggio a da come esso elabora la richiesta dell'utente. Tuttavia, una volta che l'ambiente di linguaggio contiene i dati necessari per restituire la macro di Net.Data, è possibile strutturare l'ambiente di linguaggio in modo da modificare i valori dei parametri trasferiti nel campo parm_data_array della struttura dtw_lei. La routine processOutputParms() della maschera del programma, in Appendice A, “Maschera dell'ambiente di linguaggio” a pagina 65 riporta un modo possibile di elaborare i parametri di emissione e gli esempi di come impostare i valori di parametro della tabella e di stringa.

Comunicazione delle condizioni di errore

La riuscita o il malfunzionamento di una chiamata di funzione possono essere comunicati tramite la variabile macro implicita di Net.Data, RETURN_CODE. Questa variabile viene impostata da Net.Data dopo una chiamata all'interfaccia dtw_execute(). Il suo valore viene impostato sul valore di ritorno della chiamata dtw_execute(). Questo valore viene poi utilizzato da Net.Data per elaborare il blocco MESSAGE della macro di Net.Data, se ne è stato specificato uno per questa chiamata di funzione.

Se non viene specificato alcun blocco MESSAGE o non si possiede una voce in un blocco MESSAGE specificato per gestire il codice di ritorno da dtw_execute(), Net.Data visualizza il contenuto del campo default_error_message della struttura dtw_lei. Questo campo può essere impostato dall'ambiente di linguaggio in qualsiasi momento nella routine dtw_execute(). La routine setErrorMessage() della maschera del programma, in Appendice A, “Maschera dell'ambiente di linguaggio” a pagina 65 riporta un esempio di come impostare il campo default_error_message.

Strutture della comunicazione nell'ambiente di linguaggio

Net.Data utilizza due strutture per comunicare con l'ambiente di linguaggio. L'ambiente di linguaggio deve gestire queste strutture ed impostare e trasferire le informazioni all'interno delle strutture.

- dtw_lei
- dtw_parm_data

Net.Data trasferisce una struttura di interfaccia dell'ambiente di linguaggio (ad esempio, dtw_lei) sulla funzione dell'ambiente di linguaggio che essa richiama. La struttura contiene, tra l'altro, una serie di dati di parametri contenente un elenco dei parametri da trasferire alla funzione di ambiente di linguaggio. La funzione dell'ambiente di linguaggio richiamata da Net.Data elabora la richiesta, aggiorna i parametri nella serie di dati del parametro (se applicabile) e li restituisce a Net.Data.

Net.Data scorre poi la serie dei dati del parametro, aggiorna le sue copie dei parametri in modo da riflettere i nuovi valori impostati dalla funzione dell'ambiente di linguaggio e continua l'elaborazione della macro di Net.Data.

Struttura dtw_lei

La funzione di interfaccia per ogni ambiente di linguaggio riceve un indicatore per la struttura dtw_lei. La struttura dtw_lei presenta il seguente formato:

```
typedef struct dtw_lei {         /* Inter. Amb. ling.           */
    char *nome_funzione;              /* Nome blocco Function        */
    int   flags;                      /* Indic. Inter. Amb. ling.    */

    char *istruz_exec;                /* Istr. amb. ling.            */

    dtw_parm_data_t *parm_data_array; /* Serie param.                */
    char *default_error_message;      /* Messaggio predef.           */
    void *le_opaque_data;             /* Dati spec. amb. ling.       */

    void *row;                        /* Elab. una riga per volta    */

    char  reserved[64];               /* Riservato                   */
} dtw_lei_t;
```

Campi della struttura dtw_lei:

nome_funzione Il campo nome_funzione contiene un indicatore di una stringa che riporta il nome del blocco delle funzioni. Esso risulta utile per specificare il nome blocco FUNCTION nei messaggi di errore visualizzati dall'ambiente di linguaggio.

indicatori Il campo degli indicatori è utilizzato da Net.Data per comunicare con l'ambiente di linguaggio. Specificare l'indicatore del campo degli indicatori eseguendo un'operazione OR utilizzando le seguenti costanti:

- Net.Data imposta DTW_STMT_EXEC per indicare la funzione di interfaccia dtw_execute() il cui campo exec_statement contiene il nome file e i parametri di un'istruzione EXEC.
- DTW_END_ABNORMAL viene impostato da Net.Data per indicare alla funzione di interfaccia dtw_cleanup() che si è verificata una condizione anomala o non prevista e che l'ambiente di linguaggio

deve eseguire un'operazione necessaria di aggiornamento (liberare cioè risorse) prima che Net.Data termini.

- DTW_LE_FATAL_ERROR viene impostato da una funzione di interfaccia di ambiente di linguaggio per indicare a Net.Data che si è verificato un errore grave nell'ambiente di linguaggio. Se questo indicatore viene impostato, Net.Data chiude l'elaborazione della macro di Net.Data, richiama tutta la funzione attiva dtw_cleanup() dell'ambiente di linguaggio con gli indicatori impostati su DTW_END_ABNORMAL, stampa il messaggio predefinito ed esce dal programma. L'indicatore viene contrassegnato solo se su una chiamata dell'ambiente di linguaggio viene riportato un valore di ritorno diverso da zero.
- DTW_LE_MSG_KEEP viene impostato da una funzione di interfaccia di ambiente di linguaggio per indicare a Net.Data che non deve essere liberata la memoria indicata tramite default_error_message. Se questa costante non viene impostata, Net.Data tenta di liberare memoria.
- DTW_LE_CONTINUE viene impostata dalla funzione di interfaccia dtw_execute() per indicare a Net.Data di chiamare la funzione di interfaccia dtw_getNextRow(). Net.Data richiama dtw_getNextRow() solo se l'indicatore viene impostato e il valore di ritorno della chiamata all'interfaccia dtw_execute() è zero.

exec_statement Il campo exec_statement contiene uno dei seguenti indicatori:

- Per una stringa contenente le istruzioni eseguibili (dopo la sostituzione della variabile) dal blocco FUNCTION
- Per il nome file e i parametri da un'istruzione EXEC

parm_data_array Il campo parm_data_array contiene un indicatore per una serie di strutture dtw_parm_data. La serie termina con una struttura parm_data contenente zeri. La struttura dtw_parm_data viene utilizzata da Net.Data per trasferire le variabili e il loro valore ad un ambiente di linguaggio e per richiamare tutte le modifiche del valore della variabile apportate dall'ambiente di linguaggio. Consultare “Struttura dtw_parm_data” a pagina 42 per una descrizione della struttura.

default_error_message Il campo default_error_message viene impostato dall'ambiente di linguaggio su una stringa di caratteri che descrive una condizione di errore. Se il valore di ritorno da una chiamata ad una funzione di interfaccia di ambiente di linguaggio è diverso da zero e non corrisponde al valore di un messaggio in un blocco MESSAGE, viene visualizzato il messaggio predefinito. In caso contrario, Net.Data visualizza il messaggio selezionato dal blocco MESSAGE.

le_opaque_data Il campo le_opaque_data viene impostato da una delle funzioni di interfaccia nell'ambiente di linguaggio per trasferire i parametri da una funzione di interfaccia ad un'altra. Net.Data salva l'indicatore e lo trasferisce ad un'altra funzione di interfaccia che Net.Data richiama. Dopo l'elaborazione della macro di Net.Data e prima di restituire il valore all'utente che esegue la chiamata di Net.Data, Net.Data definisce l'indicatore con un valore NULL. Poichè il campo è specifico per sottoprocessi, l'ambiente di linguaggio può memorizzare i dati specifici per il sottoprocesso. Utilizzare questo campo solo se si dispone della funzione

di interfaccia dtw_cleanup(), in modo che la funzione può liberare la memoria associata al campo le_opaque_data.

riga Il campo della riga viene impostato da Net.Data per un oggetto riga prima di chiamare una funzione di interfaccia dtw_getNextRow() dell'ambiente di linguaggio. La funzione dtw_getNextRow() inserisce una riga dei dati di tabella nell'oggetto utilizzando le funzioni di interfaccia del programma di utilità della riga di Net.Data. Net.Data elabora poi la riga e chiama dtw_getNextRow() fino a che non vi sono più righe da elaborare.

Il campo è riservato per uso IBM.

Struttura dtw_parm_data

Net.Data utilizza la struttura dtw_parm_data per trasferire i parametri ad un ambiente di linguaggio. I parametri scaturiscono da tre origini:

- Parametri espliciti specificati sulla definizione del blocco FUNCTION
- Parametri specificati sull'istruzione di configurazione ENVIRONMENT nel file di inizializzazione di Net.Data
- Variabile di ritorno specificata sulla parola chiave RETURNS su una definizione del blocco FUNCTION

Net.Data trasferisce prima i parametri espliciti, seguiti dai parametri specificati nell'istruzione ENVIRONMENT e poi la variabile di ritorno.

La struttura dtw_parm_data presenta il seguente formato:

```
typedef struct dtw_parm_data {        /* Dati parametro */
    int   parm_descriptor;            /* Descriz. param.            */
    char *parm_name;                  /* Nome parametro             */
    char *parm_value;                 /* Valore parametro           */
    void *res1;                       /* Riservato                  */
    void *res2;                       /* Riservato                  */
} dtw_parm_data_t;
```

Campi della struttura dtw_parm_data:

parm_descriptor Il campo parm_descriptor descrive il tipo e l'uso del parametro trasferito sull'ambiente di linguaggio. Net.Data imposta il campo eseguendo un'operazione OR utilizzando le seguenti costanti:

- DTW_IN indica che un parametro è di sola immissione.
- DTW_OUT indica che un parametro è di sola emissione.
- DTW_INOUT indica che un parametro è di immissione e di emissione.
- DTW_STRING indica che il valore del parametro è un indicatore per una stringa.
- DTW_TABLE indica che il valore del parametro è un indicatore per una tabella.

  Net.Data imposta sempre il campo parm_descriptor su DTW_IN, DTW_OUT o DTW_INOUT ed utilizza un OR logico con DTW_STRING e DTW_TABLE.

parm_name Il campo parm_name è un indicatore per una stringa contenente il nome del parametro. Net.Data imposta questo indicatore NULL se il parametro è una stringa letterale.

parm_value Il campo parm_value è un indicatore per un oggetto contenente il valore del parametro. Questo indicatore è impostato su NULL da Net.Data se il parametro è una variabile non ancora definita.

I campi res1 e res2 sono campi riservati.

parm_name e parm_value indicano entrambi un oggetto assegnato dal *gruppo* runtime di Net.Data, l'area di memoria utilizzata da Net.Data per l'assegnazione dinamica della memoria. Se parm_name o parm_value viene sostituito da un'altra stringa, la stringa originale deve essere sostituita con un indicatore di una stringa di caratteri assegnata dal gruppo di Net.Data. Utilizzare i programmi di utilità dtw_malloc() e dtw_free() per liberare la stringa originale.

Funzioni di interfaccia dell'ambiente di linguaggio

Net.Data utilizza quattro funzioni di interfaccia con un ambiente di linguaggio: fornire una o più di queste funzioni. Tre di queste funzioni sono facoltative, ma ogni ambiente di linguaggio deve avere una funzione di interfaccia dtw_execute(). Se una macro di Net.Data fa riferimento ad un ambiente di linguaggio che non possiede una funzione di interfaccia dtw_execute(), Net.Data restituisce un messaggio di errore e chiude l'elaborazione della macro di Net.Data.

Per richiamare un ambiente di linguaggio, fare riferimento ad esso sul blocco FUNCTION della macro di Net.Data. Le funzioni di interfaccia dell'ambiente di linguaggio devono essere richiamate nel seguente ordine:

1. dtw_initialize()
2. dtw_execute()
3. dtw_getNextRow()
4. dtw_cleanup()

La funzione dtw_execute() è la sola funzione di interfaccia che deve essere fornita in un ambiente di linguaggio.

Quando Net.Data rileva una chiamata ad una funzione che utilizza l'ambiente di linguaggio, esegue questa procedura per richiamare l'ambiente di linguaggio:

1. Net.Data chiama dtw_initialize() se è stato definito per questo ambiente di linguaggio. La funzione esegue tutte le attività di inizializzazione richieste dall'ambiente di linguaggio, come il collegamento ai database o l'assegnazione di variabili.
2. Net.Data chiama dtw_execute() per elaborare il blocco FUNCTION del file della macro contenente le istruzioni che l'ambiente di linguaggio deve elaborare.
3. Net.Data chiama dtw_getNextRow() se, alla successiva restituzione, dtw_execute() ha indicato che deve essere chiamato dtw_getNextRow().
4. Quando l'elaborazione della macro di Net.Data è completata, Net.Data chiama dtw_cleanup() per aggiornare l'ambiente (ad esempio, scollegandosi dal database o liberando le variabili) se questa funzione è stata definita per l'ambiente di linguaggio e ritorna poi al server Web.

Le seguenti sezioni descrivono le funzioni di interfaccia:

- “dtw_initialize()”
- “dtw_execute()”
- “dtw_getNextRow()” a pagina 45
- “dtw_cleanup()” a pagina 45

dtw_initialize()

La funzione di interfaccia dtw_initialize() esegue l'inizializzazione speciale che l'ambiente di linguaggio richiede, come il collegamento a un database o l'assegnazione di variabili. Questa funzione di interfaccia viene chiamata una volta ed è facoltativa.

Net.Data chiama la funzione di interfaccia dtw_initialize() una volta sola, la prima volta che Net.Data chiama un blocco FUNCTION facendo riferimento a quell'ambiente di linguaggio. I riferimenti successivi all'ambiente di linguaggio oltrepassano la chiamata alla funzione di interfaccia dtw_initialize().

Questa funzione di interfaccia non influisce sull'elaborazione del blocco dei messaggi. Un codice di ritorno con numero positivo o zero significa che l'elaborazione continua; un numero negativo che l'elaborazione non continuerà. Se il codice di ritorno è un valore diverso da zero ed esiste un messaggio predefinito nel campo default_error_message, viene emesso il messaggio predefinito; se non esiste un messaggio predefinito, Net.Data emette un messaggio di errore.

dtw_execute()

La funzione di interfaccia dtw_execute() elabora i blocchi FUNCTION del file della macro che contengono le istruzioni che l'ambiente di linguaggio deve elaborare. Ad esempio, un blocco FUNCTION che fa riferimento all'ambiente di linguaggio di un database contiene le istruzioni SQL che l'ambiente di linguaggio utilizza per interrogare il database.

La funzione di interfaccia dtw_execute() viene chiamata ogni volta che una macro di Net.Data elabora un blocco FUNCTION che fa riferimento all'ambiente di linguaggio. Quando la funzione di interfaccia dtw_execute() viene completata, il prosieguo dell'operazione dipende da se l'ambiente di linguaggio sta elaborando i dati di tabella una riga per volta. In tal caso, la funzione di interfaccia imposta l'indicatore DTW_LE_CONTINUE nella struttura dtw_lei per indicare a Net.Data di chiamare la funzione di interfaccia dtw_getNextRow(). Consultare “dtw_getNextRow()” a pagina 45 per ulteriori informazioni sulla funzione di interfaccia dtw_getNextRow() e sulle procedure ad essa collegate.

E' possibile ottimizzare le prestazioni consentendo alla funzione di interfaccia dtw_execute() di eseguire tutta l'elaborazione necessaria a produrre l'immissione per l'elaborazione del blocco del prospetto. Ad esempio, la funzione di interfaccia dell'ambiente di linguaggio SQL dtw_execute genera l'intera tabella da elaborare durante la fase del blocco del prospetto.

dtw_getNextRow()

La funzione di interfaccia dtw_getNextRow() richiama l'immissione per una elaborazione una riga per volta delle tabelle di Net.Data. Essa viene richiamata ogni volta che viene impostato l'indicatore DTW_LE_CONTINUE, che indica che è necessario elaborare un'altra riga di dati per la tabella. Utilizzare dtw_getNextRow() per gli ambienti di linguaggio del database.

Limitazione: Questa interfaccia viene richiamata solo se Net.Data è in esecuzione sui sistemi operativi OS/400 o OS/390.

Net.Data chiama dtw_getNextRow() quando si verificano le seguenti condizioni:

- La chiamata per dtw_execute() viene completata con esito positivo (valore di ritorno zero)
- Sulla funzione di interfaccia dtw_execute() è stato impostato l'indicatore DTW_LE_CONTINUE nella struttura dtw_lei.

Quando la funzione dtw_execute() imposta su on l'indicatore DTW_LE_CONTINUE, Net.Data segue questa procedura:

1. Elabora il blocco dei messaggi per il valore di ritorno della funzione di interfaccia dtw_execute().
2. Chiama la funzione di interfaccia dtw_getNextRow() dell'ambiente di linguaggio ed avvia l'elaborazione di una riga per volta.
3. Elabora il blocco del prospetto.
4. Elabora il blocco dei messaggi per il valore di ritorno della funzione di interfaccia dtw_getNextRow().
5. Stabilisce se dtw_getNextRow() ha attivato l'indicatore DTW_LE_CONTINUE:
   - In caso affermativo, l'elaborazione continua con la funzione di interfaccia dtw_getNextRow() al passo 2.
   - In caso negativo, l'elaborazione di una riga per volta termina e Net.Data continua ad elaborare la macro di Net.Data.

Quando viene chiamato dtw_getNextRow(), il campo della riga nella struttura dtw_lei viene impostato per indicare un oggetto riga. Per gestire l'oggetto riga, utilizzare i programmi di utilità di Net.Data, dtw_row_SetCols() e dtw_row_SetV(). Dopo la prima chiamata alla funzione di interfaccia dtw_getNextRow() l'oggetto riga contiene le intestazioni della colonna per la tabella. Le chiamate successive contengono i dati della tabella.

La funzione dtw_getNextRow() continua ad esser chiamata (a meno che l'elaborazione del blocco dei messaggi non indichi il contrario) fino a che rimane impostato l'indicatore DTW_LE_CONTINUE.

dtw_cleanup()

Utilizzare la funzione di interfaccia dtw_cleanup() per aggiornare l'ambiente di linguaggio se viene utilizzato dtw_initialize() per inizializzare l'ambiente di linguaggio. Scollegandosi ad esempio da un database o eliminando variabili. Questa funzione di interfaccia è facoltativa.

Durante la gestione di una richiesta di Net.Data, Net.Data chiama una funzione di interfaccia dtw_cleanup() una volta quando l'elaborazione di Net.Data si interrompe o un errore impedisce a Net.Data di elaborare il file della macro.

Net.Data imposta il campo degli indicatori nella struttura dtw_lei su DTW_END_ABNORMAL se l'elaborazione dell'aggiornamento è anomala. Le seguenti condizioni anomale forniscono esempi di quando utilizzare dtw_cleanup():

- Una funzione di interfaccia dell'ambiente di linguaggio indica che si è verificato un errore grave impostando il bit DTW_LE_FATAL_ERROR nel campo degli indicatori nella struttura dtw_lei.
- Net.Data ha rilevato un errore irrecuperabile.
- L'elaborazione del blocco dei messaggi della macro di Net.Data termina con l'uscita dal programma.

Se una funzione di interfaccia di ambiente di linguaggio imposta il campo le_opaque_data con un parametro da trasferire tra le funzioni di interfaccia, utilizzare dtw_cleanup() per liberare il campo quando l'elaborazione termina.

Questa funzione di interfaccia non influisce sull'elaborazione del blocco dei messaggi. Se il valore di ritorno è diverso da zero, viene emesso un messaggio predefinito; se non esiste alcun messaggio predefinito, il processore della macro emette un messaggio di avvertenza.

Struttura dell'istruzione dell'ambiente di linguaggio

Ogni ambiente di linguaggio contiene un'istruzione ENVIRONMENT nel file di inizializzazione, DB2WWW.INI, contenente informazioni specifiche per quell'ambiente di linguaggio. Quando viene creato un nuovo ambiente di linguaggio, è necessario strutturare un'istruzione di ambiente per il file di inizializzazione e documentare come gli utenti devono aggiungerla al file di inizializzazione.

Le istruzioni ENVIRONMENT specificano le informazioni sull'ambiente di linguaggio che Net.Data richiede per chiamare e caricare la DLL o la libreria condivisa dell'ambiente di linguaggio, come il nome dell'ambiente di linguaggio, la DLL o la libreria condivisa e l'elenco dei parametri da trasferire all'ambiente di linguaggio per ogni chiamata di funzione.

Net.Data legge le informazioni di configurazione quando viene richiamato, ma non carica le DLL o le librerie condivise dell'ambiente di linguaggio fino a che non viene chiamato un blocco FUNCTION che identifica quell'ambiente di linguaggio all'interno del file della macro. La DLL rimane caricata fino a che Net.Data non termina.

Le sezioni che seguono forniscono le informazioni sulla sintassi, sulle descrizioni dei parametri e gli esempi da utilizzare nella propria documentazione.

Sintassi dell'istruzione ENVIRONMENT

L'istruzione ENVIRONMENT presenta il seguente formato:

```
ENVIRONMENT(tipo) nome-libreria
([uso parametro, ...)
```

Ogni istruzione ENVIRONMENT deve essere su una sola riga.

Quelli che seguono sono i parametri da specificare per ogni ambiente di linguaggio:

- *tipo*

  Il nome che associa questo ambiente di linguaggio alla definizione del blocco FUNCTION in una macro di Net.Data. E' anche necessario specificare il tipo dell'ambiente di linguaggio su una definizione di blocco FUNCTION per indicare a Net.Data i processi dell'ambiente di linguaggio che la funzione chiama. Il nome non può iniziare con il prefisso `DTW_`. Questo prefisso è riservato agli ambienti di linguaggio forniti con Net.Data. Consultare "Blocco di funzioni" in *Riferimenti di Net.Data* per ulteriori informazioni sul blocco FUNCTION.

- *nome_libreria*

  Il nome dell'oggetto contenente le interfacce di ambiente di linguaggio chiamate da Net.Data. In Windows NT e OS/2, il nome DLL è specificato all'interno dell'estensione *.dll*. In AIX, il nome dell'oggetto condiviso viene specificato con l'estensione *.o* e in OS/400 il nome programma del servizio viene specificato con l'estensione *.SRVPGM*. OS/390 non ha alcuna estensione per i file DLL. Esaminare il file di inizializzazione fornito con Net.Data per il sistema operativo di cui si dispone per come specificare questo nome. Prendere in considerazione di utilizzare un percorso completo per assicurarsi che Net.Data rilevi la DLL o la libreria condivisa.

- *elenco_parametri*

  L'elenco dei parametri trasferiti all'ambiente di linguaggio su ogni chiamata di funzione, oltre a quei parametri specificati nella definizione del blocco FUNCTION. Essi vengono trasferiti nel campo *parm_data_array* della struttura *dtw_lei* che segue i parametri specificati nella definizione del blocco FUNCTION. E' necessario definire questi parametri come variabili nella macro di Net.Data prima che venga eseguita la chiamata di funzione. Se una funzione modifica il valore di questi parametri, i parametri richiamano il valore modificato una volta che la funzione ha terminato l'elaborazione.

Esempi di istruzioni ENVIRONMENT

Gli esempi che seguono riportano le istruzioni ENVIRONMENT per gli ambienti di linguaggio che Net.Data fornisce. Questi esempi illustrano come specificare i parametri. Le variabili comprese nelle istruzioni ENVIRONMENT sono quelli che consentono ai creatori di macro di Net.Data di definire nuove macro o di ricoprire quelle dell'utente. Consultare le informazioni specifiche sul sistema operativo contenute nelle relative appendici di *Riferimenti di Net.Data*, nel file README di Net.Data o nella directory del programma per ulteriori esempi.

Gli esempi che seguono riportano la sintassi per OS/2, AIX e Windows NT.

```
ENVIRONMENT (DTW_SQL)      DTWSQL    ( IN DATABASE, LOGIN, PASSWORD,
TRANSACTION_SCOPE, SHOWSQL, ALIGN, START_ROW_NUM, DTW_SET_TOTAL_ROWS)
ENVIRONMENT (DTW_SYB)      DTWSYB    ( IN DATABASE, LOGIN, PASSWORD,
TRANSACTION_SCOPE, SHOWSQL, ALIGN, START_ROW_NUM, DTW_SET_TOTAL_ROWS)
ENVIRONMENT (DTW_ORA)      DTWORA    ( IN LOGIN, PASSWORD,
TRANSACTION_SCOPE, SHOWSQL, ALIGN, START_ROW_NUM, DTW_SET_TOTAL_ROWS)
ENVIRONMENT (DTW_ODBC)     DTWODBC   ( IN DATABASE, LOGIN, PASSWORD,
TRANSACTION_SCOPE, SHOWSQL, ALIGN, DTW_SET_TOTAL_ROWS)
ENVIRONMENT (DTW_APPLET)   DTWJAVA   ( )
ENVIRONMENT (DTW_JAVAPPS)   ( OUT RETURN_CODE ) CLIETTE "DTW_JAVAPPS"
ENVIRONMENT (DTW_PERL)     DTWPERL   ( OUT RETURN_CODE )
ENVIRONMENT (DTW_REXX)     DTWREXX   ( OUT RETURN_CODE )
ENVIRONMENT (DTW_SYSTEM)   DTWSYS    ( OUT RETURN_CODE )
ENVIRONMENT (HWS_LE)       DTWHWS    ( OUT RETURN_CODE )
```

L'istruzione ENVIRONMENT può variare a seconda del sistema operativo; ad esempio OS/390 differisce leggermente per l'accesso SQL e ODBC:

```
ENVIRONMENT (DTW_SQL)      DTWSQL    ( IN LOCATION, DB2SSID, DB2PLAN,
TRANSACTION_SCOPE)

ENVIRONMENT (DTW_ODBC)     DTWODBC   ( IN LOCATION, TRANSACTION_SCOPE)
```

Programmi di utilità dell'interfaccia di programmazione dell'ambiente di linguaggio

Net.Data fornisce un'interfaccia di programmazione da utilizzare quando si crea un nuovo ambiente di linguaggio. L'interfaccia dell'ambiente di linguaggio dispone di programmi di utilità che consentono di accedere ai servizi di Net.Data che gestiscono le memoria e le variabili di configurazione e forniscono funzioni di gestione righe e tabelle. Appendice A, “Maschera dell'ambiente di linguaggio” a pagina 65 fornisce una maschera da utilizzare come modello per la creazione dell'ambiente di linguaggio.

La sezione che segue illustra i programmi di utilità dell'interfaccia dell'ambiente di linguaggio di Net.Data.

Programmi di utilità dell'ambiente di linguaggio

Gli ambienti di linguaggio utilizzano i programmi di utilità per accedere ai servizi di Net.Data. Queste funzioni si dividono in quattro categorie:

- “Programmi di utilità per la gestione della memoria”
- “Programmi di utilità per la gestione delle variabili di configurazione” a pagina 50
- “Programmi di utilità per la gestione di tabelle” a pagina 50
- “Programmi di utilità per la gestione delle righe” a pagina 51

Programmi di utilità per la gestione della memoria

Gli ambienti di linguaggio utilizzano i programmi di utilità di gestione memoria per assegnare la memoria disponibile su Net.Data e per liberare memoria utilizzando la libreria run-time di Net.Data.

L'esempio che segue indica quando sono necessari questi programmi di utilità. Net.Data è stato scritto utilizzando il programma di compilazione A, con la corrispondente libreria run-time. Un programmatore scrive un nuovo ambiente di linguaggio, ma utilizza il programma di compilazione B, che contiene una libreria run-time diversa. L'ambiente di linguaggio non riesce a liberare la memoria che Net.Data ha assegnato e Net.Data non può liberare la memoria assegnata dall'ambiente di linguaggio a causa delle potenziali incompatibilità tra le due librerie run-time.

Tabella 5. Programmi di utilità di gestione memoria

Programma di utilità	Descrizione
“dtw_malloc()” a pagina 53	Assegna memoria dal gruppo run-time di Net.Data utilizzando dtw_malloc().
“dtw_free()” a pagina 52	Libera memoria assegnata dal gruppo run-time di Net.Data utilizzando dtw_malloc().
“dtw_strdup()” a pagina 54	Assegna memoria dal gruppo run-time di Net.Data e copia la stringa specificata nella memoria assegnata utilizzando dtw_malloc().

Programmi di utilità per la gestione delle variabili di configurazione

I programmi di utilità per la gestione delle variabili di configurazione consentono agli ambienti di linguaggio di accedere alle informazioni memorizzate nel file di inizializzazione di Net.Data. Tramite queste funzioni, tutti gli ambienti di linguaggio possono condividere il file di inizializzazione di Net.Data ed utilizzare le informazioni in esso contenute per la configurazione degli ambienti di linguaggio.

Tabella 6. Funzioni del programma di utilità di configurazione

Programma di utilità	Descrizione
“dtw_getvar()” a pagina 52	Richiama il valore di una variabile di configurazione dal file di inizializzazione di Net.Data.

Programmi di utilità per la gestione di tabelle

Utilizzare le funzioni delle tabelle per gestire tutte le variabili delle tabelle macro di Net.Data trasferite all'ambiente di linguaggio.

Le righe e le colonne iniziano con il numero uno (1).

Tabella 7. Programma di utilità delle tabelle

Programma di utilità	Descrizione
“dtw_table_New()” a pagina 59	Crea un oggetto tabella.
“dtw_table_Delete()” a pagina 55	Cancella un oggetto tabella.
“dtw_table_SetCols()” a pagina 61	Imposta l'ampiezza di una tabella ed assegna la memoria per le intestazioni della colonna.
“dtw_table_GetV()” a pagina 57	Richiama un valore di tabella.
“dtw_table_SetV()” a pagina 62	Imposta un valore di tabella.
“dtw_table_GetN()” a pagina 57	Richiama un'intestazione di colonna della tabella.
“dtw_table_SetN()” a pagina 61	Imposta un'intestazione di colonna della tabella.
“dtw_table_Rows()” a pagina 60	Richiama il numero corrente di righe in una tabella.
“dtw_table_Cols()” a pagina 55	Richiama il numero corrente di colonne in una tabella.
“dtw_table_MaxRows()” a pagina 59	Richiama il numero massimo consentito di righe in una tabella.
“dtw_table_QueryColnoNj()” a pagina 60	Richiama il numero colonna di una colonna.
“dtw_table_AppendRow()” a pagina 54	Aggiunge una o più righe alla fine di una tabella.
“dtw_table_InsertRow()” a pagina 58	Inserisce una o più righe in una tabella.
“dtw_table_DeleteRow()” a pagina 56	Elimina una o più righe da una tabella.
“dtw_table_InsertCol()” a pagina 58	Inserisce una o più colonne in una tabella.
“dtw_table_DeleteCol()” a pagina 56	Elimina una o più colonne da una tabella.

Programmi di utilità per la gestione delle righe

I programmi di utilità relativi alle righe gestiscono l'oggetto riga trasferito alla funzione di interfaccia dtw_getNextRow() dell'ambiente di linguaggio durante l'elaborazione di una singola riga per volta.

Le righe iniziano con il numero uno (1).

Tabella 8. Programmi di utilità della riga

Programma di utilità	Descrizione
“dtw_row_SetCols()” a pagina 53	Imposta l'ampiezza di una riga.
“dtw_row_SetV()” a pagina 53	Imposta un valore di tabella.

Riferimenti di sintassi per i programmi di utilità

Questa sezione descrive ogni programma di utilità, il formato, l'uso e i parametri e fornisce facili esempi.

dtw_free()

Uso

Libera memoria assegnata dal gruppo run-time di Net.Data utilizzando dtw_malloc(). Il buffer indica la memoria assegnata da liberare.

Formato

```
void dtw_free(void *buffer)
```

Parametri

buffer	Un indicatore della memoria assegnata da liberare.

Esempi

```
char *myBuf;
long  nbytes = 8192;

myBuf = (char *)dtw_malloc(nbytes);

dtw_free((void *)myBuf);
```

dtw_getvar()

Uso

Richiama il valore di una variabile di configurazione specificata tramite *nome_var* dal file di inizializzazione di Net.Data. Net.Data dispone della memoria restituita da dtw_getvar(); non modificarla o liberarla.

Formato

```
char *dtw_getvar(char *nome_var)
```

Parametri

nome_var	Il nome della variabile di configurazione da richiamare.

Esempi

```
char *myBindFile;

myBindFile = dtw_getvar("BIND_FILE");
```

dtw_malloc()

Uso

Restituisce un indicatore della memoria assegnata dal gruppo run-time di Net.Data utilizzando dtw_malloc(). La memoria misura *nbyte*. Se Net.Data non riesce a restituire la memoria richiesta, viene riportato un valore NULL.

Formato

```
void *dtw_malloc(long nbyte)
```

Parametri

nbyte	Numero di byte da assegnare.

Esempi

```
char *myBuf;
long  nbytes = 8192;

myBuf = (char *)dtw_malloc(nbyte);
```

dtw_row_SetCols()

Uso

Assegna l'ampiezza della riga e la memoria per le intestazioni della colonna. E' possibile utilizzare il programma di utilità dtw_row_SetCols() una volta per riga.

Formato

```
int dtw_row_SetCols(void *riga, int col)
```

Parametri

riga	Un indicatore per una riga appena creata che non ha ancora assegnato colonne.
col	Il numero iniziale di colonne da assegnare nella nuova riga.

Esempi

```
void *myRow;

rc = dtw_row_SetCols(myRow, 5);
```

dtw_row_SetV()

Uso

Assegna un valore di tabella. L'utente che ha richiamato il programma di utilità dtw_row_SetV() ottiene la proprietà della memoria indicata tramite *src*. Per cancellare il valore di tabella corrente, impostare il valore su NULL.

Formato

```
int dtw_row_SetV(void *riga, char *src, int col)
```

Parametri

riga	Indica la riga da modificare.
src	Una stringa di carattere contenente il nuovo valore da impostare.
col	Il numero di colonna del valore da impostare.

Esempi

```
void *myTable;
char *myFieldValue = "newValue";

rc = dtw_row_SetV(myRow, myFieldValue, 3);
```

dtw_strdup()

Uso

Assegna memoria dal gruppo run-time di Net.Data e copia la stringa specificata tramite *string* nella memoria assegnata utilizzando dtw_malloc(). Se Net.Data non riesce a restituire la memoria richiesta, viene riportato un valore NULL.

Formato

```
char *dtw_strdup(char *stringa)
```

Parametri

stringa	Un indicatore del valore della stringa da copiare nella memoria assegnata.

Esempi

```
char *myString = "This string will be duplicated.";
char *myDupString;

myDupString = dtw_strdup(myString);
```

dtw_table_AppendRow()

Uso

Aggiunge una o più righe alla fine della tabella. Assegnare i valori della tabella delle nuove righe con il programma di utilità dtw_table_SetV() dopo che le righe sono state accodate alla tabella.

Formato

```
int dtw_table_AppendRow(void *tabella, int righe)
```

Parametri

tabella	Un indicatore della tabella a cui accodare le righe.
righe	Il numero di righe da accodare.

Esempi

```
void *myTable;

rc = dtw_table_AppendRow(myTable, 10);
```

dtw_table_Cols()

Uso

Restituisce il numero corrente di colonne in una tabella.

Formato

```
int dtw_table_Cols(void *tabella)
```

Parametri

tabella	L'indicatore della tabella il cui numero corrente di colonne viene restituito.

Esempi

```
void *myTable;
int currentColumns;

currentColumns = dtw_table_Cols(myTable);
```

dtw_table_Delete()

Uso

Elimina tutte le intestazioni di colonna, i valori della tabella e l'oggetto tabella.

Formato

```
int dtw_table_Delete(void *tabella)
```

Parametri

tabella	Un indicatore della tabella da cancellare.

Esempi

```
void *myTable;

rc = dtw_table_Delete(myTable);
```

---

dtw_table_DeleteCol()

Uso

Elimina una o più colonne iniziando dalla colonna specificata in *start_col*. Per eliminare tutte le righe e le colonne di una tabella, sostituire il programma di utilità dtw_table_Cols() al parametro *cols*.

```
dtw_table_DeleteCol(tabella, 1, dtw_table_Cols());
```

Formato

```
int dtw_table_DeleteCol(void *tabella, int start_col, int col)
```

Parametri

tabella	Un indicatore della tabella da modificare.
start_col	Il numero colonna della prima colonna da cancellare.
righe	Il numero di colonne da cancellare.

Esempi

```
void *myTable;

rc = dtw_table_DeleteCol(myTable, 1, 10);
```

---

dtw_table_DeleteRow()

Uso

Elimina una o più righe iniziando dalla riga specificata in *start_riga*.

Formato

```
int dtw_table_DeleteRow(void *tabella, int start_riga, int righe)
```

Parametri

tabella	Un indicatore della tabella da modificare.
start_riga	Il numero riga della prima riga da cancellare.
righe	Il numero di righe da eliminare.

Esempi

```
void *myTable;

rc = dtw_table_DeleteRow(myTable, 3, 10);
```

dtw_table_GetN()

Uso

Richiama un'intestazione colonna. Net.Data dispone della memoria indicata da *dest*; non modificarla o liberarla.

Formato

```
int dtw_table_GetN(void *tabella, char **dest, int col)
```

Parametri

tabella	L'indicatore della tabella da cui viene richiamata un'intestazione della colonna.
dest	Indica la stringa di carattere che deve contenere l'intestazione della colonna.
col	Il numero di colonna dell'intestazione colonna.

Esempi

```
void *myTable;
char *myColumnHeading;

rc = dtw_table_GetN(myTable, &myColumnHeading, 5);
```

dtw_table_GetV()

Uso

Richiama un valore da una tabella. Net.Data dispone della memoria indicata da *dest*; non modificarla o liberarla.

Formato

```
int dtw_table_GetV(void *tabella, char **dest, int riga, int col)
```

Parametri

tabella	L'indicatore della tabella da cui viene richiamato un valore.
dest	Indica la stringa di carattere che deve contenere il valore.
riga	Il numero riga della prima riga da richiamare.
col	Il numero di colonna del valore da richiamare.

Esempi

```
void *myTable;
char *myTableValue;

rc = dtw_table_GetV(myTable, &myTableValue, 3, 5);
```

dtw_table_InsertCol()

Uso

Inserisce una o più colonne dopo quella specificata.

Formato

```
int dtw_table_InsertCol(void *tabella, int after_col, int col)
```

Parametri

tabella	Un indicatore della tabella da modificare.
after_col	Il numero della colonna dopo di cui inserire le nuove colonne. Per inserire le colonne all'inizio della tabella, specificare 0.
col	Il numero di colonne da inserire.

Esempi

```
void *myTable;

rc = dtw_table_InsertCol(myTable, 3, 10);
```

dtw_table_InsertRow()

Uso

Inserisce una o più righe dopo quella specificata.

Formato

```
int dtw_table_InsertRow(void *tabella, int after_riga, int righe)
```

Parametri

tabella	Un indicatore della tabella da modificare.
after_riga	Il numero della riga dopo di cui inserire le nuove righe. Per inserire le righe all'inizio della tabella, specificare 0.
righe	Il numero di righe da inserire.

Esempi

```
void *myTable;

rc = dtw_table_InsertRow(myTable, 3, 10);
```

dtw_table_MaxRows()

Uso

Restituisce il numero massimo di righe consentite per la tabella di Net.Data definito nel programma di utilità dtw_table_New() nel parametro *row_lim*.

Formato

```
int dtw_table_MaxRows(void *tabella)
```

Parametri

tabella	L'indicatore della tabella da cui viene richiamato il numero massimo di righe.

Esempi

```
void *myTable;
int maximumRows;

maximumRows = dtw_table_MaxRows(myTable);
```

dtw_table_New()

Uso

Crea un oggetto tabella di Net.Data e inizializza tutte le intestazioni della colonna e i valori di campi su NULL. L'utente che richiama deve specificare il numero iniziale di righe e di colonne e il numero massimo di righe. Se il numero massimo di righe e di colonne è 0, è necessario utilizzare la funzione dtw_table_SetCols() per specificare il numero di campi contenuti in una riga prima di richiamare una qualsiasi funzione di tabella.

Formato

```
int dtw_table_New(void **tabella, int righe, int col, int lim_righe)
```

Parametri

tabella	Il nome della nuova tabella.
righe	Il numero iniziale di righe da assegnare nella nuova tabella.
col	Il numero iniziale di colonne da assegnare nella nuova tabella.
lim_righe	Il numero massimo di righe che la tabella può contenere.

Esempi

```
void *myTable;

rc = dtw_table_New(&myTable, 20, 5, 100);
```

dtw_table_QueryColnoNj()

Uso

Restituisce il numero colonna associato all'intestazione colonna.

Formato

```
int dtw_table_QueryColnoNj(void *tabella, char *nome)
```

Parametri

tabella	Un indicatore della tabella da interrogare.
nome	Una stringa di caratteri che specifica l'intestazione colonna per la quale viene restituito il numero colonna. Se l'intestazione della colonna non esiste nella tabella, viene restituito 0.

Esempi

```
void *myTable;
int columnNumber;

columnNumber = dtw_table_QueryColnoNj(myTable, "column 1");
```

dtw_table_Rows()

Uso

Restituisce il numero corrente di righe nella tabella.

Formato

```
int dtw_table_Rows(void *tabella)
```

Parametri

tabella	L'indicatore della tabella da cui viene richiamato il numero di righe.

Esempi

```
void *myTable;
     int currentRows;

     currentRows = dtw_table_Rows(myTable);
```

---

dtw_table_SetCols()

Uso

Imposta il numero di colonne della tabella e assegna la memoria per le intestazioni della colonna. Specificare le intestazioni della colonna quando viene creata la tabella; o specificarle richiamando questo programma di utilità prima di utilizzare tutte le altre funzioni della tabella. E' possibile utilizzare il programma di utilità dtw_table_SetCols() una volta per tabella. Successivamente, utilizzare i programmi di utilità dtw_table_DeleteCol() o dtw_table_InsertCol().

Formato

```
int dtw_table_SetCols(void *tabella, int col)
```

Parametri

tabella	Un indicatore per una nuova tabella a cui non sono state assegnate colonne o righe.
col	Il numero iniziale di colonne da assegnare nella nuova tabella.

Esempi

```
void *myTable;

rc = dtw_table_SetCols(myTable, 5);
```

---

dtw_table_SetN()

Uso

Assegna un nome all'intestazione colonna. L'utente che ha richiamato il programma di utilità dtw_table_SetN() ottiene la proprietà della memoria indicata tramite il parametro *src*. Per eliminare l'intestazione della colonna, impostare il valore dell'impostazione colonna su NULL.

Formato

```
int dtw_table_SetN(void *tabella, char *src, int col)
```

Parametri

tabella	Un indicatore della tabella di cui viene assegnata l'intestazione.
src	Una stringa di caratteri assegnata alla nuova intestazione colonna.
col	Il numero della colonna.

Esempi

```
void *myTable;
char *myColumnHeading = "newColumnHeading";

rc = dtw_table_SetN(myTable, myColumnHeading, 5);
```

dtw_table_SetV()

Uso

Assegna un valore in una tabella. L'utente che ha richiamato il programma di utilità dtw_table_SetV() ottiene la proprietà della memoria indicata tramite il parametro *src*. Per cancellare il valore di tabella, impostare il valore su NULL.

Formato

```
int dtw_table_SetV(void *tabella, char *src, int riga, int col)
```

Parametri

tabella	Un indicatore della tabella di cui viene assegnato il valore.
src	Una stringa di caratteri assegnata al nuovo valore.
riga	Il numero riga della nuova riga.
col	Il numero colonna del nuovo valore.

Esempi

```
void *myTable;
char *myTableValue = "newValue";

rc = dtw_table_SetV(myTable, myTableValue, 3, 5);
```

Appendixes

Appendice A. Maschera dell'ambiente di linguaggio

Utilizzare questa maschera per creare ambienti di linguaggio propri.

```
/******************************************************************************/
/*                                                                            */
/* Nome file                                                                  */
/*                                                                            */
/* Descrizione                                                                */
/*                                                                            */
/* Funzioni                                                                   */
/*                                                                            */
/* Entry point                                                                */
/*                                                                            */
/* Modifica attività                                                          */
/*                                                                            */
/* Indic.  Motivo      Data      Sviluppatore  Descrizione                    */
/* ------  ----------  --------  ------------  ---------------------------- */
/*                                                                            */
/******************************************************************************/

/*----------------------------------------------------------------------------*/
/* Include                                                                    */
/*----------------------------------------------------------------------------*/
#include "dtwle.h"
```

Figura 2 (Parte 1 di 14). Maschera dell'ambiente di linguaggio

```
#ifdef __MVS__
#pragma export(dtw_initialize)
#pragma export(dtw_execute)
#pragma export(dtw_getNextRow)
#pragma export(dtw_cleanup)
#endif

#ifdef _AIX_
//*--------------------------------------------------------------------*/
/* Funzione                                                           */
/*    dtw_getFp                                                       */
/*                                                                    */
/* Finalità                                                           */
/*    Imposta gli indicatori di funzione per le routine dell'ambiente */
/*    di linguaggio fonirta da questo ambiente. Se una routine della  */
/*    struttura non viene fornita, impostare questo campo su          */
/*    NULL.                                                           */
/*                                                                    */
/* Formato                                                            */
/*    int dtw_getFp(dtw_fp_t *func_pointer)                           */
/*                                                                    */
/* Parametri                                                          */
/*    func_pointer     Un indicatore per una struttura che contiene   */
/*                     gli indicatori della funzione per tutte le     */
/*                     funzioni fornite dall'ambente di linguaggio    */
/*                                                                    */
/* Codici di ritorno                                                  */
/*    Esito pos...... 0                                               */
/*    Esito neg...... -1                                              */
/*--------------------------------------------------------------------*/
int dtw_getFp(dtw_fp_t *func_pointer)
{
    func_pointer->dtw_initialize_fp = dtw_initialize;
    func_pointer->dtw_execute_fp = dtw_execute;
    func_pointer->dtw_getNextRow_fp = dtw_getNextRow;
    func_pointer->dtw_cleanup_fp = dtw_cleanup;
    return 0;
}
#endif
```

Figura 2 (Parte 2 di 14). Maschera dell'ambiente di linguaggio

```
/*--------------------------------------------------------------------*/
/*                                                                    */
/* Funzione                                                           */
/*    dtw_initialize                                                  */
/*                                                                    */
/* Finalità                                                           */
/*                                                                    */
/* Formato                                                            */
/*    int dtw_initialize(dtw_lei_t *le_interface)                     */
/*                                                                    */
/* Parametri                                                          */
/*    le_interface     Indicatore per una struttura contenente        */
/*                     i seguenti campi:                              */
/*                                                                    */
/*      function_name                                                 */
/*      flags                                                         */
/*      exec_statement                                                */
/*      parm_data_array                                               */
/*      default_error_message                                         */
/*      le_opaque_data                                                */
/*      row                                                           */
/*                                                                    */
/* Codici di ritorno                                                  */
/*    Esito pos...... 0                                               */
/*    Esito neg...... 0                                               */
/*--------------------------------------------------------------------*/
int dtw_initialize(dtw_lei_t *le_interface)
{
    return rc;
}
```

Figura 2 (Parte 3 di 14). Maschera dell'ambiente di linguaggio

```
/*--------------------------------------------------------------------*/
/*                                                                    */
/* Funzione                                                           */
/*    dtw_execute                                                     */
/*                                                                    */
/* Finalità                                                           */
/*                                                                    */
/* Formato                                                            */
/*    int dtw_execute(dtw_lei_t *le_interface)                        */
/*                                                                    */
/* Parametri                                                          */
/*    le_interface     Indicatore per una struttura contenente        */
/*                     i seguenti campi:                              */
/*                                                                    */
/*      function_name                                                 */
/*      flags                                                         */
/*      exec_statement                                                */
/*      parm_data_array                                               */
/*      default_error_message                                         */
/*      le_opaque_data                                                */
/*      row                                                           */
/*                                                                    */
/* Codici di ritorno                                                  */
/*    Esito pos...... 0                                               */
/*    Esito neg...... 0                                               */
/*--------------------------------------------------------------------*/
int dtw_execute(dtw_lei_t *le_interface)
{
    /*----------------------------------------------------------------*/
    /* Determina se è stata specificata l'istruzione %exec            */
    /*----------------------------------------------------------------*/
    if (le_interface->flags & DTW_STMT_EXEC) {
        /*------------------------------------------------------------*/
        /* Esamina l'istruzione %exec                                 */
        /*------------------------------------------------------------*/
        rc = processExecStmt(le_interface->exec_statement);
    if (rc)
          {
          }
      }
    else {
        /*------------------------------------------------------------*/
        /* Esamina i dati in linea                                    */
        /*------------------------------------------------------------*/
        rc = processInlineData(le_interface->exec_statement);
    if (rc)
          {
          }
      }
```

Figura 2 (Parte 4 di 14). Maschera dell'ambiente di linguaggio

```
    /*-------------------------------------------------------------------*/
    /* Esamina i parametri di immissione                                 */
    /*-------------------------------------------------------------------*/
    rc = processInputParms(le_interface->parm_data_array);
    if (rc)
      {
      }
    /*-------------------------------------------------------------------*/
    /* Esamina la richiesta                                              */
    /*-------------------------------------------------------------------*/
    rc = processRequest();
    if (rc)
      {
      }
    /*-------------------------------------------------------------------*/
    /* Esamina i dati di emissione                                       */
    /*-------------------------------------------------------------------*/
    rc = processOutputParms(le_interface->parm_data_array);
    if (rc)
      {
      }
    /*-------------------------------------------------------------------*/
    /* Esamina il codice di ritorno e il messaggio di errore predef.  */
    /*-------------------------------------------------------------------*/
    if (rc)
      {
        setErrorMessage(rc, &(le_interface->default_error_message));
      }
    /*-------------------------------------------------------------------*/
    /* Aggiornamento e uscita dal programma.                             */
    /*-------------------------------------------------------------------*/
    return rc;
}
```

Figura 2 (Parte 5 di 14). Maschera dell'ambiente di linguaggio

```
/*--------------------------------------------------------------------*/
/*                                                                    */
/* Funzione                                                           */
/*    dtw_getNextRow                                                  */
/*                                                                    */
/* Finalità                                                           */
/*                                                                    */
/* Formato                                                            */
/*    int dtw_getNextRow(dtw_lei_t *le_interface)                     */
/*                                                                    */
/* Parametri                                                          */
/*    le_interface     Indicatore per una struttura contenente        */
/*                     i seguenti campi:                              */
/*                                                                    */
/*      function_name                                                 */
/*      flags                                                         */
/*      exec_statement                                                */
/*      parm_data_array                                               */
/*      default_error_message                                         */
/*      le_opaque_data                                                */
/*      row                                                           */
/*                                                                    */
/* Codici di ritorno                                                  */
/*    Esito pos...... 0                                               */
/*    Esito neg...... 0                                               */
/*--------------------------------------------------------------------*/
int dtw_getNextRow(dtw_lei_t *le_interface)
{
    return rc;
}
```

Figura 2 (Parte 6 di 14). Maschera dell'ambiente di linguaggio

```
/*--------------------------------------------------------------------*/
/*                                                                    */
/* Funzione                                                           */
/*    dtw_cleanup                                                     */
/*                                                                    */
/* Finalità                                                           */
/*                                                                    */
/* Formato                                                            */
/*    int dtw_cleanup(dtw_lei_t *le_interface)                        */
/*                                                                    */
/* Parametri                                                          */
/*    le_interface     Indicatore per una struttura contenente        */
/*                     i seguenti campi:                              */
/*                                                                    */
/*      function_name                                                 */
/*      flags                                                         */
/*      exec_statement                                                */
/*      parm_data_array                                               */
/*      default_error_message                                         */
/*      le_opaque_data                                                */
/*      row                                                           */
/*                                                                    */
/* Codici di ritorno                                                  */
/*    Esito pos...... 0                                               */
/*    Esito neg...... 0                                               */
/*--------------------------------------------------------------------*/
int dtw_cleanup(dtw_lei_t *le_interface)
{
    /*----------------------------------------------------------------*/
    /* Determina se si tratta di una chiusura normale o anomala.      */
    /*----------------------------------------------------------------*/
    if (le_interface->flags & DTW_END_ABNORMAL) {
        /*------------------------------------------------------------*/
        /* Esegue una chiusura anomala.                               */
        /*------------------------------------------------------------*/
      }
    else {
        /*------------------------------------------------------------*/
        /* Esegue una chiusura normale.                               */
        /*------------------------------------------------------------*/
      }

    return rc;
}
```

Figura 2 (Parte 7 di 14). Maschera dell'ambiente di linguaggio

```
/*--------------------------------------------------------------------*/
/*                                                                    */
/* Funzione                                                           */
/*    processInputParms                                               */
/*                                                                    */
/* Finalità                                                           */
/*                                                                    */
/* Formato                                                            */
/*    unsigned long processInputParms(dtw_parm_data_t *parm__data)    */
/*                                                                    */
/* Parametri                                                          */
/*    dtw_parm_data_t *parm_data                                      */
/*                                                                    */
/* Codici di ritorno                                                  */
/*    Esito pos...... 0                                               */
/*    Esito neg....                                                   */
/*                                                                    */
/*--------------------------------------------------------------------*/
unsigned long processInputParms(dtw_parm_data_t *parm_data)
{
    /*----------------------------------------------------------------*/
    /* Scorre tutte le variabili della serie di dati del parametro.   */
    /* La serie termina con una voce NULL, cioè il campo parm_name    */
    /* è impostato su NULL, il campo parm_value è impostato su NULL e */
    /* il campo parm_descriptor è impostato su 0. Tuttavia l'unico    */
    /* controllo valido per la chiusura della serie di dati del       */
    /* è parm_descriptor == 0, poichè il campo parm_name è NULL quando*/
    /* viene trasferita una stringa letterale e il campo parm_value è */
    /* impostato su NULL quando viene trasferita una variabile non    */
    /* dichiarata.                                                    */
    /*----------------------------------------------------------------*/
    for (; parm_data->parm_descriptor != 0; ++parm_data) {
```

Figura 2 (Parte 8 di 14). Maschera dell'ambiente di linguaggio

```
        /*-------------------------------------------------------------*/
        /* Determinare l'uso di ogni parametro di immissione.          */
        /*-------------------------------------------------------------*/
        switch(parm_data->parm_descriptor & DTW_USAGE) {

            case(DTW_IN):
                /*-----------------------------------------------------*/
                /* Determinare il tipo di ogni parametro di immissione*/
                /*-----------------------------------------------------*/
                switch (parm_data->parm_descriptor & DTW_TYPE) {
                case DTW_STRING:
            break;
                case DTW_TABLE:
            break;
                default:
                        /*-----------------------------------------*/
                    /* Errore interno - tipo di dati sconosciuto      */
                        /*-----------------------------------------*/
            break;
                  }
            break;

            case(DTW_OUT):
            break;

            case(DTW_INOUT):
            break;

                default:
                /*-----------------------------------------------------*/
                /* Errore interno - uso sconosciuto                    */
                /*-----------------------------------------------------*/
            break;
           }
      }
    return rc;
}
```

Figura 2 (Parte 9 di 14). Maschera dell'ambiente di linguaggio

```
/*--------------------------------------------------------------------*/
/*                                                                    */
/* Funzione                                                           */
/*    processOutputParms()                                            */
/*                                                                    */
/* Finalità                                                           */
/*                                                                    */
/* Formato                                                            */
/*    unsigned long processOutputParms(dtw_parm_data_t *parm_data)    */
/*                                                                    */
/* Parametri                                                          */
/*    dtw_parm_data_t *parm_data                                      */
/*                                                                    */
/* Codici di ritorno                                                  */
/*    Esito pos...... 0                                               */
/*    Esito neg.......-1                                              */
/*                                                                    */
/*--------------------------------------------------------------------*/
unsigned long processOutputParms(dtw_parm_data_t *parm_data) {
    /*----------------------------------------------------------------*/
    /* Richiedere i dati di emissione nel modo specifico all'ambiente *
    /* di linguaggio. Ciò dipende interamente dall'interfaccia        */
    /* dell'ambiente di linguaggio e dal modo scelto da LE.           */
    /*----------------------------------------------------------------*/
```

Figura 2 (Parte 10 di 14). Maschera dell'ambiente di linguaggio

```
/    /*-------------------------------------------------------------*/
    /* Scorrere tutti i parametri della serie di dati del parametro,  */
    /* ricercando i parametri di emissione.                           */
    /*-------------------------------------------------------------*/
    for (; parm_data->parm_descriptor != 0; ++parm_data) {

        /*---------------------------------------------------------*/
        /* Determinare l'uso di ogni parametro.                    */
        /*---------------------------------------------------------*/
        if (pd_i->parm_descriptor & DTW_OUT) {
            /*-----------------------------------------------------*/
                /* Determinare il tipo di ogni parametro di immissione*/
            /*-----------------------------------------------------*/
            switch (pd_i->parm_descriptor & DTW_TYPE) {
                case DTW_STRING:
                    /*-------------------------------------------------*/
                    /* Conferire un nuovo valore a un parametro di    */
                    /* stringa. Se il valore del parametro non è NULL,*/
                    /* liberare memoria utilizzando il programma di   */
                    /* utilità dell'interfaccia LE se era stata       */
                    /* assegnata da Net.Data.                         */
                    /*-------------------------------------------------*/
                    if (parm_data->parm_value != NULL)
                        dtw_free(parm_data->parm_value);
                    parm_data->parm_value = dtw_strdup(newValue);
                break;
                case DTW_TABLE:
                    /*-------------------------------------------------*/
                    /* Modificare la dimensione di un parametro di    */
                    /* tabella. Utilizzare il programma di utilità    */
                    /* dell'interfaccia LE per modificare l'oggetto   */
                    /* tabella.                                       */
                    /*-------------------------------------------------*/
                    /*-------------------------------------------------*/
                    /* Richiedere prima l'indicatore dell'oggetto     */
                    /* tabella.                                       */
                    /*-------------------------------------------------*/
                    void *myTable = (void *) parm_data->parm_value;
```

Figura 2 (Parte 11 di 14). Maschera dell'ambiente di linguaggio

```
/*-----------------------------------------------*/
/* Successivamente richiede la dimensione corrente*/
/* della tabella.                                */
/*-----------------------------------------------*/
cols = dtw_table_Cols(myTable);
rows = dtw_table_Rows(myTable);
/*-----------------------------------------------*/
/* Impostare la nuova dimensione (si presume che  */
/* i valori della nuova dimensione siano validi). */
/*-----------------------------------------------*/

/*-----------------------------------------------*/
/* Impostare prima le colonne.                   */
/*-----------------------------------------------*/
if (cols > newColValue)
  {
    dtw_table_DeleteCol(myTable,
                        newColValue + 1,
                        cols - newColValue);
  }
else if (cols < new_col_value)
  {
    dtw_table_InsertCol(myTable,
                        cols,
                        newColValue - cols);
  }

/*-----------------------------------------------*/
/* Impostare le righe.                           */
/*-----------------------------------------------*/
if (newColValue > 0) {
    if (rows > newRowValue)
      {
        dtw_table_DeleteRow(myTable,
                            newRowValue + 1,
                            rows - newRowValue);
      }
    else if (rows < new_row_value)
      {
        dtw_table_InsertRow(myTable,
                            rows,
                            newRowValue - rows);
      }
  }
```

Figura 2 (Parte 12 di 14). Maschera dell'ambiente di linguaggio

```
                    /*-----------------------------------------------*/
                    /* Richiedere il valore dell'ultima riga/colonna. */
                    /*-----------------------------------------------*/
                    dtw_table_GetV(myTable,
                                   &myValue;,
                                   newRowValue,
                                   newColValue);

                    /*-----------------------------------------------*/
                    /* Cancella il valore dell'ultima riga/colonna.   */
                    /*-----------------------------------------------*/
                    dtw_table_SetV(myTable,
                                   NULL,
                                   newRowValue,
                                   newColValue);

                    /*-----------------------------------------------*/
                    /* Imposta il valore dell'ultima riga/colonna.    */
                    /*-----------------------------------------------*/
                    dtw_table_SetV(myTable,
                                   dtw_strdup(myNewValue),
                                   newRowValue,
                                   newColValue);

            break;
                default:
                    /*-----------------------------------------------*/
                    /* Errore interno - tipo di dati sconosciuto      */
                    /*-----------------------------------------------*/
            break;
              }
           }
      }

    return 0;
}
```

Figura 2 (Parte 13 di 14). Maschera dell'ambiente di linguaggio

```
/*--------------------------------------------------------------------*/
/*                                                                    */
/* Funzione                                                           */
/*    setErrorMessage()                                               */
/*                                                                    */
/* Finalità                                                           */
/*                                                                    */
/* Formato                                                            */
/*    unsigned long setErrorMessage(int returnCode,                   */
/*                                  char **defaultErrorMessage)       */
/*                                                                    */
/* Parametri                                                          */
/*    int    returnCode                                               */
/*    char **defaultErrorMessage                                      */
/*                                                                    */
/* Codici di ritorno                                                  */
/*    Esito pos...... 0                                               */
/*    Esito neg.......-1                                              */
/*                                                                    */
/*--------------------------------------------------------------------*/
unsigned long setErrorMessage(int returnCode,
                              char **defaultErrorMessage)
{

    /*----------------------------------------------------------------*/
    /* Imposta il messaggio di errore predefinito in base al codice   */
    /* di ritorno.                                                    */
    /*----------------------------------------------------------------*/
    switch(returnCode) {
        case LE_SUCCESS:
            break;
        case LE_RC1:
            *defaultErrorMessage = dtw_strdup(LE_RC1_MESSAGE_TEXT);
            break;
        case LE_RC2:
            *defaultErrorMessage = dtw_strdup(LE_RC2_MESSAGE_TEXT);
            break;
        case LE_RC3:
            *defaultErrorMessage = dtw_strdup(LE_RC3_MESSAGE_TEXT);
            break;
        case LE_RC4:
            *defaultErrorMessage = dtw_strdup(LE_RC4_MESSAGE_TEXT);
            rc = LE_RC1INTERNAL;
            break;
      }
    return 0;
}
```

Figura 2 (Parte 14 di 14). Maschera dell'ambiente di linguaggio

Appendice B. Informazioni particolari

I riferimenti contenuti in questa pubblicazione relativi a prodotti, programmi o servizi IBM non implicano che l'IBM intenda renderli disponibili in tutti i paesi in cui opera. In sostituzione a quelli forniti dall'IBM, possono essere usati prodotti, programmi o servizi funzionalmente equivalenti che non comportino violazione dei diritti di proprietà intellettuale o di altri diritti IBM. E' responsabilità dell'utente valutare e verificare la possibilità di utilizzare altri programmi e/o prodotti, fatta eccezione per quelli espressamente indicati dall'IBM.

L'IBM può avere brevetti o domande di brevetti in corso relativi a quanto trattato nella presente pubblicazione. La fornitura di questa pubblicazione non implica la concessione di alcuna licenza su di essi. Chi desiderasse ricevere informazioni relative a licenze può rivolgersi per iscritto a:

Director of Commercial Relations
IBM Europe
Schoenaicher Str. 220,
D-7030, Boeblingen
Deutschland.

Coloro che detengono la licenza su questo programma e desiderano avere informazioni su di esso allo scopo di consentire: (i) uno scambio di informazioni tra programmi indipendenti e altri (compreso questo) e (ii) l'uso reciproco di tali informazioni, dovrebbero rivolgersi a:

Site Counsel
IBM Corporation
P.O. Box 12195
3039 Cornwallis
Research Triangle Park, NC 27709-2195
USA

Il programma su licenza descritto in questa pubblicazione e tutto il materiale su licenza disponibile relativo ad esso, sono forniti dall'IBM secondo i termini dell'IBM Customer Agreement.

Questa pubblicazione non implica un uso di produzione ed è fornita "as is", senza alcuna garanzia, comprese le garanzie di commerciabilità ed idoneità ad uno scopo particolare.

Marchi

I seguenti termini sono marchi dell'IBM Corporation:

AIX
DataJoiner
DB2
Domino
IBM
IMS
Lotus
MVS
Net.Data
OS/2
OS/390
OS/400

I seguenti termini sono marchi di altre società:

Java e HotJava sono marchi della Sun Microsystems, Inc.

Microsoft, Windows, Windows NT® e il logo Windows sono marchi Microsoft Corporation.

UNIX e un marchio su licenza esclusiva X/Open Company Limited.

Altri nomi di società, prodotti e servizi possono essere marchi di altre società.

Glossario

ambiente di linguaggio. Un modulo che fornisce l'accesso da una macro Net.Data per un'origine di dati esterni come DB2 o per un linguaggio di programmazione come Perl. Con Net.Data vengono forniti alcuni ambienti di linguaggio come REXX, Perl e Oracle. E' anche possibile creare ambienti di linguaggio personali.

API. Application programming interface (interfaccia di programmazione applicazione).

applet. Un programma Java incluso in una pagina HTML. Le applet operano su browser abilitati a Java, come Netscape e vengono caricate quando viene caricata la pagina HTML.

application programming interface (API). Un'interfaccia funzionale fornita dal sistema operativo o da un programma su licenza reperibile separatamente che consente ad un programma applicativo scritto in un linguaggio di livello complesso di utilizzare dati specifici o funzioni del sistema operativo o del programma su licenza. Net.Data supporta le seguenti API proprietarie del server Web per prestazioni avanzate su processi CGI: ICAPI, GWAPI, ISAPIe NSAPI.

BLOB. Binary large object.

cache. Un tipo di memoria contenente dati di accesso recente, strutturata in modo da rendere rapido l'accesso susseguente agli stessi dati. La cache viene spesso utilizzata per contenere una copia locale dei dati utilizzati più di frequente accessibili su una rete.

Cache Manager. Il programma che gestisce una cache per una macchina. Esso consente di gestire più cache.

CGI. Common Gateway Interface.

cliette. Un processo di lunga esecuzione che gestisce le richieste del server Web. Connection Manager pianifica i processi cliette per la gestione di queste richieste.

CLOB. Character large object.

Common Gateway Interface. Una modalità standard per un server Web di trasferire il controllo ad un programma applicativo e di ricevere i blocchi di dati.

Connection Manager. Un file eseguibile, `dtwcm`, in Net.Data necessario per il supporto della Live Connection.

cookie. Un pacchetto di informazioni inviato da un server HTTP ad un browser Web poi rinviato dal browser ogni volta che accede a quel server. I cookie possono contenere tutte le informazioni arbitrarie che il server sceglie e possono essere utilizzati per mantenere uno stato di esecuzione tra transazioni HTTP che altrimenti non ne presenterebbero alcuno. *Free Online Dictionary of Computing*

database. Una raccolta di tabelle o di spazi tabella e di spazi di indice.

DBMS. Database management system.

DBMS (database management system). Un sistema software che controlla la creazione, l'organizzazione e la modifica di un database ed accede ai dati memorizzati al suo interno.

esecuzione cache. I processi di memorizzazione dei risultati utilizzati più frequentemente da una richiesta al server Web in locale per il richiamo rapido, fino all'aggiornamento delle informazioni.

firewall. Un computer dotato di software che protegge una rete interna dall'accesso esterno.

HTML. Hypertext markup language.

HTTP. Hypertext transfer protocol.

hypertext markup language. Un linguaggio di tag utilizzato per scrivere documenti Web.

hypertext transfer protocol. Il protocollo utilizzato per le comunicazioni tra un server e un browser Web.

ICAPI. Internet Connection API.

ICS. Internet Connection Server.

ICSS. Internet Connection Secure Server.

interfaccia di file non codificato. Una serie di funzioni interne di Net.Data che consentono di leggere e scrivere dati da file di testo.

Internet. Una rete internazionale di computer pubblica TCP/IP.

Internet Connection Secure Server. Server Web IBM protetto.

Internet Connection Server. Server Web IBM non protetto.

Intranet. Una rete TCP/IP all'interno di un firewall aziendale.

ISAPI. API Internet Server di Microsoft.

Java. Un linguaggio di programmazione orientato sull'oggetto e indipendente dal sistema operativo essenziale per le applicazioni Internet.

Live Connection. Una configurazione Net.Data che gestisce Connection Manager e l'API del server Web. Live Connection consente di riutilizzare le connessioni al database.

LOB. Large object.

middleware. Software di tramite tra un programma applicativo ed una rete. Gestisce l'interazione tra diverse applicazioni sui vari sistemi operativi. *Free Online Dictionary of Computing*

NSAPI. Netscape API.

null. Un valore particolare che indica l'assenza di informazioni.

percorso. Un percorso di ricerca utilizzato per individuare i file.

Perl. Un linguaggio di programmazione interpretato.

porta. Un numero a 16 bit utilizzato per comunicare tra TCP/IP e un'applicazione o un protocollo di livello superiore.

server Web. Un computer che esegue software http, come Internet Connection.

TCP/IP. Transmission Control Protocol / Internet Protocol.

tipo di dati. Un attributo di colonne e di valori letterali.

Transmission Control Protocol / Internet Protocol. Una serie di protocolli per comunicazioni che supporta le funzioni di collegamento peer-to-peer per reti locali e estese.

uniform resource locator. Un indirizzo che definisce un server HTTP e, facoltativamente una directory e un nome file, ad esempio: `http://www.software.ibm.com/data/net.data/index.html`.

URL. Uniform resource locator.

Indice analitico

E

F

G

I

M

N

O

P

R

S

T

V