Manual de Ambiente de Linguagens do Net.Data.

Manual de Ambiente de Linguagens do Net.Data.

Nota

Certifique-se de ler as informações em Apêndice B, “Avisos” na página 79 antes de utilizar estas informações e o produto que suportam.

Quarta Edição (Junho 1998)

Índice

Prefácio v
Sobre o Net.Data v
Sobre este Livro v
Quem Deve Ler Este Manual vi
Sobre Exemplos Neste Manual vi

Sobre Ambientes de Linguagem do Net.Data vii

Parte 1. Ambientes de Linguagem Fornecidos pelo Net.Data 1

Capítulo 1. Visão Geral dos Ambientes de Linguagem Fornecidos pelo Net.Data 3

Capítulo 2. Utilização dos Ambientes de Linguagem Fornecidos pelo Net.Data 7
Ambiente de Linguagem de Interface de Arquivo Plano 7
Considerações Sobre Segurança 7
As Funções Internas da FFI 8
Ambiente de Linguagem IMS Web 10
Ambiente de Linguagem de Applet Java 12
Criação de Applets Java 12
Geração de Tags Applet 12
Exemplo de Applet Java 16
Utilização da Interface de Applet Java do Net.Data 17
Ambiente de Linguagem de Aplicativos Java 19
Estrutura de Arquivos do Ambiente de Linguagem Java 19
Criação das Funções Java 20
Definição do Cliette do Ambiente de Linguagem Java 20
Configuração do Net.Data para o Ambiente de Linguagem Java 21
Criação e Execução do arquivo de macro 22
Ambiente de Linguagem ODBC 22
Ambiente de Linguagem Oracle 23
Ambiente de Linguagem Perl 26
Ambiente de Linguagem REXX 27
Ambiente de Linguagem SQL 28
Ambiente de Linguagem Sybase 28
Ambiente de Linguagem System 31
Ambiente de Linguagem de Registro Web 32
Configuração de Ambientes de Linguagem do Net.Data 33

Parte 2. Ambientes de Linguagem Não-IBM 35

Capítulo 3. Criação de um Novo Ambiente de Linguagem 37
Projeto de uma DLL ou Biblioteca Compartilhada 37
Quais Interfaces de Ambiente de Linguagem Devo Fornecer? 38
Processamento de Parâmetros de Entrada 38
Processamento de Solicitações de Usuário 39
Processamento de Parâmetros de Saída 39
Comunicação de Condições de Erro 39

Estruturas de Comunicação de Ambientes de Linguagem 40
A Estrutura dtw_lei 40
A Estrutura dtw_parm_data 42
Funções de Interface de Ambientes de Linguagem 43
dtw_initialize() 44
dtw_execute() 44
dtw_getNextRow() 45
dtw_cleanup() 46
Projeto da Instrução do Ambiente de Linguagem 46
Sintaxe da Instrução ENVIRONMENT 47
Exemplos de Instruções ENVIRONMENT 47

Capítulo 4. As Funções Utilitárias da Interface de Programação de Ambientes de Linguagem 49
Funções Utilitárias do Ambiente de Linguagem 49
Funções Utilitárias para Gerenciamento de Memória 49
Funções Utilitárias para Gerenciamento de Variáveis de Configuração . . . 50
Funções Utilitárias para Manipulação de Tabelas 50
Funções Utilitárias para Manipulação de Linhas 51
Manual de Sintaxe de Funções Utilitárias 52
dtw_free() 52
dtw_getvar() 52
dtw_malloc() 53
dtw_row_SetCols() 53
dtw_row_SetV() 53
dtw_strdup() 54
dtw_table_AppendRow() 54
dtw_table_Cols() 55
dtw_table_Delete() 55
dtw_table_DeleteCol() 56
dtw_table_DeleteRow() 56
dtw_table_GetN() 57
dtw_table_GetV() 57
dtw_table_InsertCol() 58
dtw_table_InsertRow() 58
dtw_table_MaxRows() 59
dtw_table_New() 59
dtw_table_QueryColnoNj() 60
dtw_table_Rows() 60
dtw_table_SetCols() 61
dtw_table_SetN() 61
dtw_table_SetV() 62

Parte 3. Apêndices 63

Apêndice A. Gabarito de Ambiente de Linguagem 65

Apêndice B. Avisos 79
Marcas 80

Glossário 81

Índice Remissivo 83

Prefácio

Obrigado por selecionar o Net.Data Versão 2, as ferramentas de desenvolvimento IBM para criar páginas Web dinâmicas! Com o Net.Data você pode rapidamente desenvolver páginas Web com um conteúdo dinâmico incorporando dados de uma variedade de fontes de dados e usando o poder de linguagens de programação que você já conhece.

O Net.Data Versão 2 fornece desempenho significativamente melhorado juntamente com novos recursos que lhe dão o poder de construir e pôr em prática soluções de negócios de Internet.

Sobre o Net.Data

Com o produto Net.Data da IBM, você pode criar páginas Web dinâmicas usando dados de sistemas de gerenciamento de bancos de dados (DBMSs) tanto relacionais quanto não-relacionais, incluindo bancos de dados criados para DB2, IMS, e ODBC, e usando aplicativos escritos em linguagens de programação como Java, JavaScript, Perl, C, C++, e REXX.

Você pode pensar no Net.Data como um processador macros que executa como middleware em um servidor Web. Você pode escrever programas aplicativos do Net.Data, chamados macros, que o Net.Data interpreta para criar páginas Web dinâmicas com conteúdo personalizado baseadas em entradas do usuário, o estado atual de seus bancos de dados, lógica de negócios existente, e outros fatores que você colocar em sua macro.

Uma solicitação, na forma de URL (uniform resource locator), flui de um navegador, como o Netscape ou Internet Explorer, para um servidor Web que encaminha o pedido ao Net.Data para execução. O Net.Data localiza e executa a macro, e constrói uma página Web que personaliza baseado-se em funções que você escreve. Estas funções podem:

- Encapsular lógica de negócios dentro de scripts Perl, aplicativos C e C++, ou programas REXX.
- Acessar o banco de dados tal como o DB2.

O Net.Data suporta interfaces padrão da indústria como HyperText Transfer Protocol (HTTP) e Common Gateway Interface (CGI). A HTTP é usada entre o navegador e o servidor Web, e a CGI é usada entre o servidor Web e o Net.Data. Isto permite que você selecione seu navegador ou servidor Web favorito para usar com o Net.Data. O Net.Data também suporta FastCGI e as principais APIs de servidor Web em múltiplos sistemas operacionais.

Sobre este Livro

Este livro discute os ambientes de linguagem do Net.Data, que são utilizados quando você chama programas ou funções, ou fontes de dados como bancos de dados DB2, Oracle ou Sybase do seu arquivo de macro do Net.Data. Ele descreve cada ambiente de linguagem fornecido com o Net.Data, assim como descreve a interface de ambiente de linguagem que você pode usar para desenvolver e construir seu próprio ambiente de linguagem.

Este livro pode se referir aos produtos ou recursos que estão anunciados, mas ainda não disponíveis.

Mais informações, inclusive macros de amostra do Net.Data, demonstativo, e a versão mais atual deste livro, estão disponíveis nos seguintes sites de World Wide Web:

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

Quem Deve Ler Este Manual

O usuário que escreve macros do Net.Data podem usar estas informações para aprender sobre as capacidades dos ambientes de linguagem que o Net.Data fornece. Este livro também contém as informações para os usuários que querem escrever seus próprios ambientes de linguagem para o Net.Data.

Para entender os conceitos discutidos neste livro, você deve estar familiarizado com a linguagem de programação C e as informações no *Guia de Programação e Administração do Net.Data* e no *Manual do Net.Data*.

Sobre Exemplos Neste Manual

Os exemplos usados neste manual são mantidos de maneira simples, a fim de ilustrar conceitos específicos e não mostrar todas os casos possíveis. Alguns exemplos são fragmentos que não funcionam sozinhos.

Sobre Ambientes de Linguagem do Net.Data

O Net.Data é feito para permitir que novas interfaces de linguagem de programação e banco de dados sejam incluídas de uma maneira *acessível*. Estas interfaces são chamadas de ambientes de linguagem e são acessadas como DLLs ou bibliotecas compartilhadas. Os Ambientes de linguagem fornecem acesso a aplicativos e bancos de dados que suportam suas páginas Web dinâmicas. Chamando os ambientes de linguagem com chamadas a funções e instruções SQL, você pode acessar as funções e utilitários que estes ambientes de linguagem fornecem para uso com seu aplicativo de negócios. Por exemplo, você pode acessar seu banco de dados ODBC diretamente, usar o ambiente de linguagem Perl para chamar scripts Perl, ou chamar o ambiente de linguagem de Applets Java para executar applets Java.

O arquivo de inicialização do Net.Data associa cada nome de ambiente de linguagem a uma DLL ou biblioteca compartilhada. Cada ambiente de linguagem deve suportar um conjunto padrão de interfaces definidas pelo Net.Data. O Net.Data carrega a DLL ou biblioteca compartilhada especificada no arquivo de inicialização na primeira vez que uma chamada de função para um bloco FUNCTION que especifica o ambiente de linguagem é encontrada.

O Net.Data analisa a macro do Net.Data, mantém as variáveis do Net.Data, comunica-se com os ambientes de linguagem, e formata a saída de acordo com as especificações dos blocos REPORT e MESSAGE. O ambiente de linguagem suporta as interfaces definidas para o Net.Data, torna os parâmetros do Net.Data acessíveis ao processador de linguagem de alguma maneira dependente da linguagem, chama o interpretador de linguagens, e recebe as variáveis de volta do interpretador de linguagens de alguma maneira dependente da linguagem.

Figura 1 na página viii demonstra a interação do Net.Data com ambientes de linguagem.

Trabalhar com ambientes de linguagem em um aplicativo Net.Data envolve dois tipos de tarefas.

- Usar os ambientes de linguagem fornecidos pelo Net.Data para desenvolver um aplicativo do Net.Data.
- Desenvolver novos ambientes de linguagem ou de bancos de dados para outros usuários utilizarem no desenvolvimento de aplicativos Net.Data.

Este livro é organizado para auxiliá-lo com ambas as atividades:

- Parte 1, “Ambientes de Linguagem Fornecidos pelo Net.Data” na página 1 descreve os ambientes de linguagem fornecidos pelo Net.Data que você pode usar com seus aplicativos.
- Parte 2, “Ambientes de Linguagem Não-IBM” na página 35 descreve como criar novos ambientes de linguagem e bancos de dados.

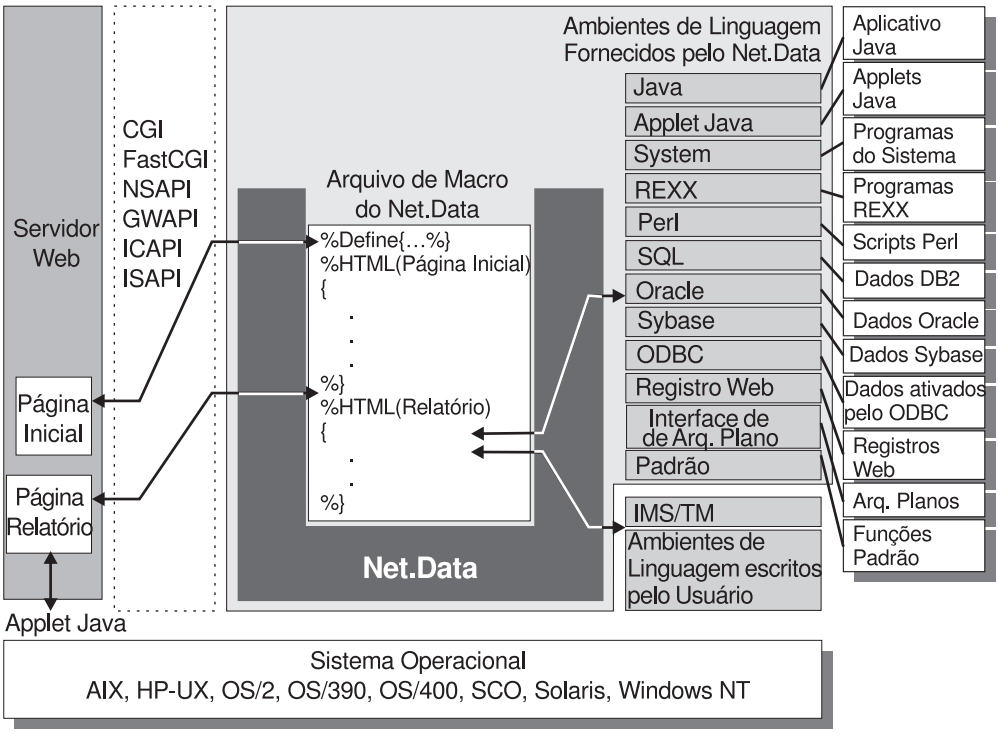

Figura 1. O Net.Data e Ambientes de Linguagem

Parte 1. Ambientes de Linguagem Fornecidos pelo Net.Data

O Net.Data fornece diversos ambientes de linguagem que lhe permitem passar informações para as fontes de dados. Por exemplo, o ambiente de linguagem para SQL permite que você passe consultas SQL nativas para um banco de dados. Semelhantemente, o ambiente de linguagem REXX permite que você chame para execução programas REXX.

Esta seção descreve cada ambiente de linguagem e de como configurar estes ambientes no arquivo de inicialização do Net.Data.

Capítulo 1, “Visão Geral dos Ambientes de Linguagem Fornecidos pelo Net.Data” na página 3

“Ambiente de Linguagem de Interface de Arquivo Plano” na página 7

“Ambiente de Linguagem IMS Web” na página 10

“Ambiente de Linguagem de Applet Java” na página 12

“Ambiente de Linguagem de Aplicativos Java” na página 19

“Ambiente de Linguagem ODBC” na página 22

“Ambiente de Linguagem Oracle” na página 23

“Ambiente de Linguagem Perl” na página 26

“Ambiente de Linguagem REXX” na página 27

“Ambiente de Linguagem SQL” na página 28

“Ambiente de Linguagem Sybase” na página 28

“Ambiente de Linguagem System” na página 31

“Ambiente de Linguagem de Registro Web” na página 32

“Configuração de Ambientes de Linguagem do Net.Data” na página 33

Capítulo 1. Visão Geral dos Ambientes de Linguagem Fornecidos pelo Net.Data

O Net.Data fornece vários ambientes de linguagem, apesar de alguns sistemas operacionais não suportarem todos os ambientes. A Tabela 1 na página 4 relaciona os ambientes de linguagem fornecidos pela IBM. Para determinar se um ambiente de linguagem é suportado em seus sistemas operacionais, consulte o apêndice de referência de sistemas operacionais do *Manual do Net.Data*. Consulte seu arquivo README ou Diretório de Programas do Net.Data para obter detalhes sobre as instruções de ambiente de linguagem que seu sistema operacional usa.

Tabela 1. Ambientes de Linguagem do Net.Data

Ambiente de Linguagem	Instrução de Ambiente	Descrição
Interface de Arquivo Plano	DTW_FILE	A interface de arquivo plano (flat file interface - FFI) fornece funções que suportam arquivos de texto como fontes de dados.
IMS Web	HWS_LE	O ambiente de linguagem de IMS Web permite que você submeta uma transação IMS usando o IMS Web e receba a saída da transação em seu navegador Web.
Applet Java	DTW_APPLET	O ambiente de linguagem applet Java permite que você use applets Java em seus aplicativos do Net.Data. Para gerar um tag de applet, você deve fornecer os qualificadores de tag e a lista de parâmetros de applet.
Aplicativo Java	DTW_JAVAPPS	Net.Data suporta seus aplicativos Java existentes com o ambiente de linguagem Java.
ODBC	DTW_ODBC	O ambiente de linguagem ODBC executa instruções SQL através de uma interface ODBC para acessar múltiplos sistemas de gerenciamento de bancos de dados.
Oracle	DTW_ORA	O ambiente de linguagem Oracle permite a você acessar diretamente seus dados Oracle.
Perl	DTW_PERL	O ambiente de linguagem Perl pode interpretar scripts Perl internos que estão especificados em um bloco FUNCTION da macro Net.Data ou pode executar scripts Perl externos armazenados em arquivos separados.
REXX	DTW_REXX	O ambiente de linguagem REXX pode interpretar programas REXX internos que estão especificados em um bloco FUNCTION da macro Net.Data ou pode executar programas REXX externos armazenados em um arquivo separado.
SQL	DTW_SQL	O ambiente de linguagem SQL executa instruções SQL através do DB2. Os resultados da instrução SQL podem ser retornados em uma variável de tabela.
Sybase	DTW_SYB	O ambiente de linguagem Sybase permite a você acessar diretamente seus dados Sybase.
System	DTW_SYSTEM	O ambiente de linguagem System suporta chamadas a programas externos que estão identificados em uma instrução EXEC no bloco FUNCTION. O ambiente de linguagem de System interpreta as instruções EXEC passando o nome e os parâmetros do programa ao sistema operacional para execução.
Registro Web	DTW_WEBREG	O ambiente de linguagem Registro Web fornece funções para o armazenamento persistente de dados relacionados a aplicativos.

Cada ambiente de linguagem requer uma instrução ENVIRONMENT no arquivo de inicialização e uma biblioteca compartilhada ou arquivo DLL no diretório `/lib` ou `/dll` do servidor. Consulte o capítulo de configuração no *Guia de Programação e Administração do net.Data* para mais informações.

Recomendação: Faça uma cópia de segurança do seu arquivo de inicialização antes de fazer as alterações.

Capítulo 2. Utilização dos Ambientes de Linguagem Fornecidos pelo Net.Data

As seguintes seções descrevem os ambientes de linguagem fornecidos pelo Net.Data, assim como as etapas para configurá-los e utilizá-los.

Ambiente de Linguagem de Interface de Arquivo Plano

Se você optar por usar arquivos planos (ou arquivos de texto corrido) como sua fonte de dados, use a interface de arquivo plano (FFI) e suas funções associadas para abrir, fechar, ler, gravar e eliminar arquivos em seu servidor Web. Você deve especificar um caminho para a variável FFI_PATH no arquivo de inicialização.

O suporte a linguagem de arquivos usa funções FFI para ler e gravar arquivos no servidor Web por solicitação do cliente Web através do navegador. A FFI enxerga o arquivo como um arquivo de registros, cada registro equivalente a uma linha em uma variável de tabela da macro do Net.Data, e cada valor em um registro equivalente a um valor de campo em um variável de tabela e macro do Net.Data. A FFI lê registros de um arquivo para linhas de uma tabela da macro do Net.Data, e grava linhas de uma tabela em registros.

Considerações Sobre Segurança

Você pode especificar quais arquivos funções FFI podem acessar com a instrução FFI_PATH no arquivo de inicialização do Net.Data. A FFI pesquisa apenas os caminhos relacionados na instrução, para que arquivos em outros diretórios fiquem seguros. Isto é uma instrução de exemplo:

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

Os caminhos relacionados em FFI_PATH são pesquisados do primeiro ao último. O Net.Data usa a primeira cópia que ele encontrar. Se FFI_PATH não estiver no arquivo de inicialização, a FFI tenta achar o arquivo no diretório atual. O arquivo de inicialização do Net.Data é enviado sem FFI_PATH.

Recomendações:

- Selecione quais diretórios são apropriados para utilizar as operações de arquivo plano. Estes diretórios precisam ser incluídos no caminho FFI_PATH para limitar pesquisas àqueles diretórios.
- Use cautela quando permitir que outros usuários realizem DTWF_REMOVE ou outras operações de exportação na macro para evitar que usuários removam ou alterem arquivos com extensões `.dll` e `.cmd` que você possa ter no diretório atual.
- Tome cuidados apropriados para proteger os arquivos no sistema usando o controle razoável sobre quais macros são incluídas no sistema.
- Não especifique um caminho em FFI_PATH que permite que usuários FTP anônimos gravem no caminho. Se você o fizer, alguém pode colocar uma macro do Net.Data em seu sistema que permite ações que não eram permitidas antes.
- Não inclua o caminho do arquivo de inicialização do Net.Data no FFI_PATH.

Dica de Autorização: Certifique-se que o servidor Web tem direitos de acesso aos arquivos usados pelas funções internos da FFI. Consulte a seção sobre especificação de direitos de acesso de servidor Web a arquivos do Net.Data no capítulo de configuração do *Guia de Programação e Administração do Net.Data* para mais informações.

As Funções Internas da FFI

Esta seção descreve dicas de utilização e assuntos a considerar quando da utilização de funções internas FFI.

Considerações gerais

- Você pode importar qualquer arquivo de texto corrido, mas a sintaxe macro do Net.Data é interpretada pelo Net.Data e tags HTML que possam estar no texto são usados para formatar o texto pelo navegador.
- Os parâmetros FFI são sensíveis a maiúsculas e minúsculas apenas se o sistema operacional for sensível a maiúsculas e minúsculas.

Diretório atual

- O diretório atual para o Net.Data depende da configuração de seu servidor Web. Se você estiver usando CGI, o diretório atual é o diretório a partir do qual o Net.Data está executando, que normalmente é `\www\cgi-bin`. Se você estiver usando uma API de servidor Web, o diretório atual pode variar. Para gravar no diretório atual, o Net.Data (ou a ID do usuário associado com a subtarefa ou processo que está executando o Net.Data) deve ter permissão de gravação. Se o roteamento de solicitações ou o mapeamento de recursos padrão do servidor forem alterados, o diretório atual pode também ser alterado.
- A maneira recomendada de especificar o diretório atual é usar caminhos absolutos para a instrução FFI_PATH e para o parâmetro FILENAME, especialmente se você estiver usando uma API de servidor Web. Todos os diretórios e subdiretórios relacionados no caminho devem ser definidos na instrução de caminho FFI_PATH no arquivo de inicialização, ou o Net.Data não poderá encontrar o arquivo. Por exemplo, o seguinte caminho requer que o diretório e os subdiretórios `/u/user/mydir/` sejam relacionados no FFI_PATH.

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

  Se você especificar apenas o nome do arquivo, como no seguinte exemplo:

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

  O Net.Data concatena todos os nomes de diretório no caminho para FFI_PATH e pesquisa pelo primeiro arquivo. Se ele não puder encontrar o arquivo então, o Net.Data assume que o arquivo não está no diretório atual. Se o arquivo não estiver no diretório atual, um arquivo com o nome de arquivo especificado será criado no atual diretório. Se o Net.Data não tiver permissão de gravação no diretório atual, ocorre um erro. Não utilize a seguinte sintaxe:

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

Parâmetro DELIMITER

- O delimitador é o sinalizador ou separador que a FFI usa quando divide o arquivo em partes (como colunas de uma linha) de acordo com a transformação solicitada.

- Para as operações de leitura, o delimitador separa o conteúdo do arquivo em linhas e colunas de uma tabela. Para operações de gravação, o delimitador indica o final de um valor em uma linha e coluna de uma tabela.

  O Net.Data passa o delimitador para a FFI como uma cadeia da macro do Net.Data e não inclui um caractere nulo no final dos caracteres a menos que explicitamente relacionado no parâmetro DELIMITER. Para usar o caractere nulo no delimitador, especifique no parâmetro DELIMITER uma barra invertida e um zero entre aspas, “/0”, ao invés de uma cadeia vazia usando aspas, “””. Se você especificar a transformação ASCIITEXT, o Net.Data usa o caractere de nova linha como o delimitador e ignora qualquer delimitador solicitado.
- Alterações indesejáveis a um arquivo podem ocorrer se você utilizar um delimitador diferente para operações de gravação em relação às de leitura. Se você utilizar um delimitador diferente para operações de gravação do que é utilizado nas de leitura, o Net.Data grava o arquivo com o novo delimitador.
- O comprimento máximo de um delimitador é de 256 caracteres.

FFI_PATH

- Os caminhos em FFI_PATH devem conter caracteres que possam ser impressos válidos. A FFI não permite caminhos que incluam uma interrogação (?) ou aspas (“”).
- Os subdiretórios dos caminhos relacionados em FILENAME não são pesquisados a menos que sejam explicitamente especificados em FFI_PATH. Especifique todos os diretórios e subdiretórios no FFI_PATH que você usa com o parâmetro *filename* no arquivo da macro. Os seguintes exemplos mostram instruções de caminho recomendadas:

  Exemplo 1: Especifica um caminho absoluto relacionando todos os diretórios e subdiretórios

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

  O Net.Data pesquisa os caminhos permitidos no FFI_PATH; se o caminho absoluto designado para o parâmetro FILENAME estiver errado ou indisponível, o Net.Data pesquisa no diretório atual e se ele não encontrar o nome do arquivo, emite um aviso.

  Exemplo 2: Especifica um nome de diretório no diretório atual

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

  O Net.Data cria novos arquivos no diretório atual. Se o Net.Data não tiver permissão para criar arquivos no diretório, o Net.Data (ou a ID do usuário associado à subtarefa ou processo que está executando o Net.Data) emite um aviso.

Função DTWF_SEARCH

- A tabela retornada para DTWF_SEARCH tem três colunas. As primeiras duas colunas contêm o número da coluna e da linha em que foi encontrada a correspondência; a última coluna contém o valor da coluna que contém os caracteres especificados no parâmetro *SearchFor*. Por exemplo, se a quarta linha do arquivo contém caracteres correspondentes na coluna três, a tabela retornada tem uma linha com o número `4` na primeira coluna para indicar a linha do arquivo de onde veio; ela tem um número `3` na segunda coluna para indicar que coluna do arquivo contém uma correspondência; e ela tem o valor completo da coluna na terceira coluna.

- O parâmetro *SearchFor* não pode incluir o conteúdo do parâmetro delimitador.

Parâmetros STARTROW e ROWS

- Para as funções DTWF_DELETE, DTWF_INSERT, DTWF_UPDATE, e DTWF_WRITE, se um valor *StartRow* maior que a última linha for especificado, *StartRow* é alterado para indicar a última linha e retorna-se um erro.
- Para as funções DTWF_READ e DTWF_SEARCH, o valor *Rows* é retornado como o número de linhas na tabela.

Parâmetro TABLE

- O máximo comprimento de uma linha em uma tabela FFI é de 16383 caracteres. Este limite inclui um caractere nulo para cada coluna na tabela da macro do Net.Data.

Parâmetro TRANSFORM

- Este parâmetro indica como o arquivo está dividido em partes em relação às linhas e colunas de uma tabela da macro do Net.Data. Por exemplo, transformação ASCIITEXT significa que cada linha do arquivo corresponde a uma linha de uma tabela da macro do Net.Data e a tabela da macro do Net.Data tem apenas uma coluna. Transformação DELIMITED significa que os caracteres na linha são examinados para achar o DELIMITER e após o DELIMITER estar no conteúdo da próxima coluna.
- Um caractere de nova linha em um arquivo indica o final de uma linha de uma tabela da macro do Net.Data para transformações ASCIITEXT e DELIMITED.

Bloqueio de arquivo

- Arquivo não são bloqueados a menos que você os abra com DTWF_OPEN. Se um arquivo não estiver bloqueado, ele pode mudar entre a hora em que é lido e atualizado. Isto pode resultar na perda das alterações anteriores. O uso de DTWF_OPEN abre o arquivo durante a execução da macro usando o mecanismo de bloqueio do sistema de arquivos.

DTWF_APPEND

- O atual conteúdo de um arquivo afeta o resultado do uso de DTWF_APPEND, especialmente o conteúdo da última coluna da última linha. Se uma nova linha seguir o último valor de coluna da última linha do arquivo, os dados anexados são colocados em uma nova linha. Caso contrário, os dados anexados tornam-se parte da última linha do arquivo.

Ambiente de Linguagem IMS Web

O ambiente de linguagem IMS Web é parte de uma solução ponta-a-ponta completa para executar suas transações IMS no ambiente da World Wide Web. O ambiente de linguagem IMS Web fornece:

- Uma macro do Net.Data com:
  - O HTML usado para digitar os dados de entrada da transação
  - Um bloco FUNCTION do Net.Data que chama para execução o ambiente de linguagem IMS Web
  - O HTML que exibe a saída da transação

- Uma DLL ou biblioteca compartilhada da transação que é chamada para execução pelo ambiente de linguagem IMS Web

Restrição: O ambiente de linguagem IMS Web do Net.Data só é suportado quando o Net.Data é executado como um aplicativo CGI. Ele também não é suportado pelo Net.Data com ICAPI.

A ferramenta IMS Web Studio gera código para a DLL e a macro, assim como um arquivo MAK para construir a DLL ou biblioteca compartilhada, a partir da fonte MFS (Message Format Service) para a transação. Após a forma executável da DLL ser construída, a DLL e os arquivos de macro são deslocados para o servidor Web que executa o Net.Data. A transação está pronta para executar no ambiente Web.

Para usar o ambiente de linguagem IMS Web:

1. Instale o componente de Tempo IMS Web no servidor Web que executa o Net.Data. Para mais informações sobre o Tempo de Execução do IMS Web, consulte *Guia do Usuário do IMS Web*:

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

2. Crie o arquivo DLL da transação.
   a. Gere os arquivos macro do C++, MAK e Net.Data para sua transação com a ferramenta IMS Web Studio.
   b. Caso esteva executando o Net.Data em um sistema operacional que é diferente do sistema operacional no qual a ferramenta IMS Web Studio é executada, desloque os arquivos de recurso da DLL para uma máquina de desenvolvimento IMS Web a um sistema operacional de destino. Por exemplo, se você executa a ferramenta IMS Web Studio no Windows NT e a plataforma de destino é AIX ou OS/390, desloque o recurso para DLL de transação a uma máquina de desenvolvimento IMS Web executando respectivamente em baixo do AIX ou OS/390.
   c. Construa a forma executável da DLL da transação utilizando o arquivo MAK gerado.
3. Copie o arquivo da DLL da transação (DTW*proj*.dll) e o arquivo de macro do Net.Data (DTW*proj*.d2w) para o seu servidor Web.
   a. Coloque a macro em um diretório do qual o Net.Data recupera macros. (Consulte a instrução MACRO_PATH no capítulo de configuração do *Guia de Programação e Administração do Net.Data)*
   b. Coloque a DLL da transação ou em um diretório do qual o servidor Web recupera DLLs ou bibliotecas compartilhadas.
4. Use a ligação no arquivo de exemplo que é gerado pela ferramenta IMS Web Studio, DTW*proj*.htm, para modificar um arquivo HTML na árvore HTML do seu servidor Web. Você pode então usar a ligação para chamar a execução do Net.Data e exibir o formulário HTML de entrada para a transação em um navegador Web. Preencha a entrada da transação, e selecione o botão de comando SUBMIT no formulário para executar a transação e receber sua saída no navegador Web.

O IMS Web usa a conexão IMS TCP/IP Open Transaction Manager Access (OTMA) Conexão de comunicação entre os ambientes de servidor Web e IMS. Consulte a home page do IMS Web para mais informações:

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

Ambiente de Linguagem de Applet Java

O ambiente de linguagem de applet Java permite que você gere facilmente tags HTML para applets Java em sua aplicação do Net.Data. Quando você chama o ambiente de linguagem de applet Java, você especifica o nome de seu applet e passa quaisquer parâmetros de que o applet precise. O ambiente de linguagem processa a macro e gera as tags applets do HTML, do qual o navegador Web usa para executar o applet.

Adicionalmente, O Net.Data fornece um conjunto de suas applet de interfaces a serem usadas para acessar parâmetros de tabela. Estas interfaces estão contidas na classe DTW_Applet.class.

As seguintes seções descrevem como usar o ambiente de linguagem de applet Java para executar seus applets Java.

Criação de Applets Java

Antes de usar o ambiente de linguagem de applet Java do Net.Data, you você precisa determinar quais applets planeja usar e escrever. Consulte sua documentação do Java para mais informações na criação de applets.

Geração de Tags Applet

Você especifica o ambiente de linguagem de chamada do applet com a chamada de função do Net.Data. É necessário para a chamada de função. A seguinte sintaxe para a chamada de função é mostrada aqui:

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

- DTWA_ identifica a chamada de função para o ambiente de linguagem de applet.
- AppletName é o nome do applet para o qual são geradas tags.
- parm1 via parmN são parâmetros usados para gerar tags PARAM.

Para gravar o arquivo de macro que geram tags applet:

1. Defina quaisquer parâmetros requeridos pelo applet na seção DEFINE do arquivo de macro. Estes parâmetros incluem atributos de tag applet, variáveis do Net.Data, e parâmetros de tabela do Net.Data que você precisa como entrada para o applet. Por exemplo:

```
%define{
DATABASE = "celdial"                    <=Net.Data variable:
                                             nome do banco de dados
MyGraph.codebase = "/netdata-java"      <=Prâmetro requerido do applet
MyGraph.height = "200"                  <=Parâmetro requerido do applet

MyGraph.width = "400"                   <=Parâmetro requerido do applet
MyTitle = "Este é o Título"       <=Net.Data variable: o nome da página Web
MyTable = %TABLE(all)                   <=Resultados de perguntas
                                             para armazenar Tabela
%}
```

2. Opcional: Especifique uma consulta ao banco de dados para gerar um conjunto de resultados para o applet. Isto é útil quando você está utilizando um applet que gera um quadro ou tabela. Por exemplo:

```
%FUNCTION(DTW_SQL) mySQL(OUT table){
selecione nome, age a partir ibmuser.guests
%}
```

3. Especifique a chamada de função na macro do Net.Data para chamar o ambiente de linguagem de applet Java e chamar para execução o applet. A chamada de função especifica o nome do applet e os parâmetros que você quer passar para o ambiente de linguagem. Estes parâmetros incluem variáveis do Net.Data, e a tabela do Net.Data ou parâmetros de coluna que você precisa como entrada para o applet.

   Por exemplo:

```
%HTML(report){                                          <=Inicie o bloco de HTML
@mySQL(MyTable)                                         <=Chamada de função SQL
                                                              mySQL
@DTWA_MyGraph( MyTitle, DTW_COLUMN(ages) MyTable) <=Chamada de função de Applet
%}
```

4. Chame para execução do Net.Data e execute o arquivo de macro. Consulte o *Guia de Programação e Administração do Net.Data* para ver como chamar o Net.Data para execução.

Atributos de Tag Applet

Você pode especificar atributos de tags applet em qualquer lugar de sua macro do Net.Data. O Net.Data substitui todas as varáveis que têm a forma *AppletName.attribute* dentro da tag de applet como atributo. A sintaxe para definir um atributo na tag de applet é mostrada aqui:

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

Estes atributos são requeridos para todas os applets:

- *codebase:* A localização de applet, o qual é identificada por uma URL.
- *altura:* A altura de applet no pixels.
- *largura:* A largura de applet no pixels.

Por exemplo, se seu applet é chamado MyGraph, você pode definir estes atributos requeridos como mostra aqui:

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

A designação atribuir não precisa estar na seção DEFINE. Você pode definir o valor com a função DTW_ASSIGN. Se você não definir uma variável para *NomeDoApplet.código*, o Net.Data inclui um parâmetro *código* padrão ao tag do applet. O valor do *parâmetro de código*é *AppletName.class*, onde *AppletName* é o nome do seu applet.

Parâmetros de Tags Applet

Você define uma lista de parâmetros para passar para o ambiente de linguagem de applet Java na chamada de função. Você pode passar os parâmetros que incluem:

- Variáveis do Net.Data (inclusive variáveis LIST)
- Tabelas do Net.Data
- Colunas de tabelas do Net.Data

Quando você passa um parâmetro, o Net.Data cria um tag de applet do Java PARAM na saída HTML com o nome e valor que você designar ao parâmetro. Você não pode passar a cadeia literal ou resultados de chamadas de função.

Parâmetros Variáveis do Net.Data: Você pode usar variáveis Net.Data como parâmetros. Se você definir uma variável no bloco DEFINE da macro e passar o valor da variável na chamada de função DTWA_*NomeDoApplet*, o Net.Data gera um tag PARAM que tem o mesmo nome e valor que a variável. Por exemplo, dada a seguinte instrução macro:

```
%define{

...

MyTitle = "Este é o meu Título"
%}

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

O Net.Data produz o seguinte tag PARAM de applet:

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

Parâmetros de Tabela do Net.Data:

O Net.Data automaticamente gera um tag PARAM com o nome DTW_NUMBER_OF_TABLES toda vez que o ambiente de linguagem de applet Java é chamado, especificando se a chamada de função passou alguma variável de tabela. O valor é o número de variáveis de tabela que o Net.Data usa na função. Se nenhuma variável de tabela for especificada na chamada de função, o seguinte tag é gerado:

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

Você pode passar uma ou mais variáveis de tabela do Net.Data como parâmetros na chamada de função. Se você especificar uma variável de tabela do Net.Data em uma chamada de função DTWA_*NomeDoApplet*, o Net.Data gera os seguintes tags PARAM:

Nome da tabela de parâmetro tag :

Esta tag especifica os nomes de tabelas para passar. A tag têm a seguinte sintaxe:

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

Onde `i` é o número da tabela baseada na ordem da chamada função e *tname* é o nome da tabela.

Tags de parâmetros de especificação de colunas:

Tags PARAM são geradas para especificar o número de linhas e colunas da da tabela particular. Esta tag têm a seguinte sintaxe:

```
<param name = 'DTW_tname_NUMBER_OF_ROWS' value = "rows" >
<param name = 'DTW_tname_NUMBER_OF_COLUMNS' value = "cols" >
```

Onde o nome da tabela é *tname*, *rows* é o número de linhas na tabela, e *cols* é o número de colunas da tabela. Este par de tags é gerado para cada tabela exclusiva especificada na chamada de função.

Tags de parâmetros de valor de coluna:

Esta tag PARAM especifica o nome da coluna da coluna particular. Esta tag têm a seguinte sintaxe:

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

Onde o nome da tabela é *tname*, *j* é o número da coluna, e *cname* é o nome da coluna na tabela.

Tags de parâmetros de valor de linha:

Esta tag PARAM especifica os valores para uma linha particular e coluna. Esta tag têm a seguinte sintaxe:

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

Onde o nome da tabela é *tname*, *cname* é o nome da coluna, *k* é o número da linha, e *val* é o valor que corresponde ao valor na linha da coluna correspondente.

Parâmetros de Coluna do Net.Data: Você pode passar uma coluna de tabela como um parâmetro em uma chamada de função para gerar tags para uma coluna específica. O Net.Data gera os tags applet correspondentes apenas para a coluna especificada. Um parâmetro de coluna de tabela usa a seguinte sintaxe:

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

Onde *x* é o nome ou o número da coluna na tabela.

Os parâmetros de coluna de tabela usam os mesmos tags applet definidos para os parâmetros da tabela.

Texto Alternativo para Tag de Applet em que o Navegador que não está habilitado para Java

A variável DTW_APPLET_ALTTEXT especifica o texto que exibe no navegador que não faz o suporte Java ou retornou ao suporte desativado do Java. Por exemplo, a seguinte definição de variável:

```
%define DTW_APPLET_ALTTEXT = "<P>Sorry, seu navegador não está habilitado para Java."
```

produz o seguinte texto e tag HTML:

```
<P>Desculpe, seu navegador não está habilitado para Java.<BR>
```

Caso esta variável não estiver definida, não altere o texto exibido.

Exemplo de Applet Java

O seguinte exemplo demonstra um arquivo de macros do Net.Data que chama o ambiente de linguagem de applet Java e o tag de applet resultante que o ambiente de linguagem gera.

O arquivo de macro do Net.Data contém as seguintes chamadas de função ao ambiente de linguagem de applet:

```
%define{
DATABASE = "celdial"
DTW_APPLET_ALTTEXT = "<P>Desculpe, seu navegador não está habilitado para Java."
DTW_DEFAULT_REPORT = "no"
MyGraph.codebase = "/netdata-java/"
MyGraph.height = "200"
MyGraph.width = "400"
MyTitle = "Este é o meu Título"
%}
%FUNCTION(DTW_SQL) mySQL(OUT table){
selecione nome, age a partir ibmuser.guests
%}
%HTML (report){
@mySQL(MyTable)
@DTWA_MyGraph( MyTitle, DTW_COLUMN(ages) MyTable )
%}
```

As linhas da macro do Net.Data na seção DEFINE especificam a primeira linha de tag applet:

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

O ambiente de linguagem gera um tag de applet com os seguintes qualificadores:

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

O Net.Data retorna os resultados da consulta SQL da seção SQL do arquivo de macro do Net.Data na tabela de saída, `MyTable`. Esta tabela é especificada na seção DEFINE:

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

A chamada ao applet na macro é especificada na seção HTML:

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

Baseado nos parâmetros da chamada de função, o Net.Data gera o tag de applet completo que contém as informações sobre a tabela de resultados, como o número de colunas, o número de linhas retornado, e as linhas de resultado. O Net.Data gera um tag de parâmetro para cada célula na tabela de resultados, conforme mostrado no seguinte exemplo:

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

O nome do parâmetro, *DTW_MyTable_ages_VALUE_1*, especifica a célula da tabela (linha 1, coluna `ages`) na tabela, `MyTable`, que tem um valor de 4. A palavra-chave, DTW_COLUMN, na chamada de função ao applet, especifica que

você está interessado apenas na coluna ages da tabela resultante, MyTable, mostrada aqui:

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

A seguinte saída mostra o tag de applet completo que o Net.Data gera para o exemplo:

```
<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>Desculpe, seu navegador não está habilitado para Java.<BR>
</applet>
```

Utilização da Interface de Applet Java do Net.Data

O Net.Data fornece um conjunto de interfaces na chamada classe DTW_Applet.class, do qual você pode usar com o processo de auxílio do seu applets Java tags PARAM tags que são geradas para as variáveis de tabela. Você pode criar um applet que extende esta interface para a chamada de rotina pelo seu applet.

O Net.Data fornece estas interfaces:

- **int GetNumberOfTables()** retorna o número de tabelas encontrado no tag de applet.
- **String [] GetTableNames()** retorna uma lista de nomes de tabela encontrada no tag de applet.
- **int GetNumberOfColumns(String table_name)** retorna o número de colunas na tabela table_name.
- **int GetNumberOfRows(String table_name)** retorna o número de linhas na tabela table_name.
- **String[] GetColumnNames(String table_name)** retorna os nomes de colunas na tabela table_name.
- **String[][] GetTable(String table_name)** retorna uma matriz bidimensional de cadeias que contém os valores da linhas e colunas da tabela.

Para acessar as interfaces, use a palavra-chave EXTENDS no seu código de applet para tornar seu applet uma subclasse da classe DTW_APPLET, conforme mostrado no seguinte exemplo:

```
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 table_name;

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

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

        System.out.println("Table: " + table_name + " tem " + ncols + " colunas e
                           " + nrows + " linhas.");

        String [] col_names = GetColumnNames(table_name);

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

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

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

        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 de Linguagem de Aplicativos Java

O Net.Data suporta seus aplicativos Java existentes com o ambiente de linguagem Java. Com o suporte ao applet Java e métodos Java (ou aplicativos), você pode acessar o DB2 através da API Java Database Connectivity (JDBC**).

Detalhes sobre a JDBC estão disponíveis nos seguintes sites Web:

- A IBM Software tem o JDK 1.1 ou superior, que é necessário para usar JDBC com o Net.Data:

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

- A JavaSoft tem controladores JDBC adicionais, documentação de API JDBC, e as atualizações mais recentes do JDBC:

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

O ambiente de linguagem Java fornece uma interface como Remote Procedure Call (RPC). Você pode emitir chamadas de função do seu arquivo de macro do Net.Data com cadeias do Net.Data como parâmetros e sua função chamada do Java pode retornar uma cadeia. Você deve usar o Net.Data Live Connection quando você usar o ambiente de linguagem Java (consulte o capítulo sobre desempenho do *Guia de Administração e Programação do Net.Data* para mais informações sobre o Live Connection). Para usar o ambiente de linguagem do Net.Data você deve completar as seguintes etapas. Estas etapas são descritas em detalhes nas seções subseqüentes.

1. Escreva suas funções Java.
2. Crie um cliette do Net.Data para todas suas funções Java (cliettes do Net.Data lançam a Java Virtual Machine onde sua função do Java executa).
3. Defina uma instrução de cliette no arquivo de configuração do Live Connection.
4. Inicie o Connection Manager
5. Execute o arquivo de macro do Net.Data que chama a execução do ambiente de linguagem Java.

Cada vez que você apresentar novas funções Java, você deve recriar o cliette Java.

Estrutura de Arquivos do Ambiente de Linguagem Java

O Net.Data cria diversos diretórios durante sua instalação. Estes diretórios incluem os arquivos que você precisa para criar suas funções Java, definir o cliette, e executar a macro com o ambiente de linguagem Java:

- Uma função Java de exemplo chamada `UserFunctions.java`.
- Um arquivo de exemplo chamado `makeClas`. Quando executado, este arquivo cria uma classe cliette do Net.Data para suas funções Java.
- Um arquivo de exemplo chamado `launchjv` usado pelo cliette do Net.Data para lançar a Java Virtual Machine e executar sua função Java.

Tabela 2 na página 20 descreve os nomes do diretório e do arquivo para os arquivos em seu sistema operacional.

Tabela 2. Os Arquivos Usados para Criar Funções do Java

Sistema Operacional	Nnome do Arquivo	Diretório
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

Criação das Funções Java

Altere o arquivo de exemplo de função Java `UserFunctions.java`, ou crie um novo arquivo baseado no seguinte arquivo de exemplo, chamado `myfile.java`:

```
====================myfile.java====================
import mypackage.*                                    <=contêm sua funções
 public String myfctcall(...parâmetros do arquivo de macro...)
{
 return ( mypackage.mymethod(...parâmetros...));      <=chamada de alto
                                                  nível a suas funções
 }

public String lowlevelcall(...parâmetros...)
{
 string result;
  .......código utilizado muitas funções de seu pacote...
  return(result)
}
```

Definição do Cliette do Ambiente de Linguagem Java

Altere o arquivo de exemplo, `makeClas.bat`, ou crie um novo arquivo .bat para gerar uma classe de cliettes do Net.Data, chamada `dtw_samp.class`, para todas as suas funções. O seguinte exemplo mostra como o arquivo em lote, `CreateServer`, processa três funções Java:

```
rem Arquivo em lote a criar dtw_samp para Net.Data
java CreateServer dtw_samp.java UserFunctions.java myfile.java
javac dtw_samp.java
```

O arquivo em lote processa os seguintes arquivo, juntamente com o arquivo canhoto fornecido com o Net.Data, chamado `Stub.java` para criar `dtw_samp.class`.

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

Escrever um aplicativo ou applet JDBC é bastante semelhante a escrever um aplicativo em C utilizando o DB2 CLI ou ODBC para acessar um banco de dados. A principal diferença entre aplicativos e applet é que um aplicativo pode requerer software especial para se comunicar com o DB2, por exemplo o DB2 Client

Application Enabler. O applet depende de um navegador Web habilitado para Java, e não requer qualquer código DB2 instalado no cliente.

Seu sistema requer alguma configuração antes de usar o JDBC. Estas considerações são discutidas no site Web de Suporte a Aplicativos e Applet do JDBC do DB2:

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

Configuração do Net.Data para o Ambiente de Linguagem Java

Para usar o ambiente de linguagem Java, você deve configurar o Net.Data. Use as seguintes etapas para completar estas etapas de configuração:

1. Crie um arquivo em lote para lançar o aplicativo Java porque o Net.Data não pode iniciar um aplicativo Java diretamente. O Net.Data usa este arquivo para lançar a Java Virtual Machine, que executa sua função Java. O arquivo em lote deve incluir a instrução java-classpath para assegurar que os pacotes Java necessários (os pacotes padrão e específico para aplicativos) possam ser encontrados. Por exemplo, o arquivo em lote, `launchjv.bat`, contém o seguinte java-classpath:

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

2. Defina um cliette para funcionar com o ambiente de linguagem Java no arquivo de configuração do Live Connection, `dtwcm.cnf`. Especifique os números de porta exclusivos para o cliette e o nome do arquivo em lote relacionado através da variável de configuração EXEC_NAME. No seguinte exemplo, o nome do cliette Java é definido como DTW_JAVAPPS e a variável de configuração EXEC_NAME é definida com o nome do arquivo em lote, `launchjv.bat`:

   ```
   CLIETTE DTW_JAVAPPS{
   MIN_PROCESS=1                <= Necessário: este valor deve ser 1 porque
                                 o cliette JAVAPPS tem múltiplas subtarefas
   MAX_PROCESS=1                <= Necessário: este valor deve ser 1 porque
                                 o cliette JAVAPPS tem múltiplas subtarefas
   START_PRIVATE_PORT=5100      <= Deve ser um número de porta exclusivo
   START_PUBLIC_PORT=5300       <= Deve ser um número de porta exclusivo
   EXEC_NAME=launchjv.bat       <= O nome do arquivo em lote que inclui as
                                 instruções classpath
   }
   ```

   Quando você inicia o Net.Data Connection Manager, o Net.Data inicia o cliette Java especificado no arquivo de configuração. O cliette se torna disponível para processar solicitações de ambiente de linguagem Java de seus aplicativos macro do Net.Data.

3. Atualize a instrução de caminho DTW_JAVAPPS ENVIRONMENT no arquivo de inicialização do Net.Data, `db2www.ini`, incluindo cada nome de cliette na instrução. Por exemplo:

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

Criação e Execução do arquivo de macro

Após você ter criado a função Java, definido a classe do cliette, e configurado o Net.Data, você pode executar o arquivo de macro que contém referências à função Java.

1. Crie um arquivo de macro que chama suas funções Java. Por exemplo, a chamada de função *myfctcall* chama a função de exemplo fornecida com o Net.Data, usando o cliette DTW_JAVAPPS.

```
%function (DTW_JAVAPPS) myfctcall( ....parâmetros do arquivo de macro ....)

%{ chamar a amostra fornecida com o Net.Data %}
%function (DTW_JAVAPPS) reverse_linel(str);

%HTML_REPORT{
você deve ver a cadeia "Hello World" ao contrário.
@reverse_line("Hello World")
Você deve ter o resultado de sua chamada de função.
@myfctcall( ... ....)
%}
```

2. Inicie o Connection Manager. Consulte o capítulo de desempenho no *Guia de Programação e Administração do net.Data* para mais informações sobre o Connection Manager.
3. Lance a macro e chame o Net.Data usando uma ligação HTML, forma HTML ou padrão URL. Por exemplo, chame o arquivo de macro do Net.Data, `mymacro.d2w`, com o seguinte padrão URL:

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

Ambiente de Linguagem ODBC

O ambiente de linguagem Open Database Connectivity (ODBC) executa as instruções SQL através de uma interface ODBC. A ODBC é baseada na especificação X/Open SQL CAE, que permite que um único aplicativo acesse os diversos sistemas de gerenciamento de bancos de dados.

Para usar o ambiente de linguagem ODBC, você deve ter um controlador ODBC e um gerenciador de controladores. A documentação de seu controlador ODBC descreve como instalar e configurar seu ambiente ODBC.

Enviar instruções SQL em um ambiente ODBC é semelhante a outras funções do Net.Data. O seguinte exemplo é uma macro do Net.Data que envia múltiplas instruções SQL para o banco de dados que é sua fonte de dados ODBC. Os sistemas operacionais que usam a variável DATABASE devem especificar o mesmo banco de dados que a fonte de dados no arquivo 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 de Linguagem Oracle

O ambiente de linguagem Oracle fornece acesso nativo a seus dados Oracle. Você pode acessar as tabelas Oracle do Net.Data executando em modo CGI, FastCGI, NSAPI, ISAPI, ou GWAPI. Este ambiente de linguagem suporta o Oracle 7.2, 7.3, e 8.0.

Restrições:

- Procedimento Armazenado não são suportados através deste ambiente de linguagem.
- A variável DATABASE não é usada para acessar bancos de dados Oracle.
- A variável LOGIN deve conter o nome da ocorrência de banco de dados Oracle. Por exemplo, *ora73* é o nome da ocorrência definido na seguinte variável LOGIN:

  ```
  LOGON=admin@ora73
  ```

Para acessar o Oracle do Net.Data

1. Verifique se a instrução ENVIRONMENT no arquivo de inicialização do Net.Data está correta para o ambiente de linguagem Oracle. Consulte o

capítulo de configuração no *Guia de Programação e Administração do Net.Data* para ver as etapas e exemplos.

2. Certifique-se de que os componentes apropriados do Oracle estão instalados e funcionando conforme segue:

   a. Instale o SQL*Net na máquina em que o Net.Data está instalado, se já não estiver instalado. Para mais informações, consulte a seguinte URL:

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

   b. Verifique se a função *tnsping* do Oracle pode ser usada com a mesma autorização de segurança que seu servidor Web usa. Para verificar, efetue logon com a ID de usuário do seu servidor Web e digite:

      `tnsping` *`nome-da-ocorrência-de-oracle`*

      Onde *nome-da-ocorrência-de-oracle* é o nome do sistema Oracle que suas macros do Net.Data acessa.

      Você pode não conseguir verificar a função *tnsping* no Windows NT se seu servidor Web executar sob autoridade de sistema. Se for o caso, salte esta etapa.

   c. Verifique se as tabelas do Oracle podem ser acessadas com a mesma autorização de segurança que seu servidor Web usa. Para verificar, digite uma instrução SQL SELECT, usando a ferramenta de comando de linha SQL*Plus, para acessar uma tabela Oracle com uma instrução SQL SELECT com a autoridade do seu servidor Web. Por exemplo:

      `SELECT * FROM` *`nome da tabela`*

      Você pode não conseguir verificar o acesso a tabelas no Windows NT se seu servidor Web executar sob autoridade de sistema. Se for o caso, salte esta etapa.

   Detecção de Problemas: Não prossiga se as etapas acima falharem. Se qualquer uma das etapas falhar, verifique sua configuração do Oracle.

3. Certifique-se de que as variáveis de ambiente do Oracle estejam definidas corretamente no seu processo de servidor Web.

   - Para AIX, coloque as seguintes linhas no arquivo `/etc/environment`:

     `ORACLE_SID=`*`nome-da-ocorrência-de-oracle`*
     `ORACLE_HOME=`*`diretório-da-biblioteca-de-tempo-de-execução-do-oracle`*

   - Para o Windows NT, use o painel de Controle de Propriedades do Sistema para incluir as seguintes variáveis de ambiente:

     `ORACLE_SID=`*`nome-da-ocorrência-de-oracle`*
     `ORACLE_HOME=`*`diretório-da-biblioteca-de-tempo-de-execução-do-oracle`*

   Dica: Você pode precisar de linhas adicionais para outras variáveis de ambiente do Oracle, dependendo dos recursos do Oracle que você pretende utilizar, como suporte a idiomas nacionais e commit de dupla fase. Consulte a documentação de administração do Oracle para mais informações sobre estas variáveis de ambiente.

4. Teste a conexão para o Oracle a partir do Net.Data. No seu arquivo de macro do Net.Data, especifique os valores apropriados nas variáveis LOGIN e PASSWORD. *Não* defina a variável DATABASE quando for acessar bancos de dados do Oracle. O seguinte é um exemplo de uma instrução de conexão em um arquivo de macro:

```
%DEFINE LOGIN=user_ID@nome-da-ocorrência-remota-de-oracle
%DEFINE PASSWORD=senha
```

Ocorrências locais do Oracle:

Se você acessar apenas a ocorrência local do Oracle, não especifique o nome da ocorrência-remota-do-oracle como parte da ID que efetua o login do usuário, como no seguinte exemplo:

```
%DEFINE LOGIN=user_ID
%DEFINE PASSWORD=senha
```

Live Connection:

Se você usar o Live Connection, você pode especificar LOGIN e PASSWORD no arquivo de configuração do Live Connection, apesar de não ser recomendado por motivos de segurança. Por exemplo:

```
LOGIN=ID_do_usuário
PASSWORD=senha
```

Dica: Não especifique a variável DATABASE para o Oracle.

5. Teste sua configuração executando um procedimento de base CGI para assegurar que a ocorrência de Oracle pode ser acessada de seu servidor Web, como no seguinte exemplo:

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

Alternativamente, você pode executar *tnsping* diretamente a partir de uma macro do Net.Data, como no seguinte exemplo:

```
%DEFINE testora = %exec "tnsping nome-da-ocorrência-de-oracle"
%HTML (report){
< P>Prestes a testar o acesso ao Oracle com tnsping
< hr>
$(testora)
< hr>
< P>O teste de Oracle está concluído.
%}
```

Detecção de Problemas:

Se a etapa de verificação falhar, verifique se todas as etapas precedentes foram bem-sucedidas verificando os seguintes itens:

- Verifique sua configuração do Oracle.
- Verifique se a sintaxe da variável de ambiente Oracle está correta e se não estão faltando variáveis.
- Verifique a conexão do Oracle, certificando-se de que você digitou a ID de usuário e a senha corretamente.

Se a etapa de verificação ainda assim falhar, entre em contato com o Suporte IBM.

Exemplo:

Após completar as etapas de verificação de acesso, você pode fazer chamadas ao ambiente de linguagem Oracle com funções no arquivo de macro, como no seguinte exemplo:

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

Ambiente de Linguagem Perl

O ambiente de linguagem Perl pode interpretar scripts Perl internos que estão especificados em um bloco FUNCTION da macro Net.Data ou pode executar scripts Perl externos armazenados em arquivos separados no servidor. Chamadas a scripts externos Perl são identificadas em um bloco FUNCTION por uma instrução EXEC, por exemplo:

```
%EXEC{ nome-do-script-perl [parâmetros opcionais] %}
```

O ambiente de linguagem Perl não pode passar diretamente ou recuperar variáveis Net.Data, por isto elas são disponibilizadas para scripts Perl desta maneira:

- O Net.Data passa os parâmetros de entrada para o script Perl como variáveis de ambiente. O script Perl pode recuperar os parâmetros lendo a matriz associativa Perl.
- O script Perl passa os parâmetros de saída de volta para o ambiente de linguagem gravando em um canal nomeado cujo nome do Net.Data passa na variável de ambiente, DTWPIPE. Use a sintaxe da instrução DEFINE para gravar os dados no canal nomeado:

  nome = valor

  Para itens de dados múltiplos, separe cada item com um caractere de nova linha ou espaço em branco.

  Se um nome de variável for o mesmo que de um parâmetro de saída e usar a sintaxe acima, o novo valor substitui o valor atual. Se o nome da variável não corresponder a um parâmetro de saída, o Net.Data o ignora.

O seguinte exemplo mostra como o Net.Data passa as variáveis de um arquivo de macro.

```
%FUNCTION(DTW_PERL) today() RETURNS(result) {
  $date = ãdataã;
  chop
$date;
  open(DTW, "> $ENV{DTWPIPE}") || die "Não pode abrir: $!";
  print DTW "result = \"$date\"\n";
%}
%HTML(INPUT) {
  @today()
%}
```

Se o script Perl estiver em um arquivo externo chamado `today.prl`, a mesma função pode ser escrita como no seguinte exemplo:

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

Os programas do ambiente de linguagem Perl acessam os valores de um parâmetro de tabela através de seu nome Net.Data. Os títulos de coluna para a tabela T são T_N_*i*, e os valores de campo são T_V_*i_j*. O número de linhas e colunas na tabela T são T_ROWS e T_COLS.

Os blocos REPORT e MESSAGE são permitidos como em qualquer seção FUNCTION. Eles são processados pelo Net.Data, não pelo ambiente de linguagem. Um programa Perl pode, entretanto, gravar texto no fluxo de saída padrão e manipular a forma HTML de saída diretamente.

Dica de Autorização: Certifique-se que o servidor Web tem direitos de acesso a quaisquer arquivos externos executáveis referidos por este ambiente de linguagem, inclusive a versão correta do interpretador Perl. Consulte a seção sobre especificação de direitos de acesso de servidor Web a arquivos do Net.Data no capítulo de configuração do *Guia de Programação e Administração do Net.Data* para mais informações.

Ambiente de Linguagem REXX

O ambiente de linguagem REXX pode interpretar programas REXX internos que estão especificados em um bloco FUNCTION da macro Net.Data ou pode executar programas REXX externos armazenados em arquivos separados. Chamadas a programas REXX externos são identificadas em um bloco FUNCTION por uma instrução, por exemplo:

```
%EXEC{ nome-do-arquivo-de-programa-REXX [parâmetros opcionais] %}
```

O ambiente de linguagem REXX usa a API RexxStart() para dizer para o interpretador REXX executar o arquivo especificado, depois passa os parâmetros que seguem o nome do arquivo para o programa como se eles fossem digitados na linha de comandos. Para o programa REXX, todos os parâmetros são recebidos como ARG[1].

Dica de Autorização: Certifique-se que o servidor Web tem direitos de acesso a quaisquer arquivos externos executáveis referidos por ambientes de linguagem. Consulte a seção sobre especificação de direitos de acesso de servidor Web a arquivos do Net.Data no capítulo de configuração do *Guia de Programação e Administração do Net.Data* para mais informações.

Substituição de Variável:

A substituição de variável é realizada apenas na seção instruções executáveis do bloco FUNCTION. Os parâmetros, entretanto, são tornados acessíveis ao programa REXX independentemente de o programa ser definido internamente em um bloco FUNCTION ou externamente em um arquivo separado. O ambiente de linguagem REXX usa a função RexxVariablePool() do processador de linguagem REXX para compartilhar variáveis do Net.Data com o programa REXX. Isto permite ao programa REXX manipular diretamente as variáveis Net.Data identificadas na lista de parâmetros.

Um programa REXX acessa os valores de um parâmetro de tabela como variáveis stem do REXX. Para um programa REXX, os títulos de coluna para a tabela T são T_N.*i* e os valores de campo são T_V.*i.j*. Os números de linhas e colunas na tabela T são T_ROWS e T_COLS.

Desenvolvendo o sistema operacional do AIX:

Se voc6e tem muitas chamadas de ambiente de linguagem REXX no seu sistema AIX, considere a definição da variável de ambiente RXQUEUE_OWNER_PID para 0. Macros que realiza muitas chamadas de ambiente de linguagem REXX que pode desenvolver facilmente muitos processo, desenvolver recursos de sistema.

Você pode definir variável de ambiente em três modos:

- No arquivo de macro usando a função de geração DTW_SETENV :

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

- No arquivo de ambiente de sistema AIX inserindo as seguintes instruções:

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

  Este método afeta este procedimento do REXX para uma máquina completa.

- No arquivo de ambiente do servidor Web HTTP; por exemplo, para o Domino Go Webserver, insira a seguinte instrução:

  ```
  InheritEnv RXQUEUE_OWNER_PID = 0
  ```

  Este método afeta este procedimento do REXX ao servidor Web.

Ambiente de Linguagem SQL

O ambiente de linguagem SQL é usado para executar instruções SQL através do DB2. O resultado de uma instrução SQL pode ser retornado na tabela padrão do Net.Data ou em uma tabela especificada por você.

O Net.Data suporta qualquer instrução SQL que você autorizar. Você pode conectar-se a um banco de dados para cada seção HTML quando chama para a execução do Net.Data como um aplicativo CGI e deve especificar o nome do banco de dados com a variável DATABASE (exceto com o OS/390). Se seu banco de dados DB2 estiver na mesma máquina que seu servidor Web, nenhuma configuração adicional é requerida. Caso contrário, dependendo do sistema operacional utilizado, você pode acessar bancos de dados remotos utilizando o Client Application Enabler (CAE) ou usar os Database Connection Services (DBCS) para receber todo o suporta a transações que o DB2 suporta. Você poderá também utilizar o DataJoiner para acessar outros bancos de dados. A utilização do DataJoiner permite que você use commit de dupla fase com bancos de dados que o suportam.

Ambiente de Linguagem Sybase

O ambiente de linguagem Sybase fornece acesso nativo a seus dados Sybase. Você pode acessar tabelas Sybase do Net.Data executando em modo CGI, FastCGI, NSAPI, ISAPI, ou GWAPI.

Restrições:

- O ambiente de linguagem Sybase não suporta objetos grandes, como imagens e áudio. Os procedimentos armazenado são suportados apenas para procedimentos sem uma instrução SELECT.
- O ambiente de linguagem Sybase requer que o Live Connection utilize FastCGI.

Para acessar o Sybase do Net.Data

1. Verifique se a instrução ENVIRONMENT no arquivo de inicialização do Net.Data está correta para o ambiente de linguagem Sybase. Consulte o capítulo de configuração no *Guia de Programação e Administração do Net.Data* para ver as etapas e exemplos.
2. Certifique-se de que os componentes apropriados do Sybase estão instalados e funcionando conforme segue:
   a. Instale o Sybase Open Client na máquina em que o Net.Data está instalado, se já não estiver instalado. Para mais informações, consulte a documentação do Sybase Open Client.
   b. Verifique se a função *ping* do Sybase pode ser usada com a mesma autorização de segurança que seu servidor Web usa. Para verificar, efetue logon com a ID de usuário do seu servidor Web e digite:

      ```
      ping nome-da-ocorrência-de-Sybase
      ```

      Onde *nome-da-ocorrência-de-sybase* é o nome do sistema Sybase que suas macros do Net.Data acessam.

      Você pode não conseguir verificar a função *ping* no Windows NT se seu servidor Web executar sob autoridade de sistema. Se for o caso, salte esta etapa.
   c. Verifique se as tabelas do Sybase podem ser acessadas com a mesma autorização de segurança que seu servidor Web usa. Para verificar, digite uma instrução SQL SELECT, usando a ferramenta de comando de linha ISQL, para acessar uma tabela Sybase com a autoridade do seu servidor Web. Por exemplo:

      ```
      SELECT * FROM tablename
      ```

      Você pode não conseguir verificar o acesso a tabelas no Windows NT se seu servidor Web executar sob autoridade de sistema. Se for o caso, salte esta etapa.

   Detecção de Problemas: Não prossiga se as etapas acima falharem. Se qualquer uma das etapas falhar, verifique sua configuração do Sybase.
3. Certifique-se de que as variáveis de ambiente do Sybase estejam definidas corretamente no seu processo de servidor Web.
   - Para AIX, coloque as seguintes linhas no arquivo `/etc/environment`:

     ```
     DSQUERY=nome-da-ocorrência-de-sybase
     SYBASE=diretório-da-biblioteca-de-tempo-de-execução-do-sybase
     ```
   - Para o Windows NT, use o painel de Controle de Propriedades do Sistema para incluir as seguintes variáveis de ambiente:

     ```
     DSQUERY=nome-da-ocorrência-de-sybase
     SYBASE=diretório-da-biblioteca-de-tempo-de-execução-do-sybase
     ```

   Dica: Você pode precisar de linhas adicionais para outras variáveis de ambiente do Sybase, dependendo dos recursos do Sybase que você pretende utilizar, como suporte a idiomas nacionais e commit de dupla fase. Consulte a documentação de administração do Sybase para mais informações sobre estas variáveis de ambiente.
4. Teste a conexão para o Sybase a partir do Net.Data. No seu arquivo de macro do Net.Data, especifique os valores apropriados nas variáveis LOGIN,

PASSWORD e DATABASE. O seguinte é um exemplo de uma instrução de conexão em um arquivo de macro:

```
%DEFINE DATABASE=nome-do-banco-de-dados
%DEFINE LOGIN=user_ID@nome-da-ocorrência-remota-do-sybase
%DEFINE PASSWORD=senha
```

Live Connection: Se você usar o Live Connection, você pode especificar LOGIN e PASSWORD no arquivo de configuração do Live Connection, apesar de não ser recomendado por motivos de segurança. Por exemplo:

```
DEFINE DATABASE=nome-do-banco-de-dados
LOGIN=ID_do_usuário
PASSWORD=senha
```

5. Teste sua configuração executando um procedimento de base CGI para assegurar que a ocorrência do Sybase pode ser acessada de seu servidor Web, como no seguinte exemplo:

```
#! /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 tablename
EOFF
echo
```

Detecção de Problemas:

Se a etapa de verificação falhar, verifique se todas as etapas precedentes foram bem-sucedidas verificando os seguintes itens:

- Verifique sua configuração do Sybase
- Verifique se a sintaxe da variável de ambiente Sybase está correta e se não estão faltando variáveis.
- Verifique a conexão do Sybase, certificando-se de que você digitou a ID de usuário e a senha corretamente.

Se a etapa de verificação ainda assim falhar, entre em contato com o Suporte IBM.

Exemplo:

Após completar as etapas de verificação de acesso, você pode fazer chamadas ao ambiente de linguagem Sybase com funções no arquivo de macro, como no seguinte exemplo:

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

Ambiente de Linguagem System

O ambiente de linguagem System é um ambiente definido pelo Net.Data que suporta chamadas a programas externos que estão identificados em uma instrução EXEC no bloco FUNCTION.

O ambiente de linguagem System interpreta as instruções EXEC transmitindo o nome e parâmetros do programa ao sistema operacional para execução através da chamada de função system() de linguagem C. Este método não permite que o programa externo passe variáveis diretamente de e para o Net.Data, como faz o ambiente de linguagem REXX, por isso o Net.Data processa as variáveis pelo seguinte método:

- O Net.Data passa os parâmetros de entrada para o programa externo como as variáveis de ambiente e o ambiente externo as recupera:
  - Um script UNIX CSHELL se refere às variáveis de ambiente precedendo o nome da variável com um sinal de cifrão ($), como em `$x`.
  - Um script de linguagem Perl se refere a elas através de referências à ENV da matriz associativa, como `%ENV{'x'}`.
  - Um arquivo em lote DOS se refere ao nome da variável encapsulado entre sinais de porcentagem (%), como em `%x%`.
- A maioria dos sistemas operacionais passa os parâmetros de saída de volta ao ambiente de linguagem escrevendo em um canal nomeado cujo nome o Net.Data passa na variável de ambiente, DTWPIPE. (O Net.Data for OS/400 passa os parâmetros de volta para o ambiente de linguagem usando variáveis de ambiente.) Use a sintaxe da instrução DEFINE para gravar dados no canal nomeado:

  ```
  nome = valor
  ```

  Para itens de dados múltiplos, separe cada item com um caractere de nova linha ou espaço em branco.

  Se um nome de variável for o mesmo que de um parâmetro de saída, o novo valor substitui o valor atual. O Net.Data ignora nomes de variáveis que não correspondem a nenhum parâmetro de entrada.

Os programas do ambiente de linguagem system acessam os valores de um parâmetro de tabela através de seu nome Net.Data. Os títulos de coluna para a tabela T são T_N_*i*, e os valores de campo são T_V_*i*_*j*. O número de linhas e colunas na tabela T são T_ROWS e T_COLS.

Dica de Autorização: Certifique-se que o servidor Web tem direitos de acesso a quaisquer arquivos externos chamados pelo ambiente de linguagem System. Consulte a seção sobre especificação de direitos de acesso de servidor Web a arquivos do Net.Data no capítulo de configuração do *Guia de Programação e Administração do Net.Data* para mais informações.

Ambiente de Linguagem de Registro Web

O Registro Web do Net.Data fornece armazenamento persistente para dados relativos a aplicativos. Um registro Web pode ser usado para armazenar informações de configuração e outros dados que podem ser acessados dinamicamente no tempo de execução por aplicativos baseados em Web. Você pode acessar registros Web através macro do Net.Data utilizando o Net.Data e o suporte embutido de registro Web e a partir de programas CGI escritos para este fim. O registro Web está disponível em um subconjunto de sistemas operacionais. Consulte o apêndice no manual de sistemas operacionais do Net.Data *Manual do Net.Data*

O desenvolvimento de página Web padrão requer que URLs sejam colocadas diretamente na fonte HTML da página. Isto torna difícil a mudança de links. A natureza estática também limita o tipo de links que podem ser facilmente colocados em uma página Web. A utilização de um registro Web para armazenar dados relacionados a aplicativos, por exemplo URLs, pode auxiliar na criação de páginas HTML com links dinamicamente definidos.

As informações podem ser armazenadas e mantidas em um registro por desenvolvedores de aplicativos e administradores Web que têm acesso ao registro. Os aplicativos recuperam as informações de seus registros associados no tempo de execução. Isto facilita o projeto de aplicativos flexíveis e também permite o movimento de aplicativos e servidores. Você pode usar macros do Net.Data para criar páginas HTML usando as ligações dinamicamente definidas.

As informações são armazenadas em um registro Web na forma de entradas do registro. Cada entrada de registro consiste de um par de cadeias de caracteres: uma cadeia RegistryVariable e uma cadeia RegistryData correspondente. Qualquer informação que possa ser representada por um par de cadeias pode ser armazenada como uma entrada de registro. O Net.Data utiliza a cadeia variável como uma chave de pesquisa para localizar e recuperar entradas específicas de um registro.

Você pode ver um exemplo do conteúdo de um registro Web em Tabela 3.

Tabela 3. Registro Web de Exemplo

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

Eis algumas razões para considerar a utilização de um registro Web:

- Você pode utilizar um registro Web para armazenar aliases para servidores e URLs, facilitando a relocalização de aplicativos e servidores.
- Os desenvolvedores de aplicativos podem enviar seus aplicativos baseados em Web com dados, como URLs, pré-definidos no registro. O usuário final pode alterar os dados do registro para alterar o comportamento do aplicativo.

- Um registro Web pode ser utilizado para realizar pesquisas URL baseadas no nome do produto, idioma nacional, fabricante, e assim por diante.

Entradas indexadas no registro Web são entradas cujas cadeias RegistryVariable têm uma cadeia Index adicional anexada, utilizando a seguinte sintaxe:

```
RegistryVariable/Index
```

O usuário fornece o valor da cadeia Index em um parâmetro separado para uma função interna feita para funcionar com entradas indexadas. Múltiplas entradas indexadas de registro podem ter o mesmo valor de cadeia RegistryVariable, mas elas podem manter sua exclusividade tendo valores de cadeia Index distintos.

Tabela 4. Registro Web Indexado de Exemplo

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

Apesar das duas entradas indexadas acima terem o mesmo valor de cadeia RegistryVariable `Smith`, a cadeia de indexação é diferente em cada caso. Elas são tratadas como duas entradas distintas pelas funções de registro Web.

Configuração de Ambientes de Linguagem do Net.Data

Antes de poder utilizar os ambientes de linguagem com o Net.Data, você deve configurar o arquivo de inicialização do Net.Data, e se você estiver utilizando o Live Connection, o arquivo de configuração do Live Connection.

Segue uma visão geral destas tarefas. Consulte o capítulo de configuração no *Guia de Programação e Administração do Net.Data* para mais informações detalhadas sobre como configurar os ambientes de linguagem do Net.Data. Adicionalmente, verifique as seções de ambiente de linguagem anteriores para obter instruções de configuração especiais.

- Verifique ou atualize a instrução ENVIRONMENT no arquivo de inicialização do Net.Data, `db2www.ini`. Estas instruções estão descritas no capítulo "Configuração do Net.Data" no *Guia de Programação e Administração do Net.Data*.
- Defina cliettes bancos de dados ou o ambiente de linguagem de aplicativos Java no arquivo do Live Connection, `dtwcm.cnf`. a definição de cliette está descrita no capítulo de configuração do *Guia de Programação e Administração do Net.Data*.

Parte 2. Ambientes de Linguagem Não-IBM

Além de fornecer ambientes de linguagem, o Net.Data permite que você inclua seus próprios ambientes de linguagem e banco de dados. O Net.Data acessa seus ambientes de linguagem como bibliotecas de ligação dinâmica ou bibliotecas compartilhadas, separadamente do arquivo executável do Net.Data. Cada ambiente de linguagem deve suportar um conjunto de interfaces definidas pelo Net.Data. Os seguintes capítulos discutem como criar um novo ambiente de linguagem e descrevem a interface e os ambientes de programação de ambientes de linguagem.

Capítulo 3, “Criação de um Novo Ambiente de Linguagem” na página 37

“Projeto da Instrução do Ambiente de Linguagem” na página 46

Capítulo 4, “As Funções Utilitárias da Interface de Programação de Ambientes de Linguagem” na página 49

Capítulo 3. Criação de um Novo Ambiente de Linguagem

O Net.Data usa ambientes de linguagem como interfaces de linguagem de programação e de bancos de dados assecíveis, acessadas como arquivos DLL ou bibliotecas compartilhadas. O Net.Data fornece um conjunto de ambientes de linguagem, mas quando estes não atenderem as necessidades de seus aplicativos, você pode criar novos ambientes de linguagem. Antes de decidir criar um novo ambiente de linguagem, determine se os ambientes de linguagem fornecidos pela IBM que são enviados com o Net.Data satisfazem suas necessidades.

Quando você decide criar um novo ambiente de linguagem, você deve completar as seguintes etapas:

- Determinar quais interfaces e funções você deve fornecer para o ambiente de linguagem. A interface dtw_execute deve ser fornecida, e todas as interfaces fornecidas devem corresponder exatamente os protótipos que estão definidos no cabeçalho de linguagem C dtwle.h.
- Criar um arquivo de construção para construir a DLL ou biblioteca compartilhada.
- Construir uma DLL ou biblioteca compartilhada que implementa o conjunto de rotinas de interface de ambiente de linguagem que você deseja fornecer. Consulte a documentação do C ou C++ para seu sistema operacional para aprender como criar arquivos de construção e construir DLLs ou bibliotecas compartilhadas.
- Tornar todas as interfaces externamente disponíveis da DLL ou biblioteca compartilhada para que o Net.Data possa chamá-las.
- Determinar sua instrução de configuração ENVIRONMENT, e depois incluí-la no arquivo de inicialização do Net.Data. Consulte “Projeto da Instrução do Ambiente de Linguagem” na página 46 para obter detalhes.
- Incluir funções no arquivo de macro do Net.Data que usa o novo ambiente de linguagem.

Este capítulo descreve como projetar o ambiente de linguagem.

- “Projeto de uma DLL ou Biblioteca Compartilhada”
- “Estruturas de Comunicação de Ambientes de Linguagem” na página 40
- “Funções de Interface de Ambientes de Linguagem” na página 43
- “Projeto da Instrução do Ambiente de Linguagem” na página 46

Projeto de uma DLL ou Biblioteca Compartilhada

Quando você constrói um ambiente de linguagem, você atualiza o gabarito fornecido em Apêndice A, “Gabarito de Ambiente de Linguagem” na página 65 para incluir as funções de interface de ambiente e as estruturas de comunicação usadas pelo Net.Data para se comunicar com seu ambiente de linguagem e para passar os parâmetros para o ambiente de linguagem.

As seguintes seções descrevem assuntos de conceitos e projeto para as funções e estruturas. Os utilitários fornecidos na interface do ambiente de linguagem são

descritos em Capítulo 4, “As Funções Utilitárias da Interface de Programação de Ambientes de Linguagem” na página 49.

- “Quais Interfaces de Ambiente de Linguagem Devo Fornecer?”
- “Processamento de Parâmetros de Entrada”
- “Processamento de Solicitações de Usuário” na página 39
- “Processamento de Parâmetros de Saída” na página 39
- “Comunicação de Condições de Erro” na página 39

Quais Interfaces de Ambiente de Linguagem Devo Fornecer?

Quando você escreve um ambiente de linguagem, você deve determinar quais interfaces fornecer. Suas opções dependem de que você deseja que o ambiente de linguagem faça. Por exemplo, se o ambiente de linguagem for acessar dados de banco de dados, você fará diferentes escolhas do que se for para uma linguagem de script. A seguinte seção descreve as interfaces de ambiente de linguagem do Net.Data.

dtw_execute() Você deve fornecer a interface dtw_execute() para passar os parâmetros de entrada do arquivo de macro; é a única interface necessária para todos os ambientes de linguagem. O Net.Data passa todos os parâmetros para dtw_execute() através da estrutura de comunicação do ambiente de linguagem, dtw_lei .

dtw_initialize() Fornecer a interface dtw_initialize() para alocar ou inicializar dados. O Net.Data chama esta interface apenas uma vez para cada chamada para execução da macro, antes da primeira chamada de função a seu ambiente de linguagem. Se não houver chamadas de função para seu ambiente de linguagem, o Net.Data não chama a interface dtw_initialize().

dtw_cleanup()

Fornecer a interface dtw_cleanup() quando você fornece a interface dtw_initialize(), e você quer permitir manipulação de erros quando a macro termina de maneira anormal. O Net.Data chama esta interface apenas uma vez para cada chamada para execução da macro.

dtw_getNextRow() Fornecer a interface dtw_getNextRow() como parte de um ambiente de linguagem de banco de dados ou um ambiente de linguagem que pode processar dados uma linha de cada vez. Esta interface é chamada se o Net.Data estiver executando nos sistemas operacionais OS/400 ou OS/390, apenas.

Processamento de Parâmetros de Entrada

Os ambientes de linguagem do Net.Data usam a interface dtw_execute() para receber e processar parâmetros. A interface dtw_execute funciona com a estrutura dtw_lei, que é usada pelo Net.Data para se comunicar com o ambiente de linguagem. Use as seguintes recomendações para o processamento de parâmetros, quando for escrever seu ambiente de linguagem.

- Especifique quaisquer parâmetros implícitos no arquivo de inicialização. O Net.Data passa os parâmetros especificados aqui em todas as chamadas de função ao ambiente de linguagem após passar os parâmetros especificados pelo transcritor macro no bloco FUNCTION em execução.

- Receber os parâmetros de entrada para a interface dtw_execute() como parte da estrutura dtw_lei. O transcritor da macro determina a ordem em que o Net.Data passa os parâmetros quando os especifica na definição do bloco FUNCTION da macro do Net.Data.

A rotina processInputParms() no gabarito de programa, em Apêndice A, “Gabarito de Ambiente de Linguagem” na página 65 mostra um método de processar parâmetros de entrada.

Processamento de Solicitações de Usuário

Como um ambiente de linguagem processa uma solicitação do usuário depende de como o ambiente de linguagem recebe a solicitação. O Net.Data fornece várias maneiras diferentes de comunicar uma solicitação a seu ambiente de linguagem:

- Através do nome da função especificado em um bloco FUNCTION. Em cada chamada de função, o Net.Data passa o nome da função ao ambiente de linguagem no campo function_name da estrutura dtw_lei.
- Através da lista de parâmetros do bloco FUNCTION. Você pode especificar que um parâmetro na lista de parâmetros pode indicar uma solicitação do usuário. Em cada chamada de função, o Net.Data passa os parâmetros ao ambiente de linguagem no campo parm_data_array da estrutura dtw_lei.
- Através da seção de instruções executáveis de um bloco FUNCTION. Em cada chamada de função, o Net.Data passa instruções executáveis especificadas no bloco FUNCTION para o ambiente de linguagens no campo exec_statement da estrutura dtw_lei.

Processamento de Parâmetros de Saída

O método que você usa para processar os parâmetros de saída depende inteiramente de seu ambiente de linguagem e como ele processa solicitações do usuário. Entretanto, uma vez que o ambiente de linguagem tenha os dados que ele precisa para retornar para a macro do Net.Data, você pode projetar o ambiente de linguagem para modificar os valores dos parâmetros passados no campo parm_data_array da estrutura dtw_lei. A rotina processOutputParms() no gabarito de programa, em Apêndice A, “Gabarito de Ambiente de Linguagem” na página 65, mostra uma maneira possível de processar parâmetros de saída, assim como exemplos de como definir valores tanto de parâmetros de cadeia quanto de tabela.

Comunicação de Condições de Erro

O sucesso ou fracasso de uma chamada de função pode ser comunicado através da variável implícita do Net.Data, RETURN_CODE. Esta variável é definida pelo Net.Data após retornar de uma chamada à interface dtw_execute(). Seu valor é definido pelo valor de retorno da própria chamada do dtw_execute(). Este valor é então usado pelo Net.Data para processar o bloco MESSAGE da macro do Net.Data, se tiver sido especificado um para esta chamada de função.

Se você não especificar um bloco MESSAGE, ou não tiver uma entrada em um bloco MESSAGE especificado para manipular o código de retorno de dtw_execute(), o Net.Data exibe o conteúdo do campo default_error_message na estrutura dtw_lei. Este campo pode ser definido pelo ambiente de linguagem a qualquer momento da rotina dtw_execute. A rotina setErrorMessage() no gabarito

de programa, em Apêndice A, “Gabarito de Ambiente de Linguagem” na página 65 mostra um exemplo de como definir o campo default_error_message.

Estruturas de Comunicação de Ambientes de Linguagem

O Net.Data usa duas estruturas para comunicar-se com seu ambiente de linguagem. Seu ambiente de linguagem deve trabalhar com estas estruturas e definir e passar informações dentro destas estruturas.

- dtw_lei
- dtw_parm_data

O Net.Data passa uma estrutura de interface de ambiente de linguagem (por exemplo, dtw_lei) para a função do ambiente de linguagem que ela chama. A estrutura contém, entre outras, coisas, uma matriz de dados de parâmetros que contém uma lista de parâmetros a serem passados à função do ambiente de linguagem. A função do ambiente de linguagem chamada pelo Net.Data processa a solicitação, atualiza os parâmetros na matriz de dados de parâmetros (se for aplicável), e retorna para o Net.Data.

O Net.Data atravessa então a matriz de dados de parâmetro, atualiza suas cópias dos parâmetros para refletirem os novos valores definidos pela função do ambiente de linguagem, e continua o processamento da macro do Net.Data.

A Estrutura dtw_lei

A função de interface de cada ambiente de linguagem recebe um ponteiro para a estrutura dtw_lei. A estrutura dtw_lei tem o seguinte formato:

```
typedef struct dtw_lei {              /* Interface do Amb. de Ling.       */
    char *function_name;              /* Nome do bloco de função     */
    int   flags;                      /* Sinalizadores do A. de Ling.*/

    char *exec_statement;             /* Instruções do A. de Ling.   */

    dtw_parm_data_t *parm_data_array; /* Matriz de parâmetros        */
    char *default_error_message;      /* Mensagem padrão             */
    void *le_opaque_data;             /* Dados específicos do A.L.   */

    void *row;                        /* Para proc. linha-por-vez    */

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

Campos na estrutura dtw_lei:

function_name O campo function_name contém um ponteiro para uma cadeia que contém o nome do bloco de funções. Isto pode ser útil para especificar o nome do bloco FUNCTION em mensagens de erro exibidas pelo ambiente de linguagem.

sinalizadores O campo sinalizador é usado pelo Net.Data para se comunicar com o ambiente de linguagem. Especifique o ponteiro do campo sinalizador realizando uma operação OR usando as seguintes constantes.

- O Net.Data define DTW_STMT_EXEC para dizer para a função de interface dtw_execute() que o campo exec_statement contém o nome de arquivo e os parâmetros de uma instrução EXEC.
- DTW_END_ABNORMAL é definida pelo Net.Data para dizer para a função de interface dtw_cleanup() que uma condição anormal ou imprevista ocorreu e que o ambiente de linguagem deve realizar qualquer limpeza necessária (ou seja, desocupar recursos presos) antes de o Net.Data finalizar.
- DTW_LE_FATAL_ERROR é definia por uma função de interface do ambiente de linguagem para dizer ao Net.Data que ocorreu um erro fatal no ambiente de linguagem. Se este sinalizador for definido, o Net.Data para de processar a macro do Net.Data, chama a função de interface dtw_cleanup() de um ambiente de linguagem ativo com sinalizadores definidos como DTW_END_ABNORMAL, exibe a mensagem padrão, e sai. O sinalizador é verificado apenas se um valor de retorno não nulo for retornado de uma chamada a ambiente de linguagem.
- DTW_LE_MSG_KEEP é definia por uma função de interface do ambiente de linguagem para dizer ao Net.Data que a memória apontada por default_error_message não deve ser desocupada. Se esta constante não for definida, o Net.Data tenta desocupar a memória.
- DTW_LE_CONTINUE é definida pela função de interface dtw_execute() para dizer ao Net.Data para chamar a função de interface dtw_getNextRow(). O Net.Data chama dtw_getNextRow() apenas se o sinalizador estiver definido e o valor de retorno da chamada à função de interface dtw_execute for zero.

exec_statement O campo exec_statement contém um dos seguintes ponteiros:

- Para uma cadeia que contém as instruções executáveis (após substituição de variáveis) do bloco FUNCTION
- Para o nome de arquivo e parâmetros de uma instrução EXEC

parm_data_array O campo parm_data_array contém um ponteiro para uma matriz de estruturas dtw_parm_data. A matriz termina com uma estrutura parm_data que contém zeros. A estrutura dtw_parm_data é utilizada pelo Net.Data para passar variáveis e o valor associado a um ambiente de linguagem e para recuperar quaisquer alterações ao valor da variável que podem ser feitas pelo ambiente de linguagem. Consulte “A Estrutura dtw_parm_data” na página 42 para obter uma descrição da estrutura.

default_error_message O campo default_error_message é definido pelo ambiente de linguagem como uma cadeia de caracteres que descreve uma condição de erro. Se o valor de retorno de uma chamada a uma função de interface de ambiente de linguagem for não-nulo e o valor de retorno não corresponder ao valor da mensagem no bloco MESSAGE, será exibida a mensagem padrão. Caso contrário, o Net.Data exibe a mensagem selecionada do bloco MESSAGE.

le_opaque_data O campo le_opaque_data é definido por qualquer uma das funções de interface no ambiente de linguagem para passar parâmetros de uma função de interface para outra. O Net.Data salva o ponteiro e o

passa para outra função de interface que o Net.Data chama. Após processar a macro do Net.Data, e antes de retornar ao responsável pela chamada do Net.Data, o Net.Data define o ponteiro como NULL. Como o campo é específico para cada subtarefa, ambientes de linguagem podem armazenar dados que são específicos de subtarefa. Use este campo apenas se você tiver a função de interface dtw_cleanup(), de modo que a função possa desocupar a memória associada ao campo le_opaque_data.

row O campo row é definido pelo Net.Data como um objeto linha antes de uma chamada à função de interface dtw_getNextRow() de um ambiente de linguagem. A função dtw_getNextRow() insere uma linha de dados de tabela no objeto usando as funções de interface utilitárias de linha. O Net.Data processa então a linha e chama dtw_getNextRow() até que não haja mais linhas a processar.

O campo reserved é reservado para uso da IBM.

A Estrutura dtw_parm_data

O Net.Data usa a estrutura dtw_parm_data para passar parâmetros para uma ambiente de linguagem. Parâmetros são obtidos de três fontes:

- Parâmetros explícitos que são especificados na definição do bloco FUNCTION
- Parâmetros que são especificados na instrução de configuração ENVIRONMENT no arquivo de inicialização do Net.Data
- A variável de retorno que é especificada na palavra-chave RETURNS em uma definição de bloco FUNCTION

O Net.Data passa os parâmetros explícitos primeiro, seguidos por parâmetros especificados na instrução ENVIRONMENT, e depois a variável de retorno.

A estrutura dtw_parm_data tem o seguinte formato:

```
typedef struct dtw_parm_data {   /* Dados de parâmetro         */
    int   parm_descriptor;            /* Descritor de parâmetros     */
   char *parm_name;                   /* Nome do parâmetro           */
    char *parm_value;                 /* Valor do parâmetro          */
    void *res1;                       /* Reservado                   */
    void *res2;                       /* Reservado                   */
} dtw_parm_data_t;
```

Campos na estrutura dtw_parm_data:

parm_descriptor O campo parm_descriptor descreve o tipo e uso do parâmetro que está sendo passado para o ambiente de linguagem. O Net.Data define o campo realizando uma operação OR usando as seguintes constantes:

- DTW_IN indica que um parâmetro é um parâmetro apenas de entrada.
- DTW_OUT indica que um parâmetro é um parâmetro apenas de saída.
- DTW_INOUT indica que um parâmetro é um parâmetro de entrada e de saída.

- DTW_STRING indica que o valor do parâmetro é um ponteiro para uma cadeia.
- DTW_TABLE indica que o valor do parâmetro é um ponteiro para uma tabela.

  O Net.Data sempre define o campo parm_descriptor como DTW_IN, DTW_OUT, ou DTW_INOUT e usa um OR lógico com DTW_STRING e DTW_TABLE.

parm_name O campo parm_name é um ponteiro para uma cadeia que contém o nome do parâmetro. O Net.Data define este ponteiro como NULL se o parâmetro for uma cadeia literal.

parm_value O campo parm_value é um ponteiro para um objeto que contém o valor do parâmetro. Este ponteiro é definido como NULL pelo Net.Data se o parâmetro for uma variável que ainda não esteja definida.

Os campos res1 e res2 são campos reservados.

Tanto parm_name quanto parm_value apontam para um objeto alocado da *pilha* de tempo de execução do Net.Data, a área da memória utilizada para alocação dinâmica de memória pelo Net.Data. Se parm_name ou parm_value forem substituídas por outra cadeia, a cadeia original deve ser desocupada e substituída por um ponteiro para uma cadeia de caracteres alocada da pilha do Net.Data. Use as funções utilitárias dtw_malloc() e dtw_free() para desocupar a cadeia original.

Funções de Interface de Ambientes de Linguagem

O Net.Data usa quatro funções de interface com um ambiente de linguagem: você fornece uma ou mais destas funções. Três destas funções são opcionais, mas todo ambiente de linguagem deve ter uma função de interface dtw_execute(). Se uma macro do Net.Data fizer referência a um ambiente de linguagem que não tem uma função de interface dtw_execute(), o Net.Data retorna uma mensagem de erro e para de processar a macro do Net.Data.

Para chamar um ambiente de linguagem, refira-se a ele no bloco FUNCTION da macro do Net.Data. As funções de interface de ambiente de linguagem, devem ser chamadas na seguinte ordem:

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

A função dtw_execute() é a única função de interface que você deve fornecer no ambiente de linguagem.

Quando o Net.Data encontra uma chamada a uma função que usa o ambiente de linguagem, ele usa as seguintes etapas para chamar o ambiente de linguagem:

1. O Net.Data chama dtw_initialize() se ela tiver sido definida para este ambiente de linguagem. A função realiza quaisquer tarefas de inicialização solicitadas pelo ambiente de linguagem, como conectar-se a bancos de dados, ou alocação de variáveis.

2. O Net.Data chama dtw_execute() para processar o bloco FUNCTION do arquivo de macro que contém instruções que o ambiente de linguagem deve processar.
3. O Net.Data chama dtw_getNextRow() se, em um retorno bem-sucedido, dtw_execute() tiver indicado que dtw_getNextRow() deve ser chamada.
4. Quando o processamento da macro do Net.Data estiver completo, o Net.Data chama dtw_cleanup() para limpar o ambiente (por exemplo, desconectar-se do banco de dados ou desocupar variáveis) se esta função estiver definida para o ambiente de linguagem, e depois retorna para o servidor Web.

As seguintes seções descrevem as funções de interface.

- “dtw_initialize()”
- “dtw_execute()”
- “dtw_getNextRow()” na página 45
- “dtw_cleanup()” na página 46

dtw_initialize()

A função de interface dtw_initialize() realiza qualquer inicialização especial de que o ambiente de linguagem necessite, como conectar-se a um banco de dados ou alocar variáveis. Esta função de interface é chamada uma vez e é opcional.

O Net.Data chama a função de interface dtw_initialize() de um ambiente de linguagem apenas uma vez, na primeira vez que o Net.Data chama um bloco FUNCTION que faça referência àquele ambiente de linguagem. Referências subseqüentes ao ambiente de linguagem desviam a chamada para a função de interface dtw_initialize().

Esta função de interface não afeta o processamento de blocos de mensagens. Um código de retorno positivo ou nulo significa que o processamento continua; um código de retorno negativo significa que o processamento não continua. Se o código de retorno for não-nulo e houver uma mensagem definida no campo default_error_message, emite-se a mensagem padrão; se não houver mensagem padrão, o Net.Data emite uma mensagem de erro.

dtw_execute()

A função de interface dtw_execute() processa blocos FUNCTION do arquivo de macro que contêm instruções que devem ser processadas pelo ambiente de linguagem. Por exemplo, um bloco FUNCTION que se refere a um ambiente de linguagem de banco de dados contém instruções SQL que o ambiente de linguagem usa para consultar o banco de dados.

A função de interface dtw_execute() é chamada sempre que uma macro do Net.Data processar um bloco FUNCTION que se refere ao ambiente de linguagem. Quando a função de interface dtw_execute() se completa, a próxima coisa a acontecer depende de o ambiente de linguagem estar processando dados de tabela uma linha por vez. Se for, a função de interface define o sinalizador DTW_LE_CONTINUE na estrutura dtw_lei para dizer ao Net.Data para chamar a função de interface dtw_getNextRow(). Consulte “dtw_getNextRow()” na página 45 para mais informações sobre a função de interface dtw_getNextRow() e suas etapas de processamento.

Você pode otimizar o processamento fazendo com que a função de interface dtw_execute() realize todo o processamento necessário para produzir a entrada para o processamento do bloco de relatório. Por exemplo, a função de interface do ambiente de linguagem SQL dtw_execute gera a tabela inteira a ser processada durante a fase de bloco de relatório.

dtw_getNextRow()

A função de interface dtw_getNextRow() recupera entrada para processamento linha-por-vez de tabelas do Net.Data. Ela é chamada cada vez que o sinalizador DTW_LE_CONTINUE é definido, indicando que outra linha de dados precisa ser processada para a tabela. Use dtw_getNextRow() para ambientes de linguagem de banco de dados.

Restrição: Esta função de interface só é chamada se o Net.Data estiver executando em sistemas operacionais OS/400 ou OS/390.

O Net.Data chama dtw_getNextRow() quando as seguintes condições são atingidas:

- A chamada à chamada dtw_execute() do ambiente de linguagem se completa com sucesso (valor de retorno de zero)
- A função de interface dtw_execute() definiu o sinalizador DTW_LE_CONTINUE na estrutura dtw_lei.

Quando a função dtw_execute() define o sinalizador DTW_LE_CONTINUE como ligado, o Net.Data realiza as seguintes etapas:

1. Processa o bloco de mensagens para o valor de retorno da função de interface dtw_execute().
2. Chama a função de interface dtw_getNextRow() de um ambiente de linguagem e inicia o processamento linha-por-vez.
3. Processa o bloco de relatórios.
4. Processa o bloco de mensagens para o valor de retorno da função de interface dtw_getNextRow().
5. Determina se dtw_getNextRow() ligou o sinalizador DTW_LE_CONTINUE:
   - Se sim, o processamento continua com a função de interface dtw_getNextRow() na etapa 2.
   - Se não, o processamento linha-por-vez termina e o Net.Data continua o processamento da macro do Net.Data.

Quando dtw_getNextRow() é chamada, o campo row na estrutura dtw_lei é definido para apontar para um objeto linha. Para manipular o objeto linha, use as funções utilitárias do Net.Data, dtw_row_SetCols() e dtw_row_SetV(). O Net.Data assume que após a primeira chamada à função de interface dtw_getNextRow() o objeto linha contém os títulos de coluna para a tabela. Chamadas subseqüentes contêm os dados reais da tabela.

A função dtw_getNextRow() continua a ser chamada (a menos que o processamento de blocos de mensagens indique o contrário) enquanto o sinalizador DTW_LE_CONTINUE estiver definido.

dtw_cleanup()

Use a função de interface dtw_cleanup() para limpar o ambiente de linguagem se você usa dtw_initialize() para inicializar o ambiente de linguagem. Por exemplo, desconectar-se de um banco de dados ou desocupar variáveis. Esta função de interface é opcional.

Na manipulação de uma solicitação do Net.Data, o Net.Data chama a função de interface dtw_cleanup() de um ambiente de linguagem uma vez quando o processamento do Net.Data termina ou um erro interrompe o processamento pelo Net.Data do arquivo de macro.

O Net.Data define o campo flags na estrutura dtw_lei como DTW_END_ABNORMAL se o processamento de limpeza for anormal. As seguintes condições anormais fornecem exemplos de quando usar dtw_cleanup():

- Uma função de interface de um ambiente de linguagem indica que um erro fatal ocorreu definindo o bit DTW_LE_FATAL_ERROR no campo flags na estrutura dtw_lei.
- O Net.Data encontra um erro irrecuperável.
- O processamento do bloco de mensagens da macro do Net.Data resulta em uma saída.

Se a função de interface de um ambiente de linguagem definir o campo le_opaque_data com um parâmetro a ser passado entre funções de interface, use dtw_cleanup() para livrar o campo quando o processamento for concluído.

Esta função de interface não afeta o processamento de blocos de mensagens. Se o valor de retorno for não-nulo, uma mensagem padrão é emitida; se não existir mensagem padrão, o processador da macro emite uma mensagem de aviso.

Projeto da Instrução do Ambiente de Linguagem

Cada ambiente de linguagem tem uma instrução ENVIRONMENT no arquivo de inicialização, DB2WWW.INI, que contém informações específicas para aquele ambiente de linguagem. Quando você cria um novo ambiente de linguagem, você precisa projetar uma nova instrução de ambiente para o arquivo de inicialização e documentar como usuários devem incluí-la no arquivo de inicialização.

As instruções ENVIRONMENT especificam informações sobre o ambiente de linguagem que o Net.Data requer para chamar e carregar a DLL ou biblioteca compartilhada do ambiente de linguagem, como o nome do ambiente de linguagem, o nome da DLL ou biblioteca compartilhada, e a lista de parâmetros a ser passada para o ambiente de linguagem para cada chamada de função.

O Net.Data lê as informações de configuração quando ele é chamado para execução, mas não carrega DLLs ou bibliotecas compartilhadas do ambiente de linguagem até que um bloco FUNCTION que identifique aquele ambiente de linguagem seja chamado de dentro de arquivo de macro. A DLL permanece carregada até a conclusão do Net.Data.

As seguintes seções fornecem informações sobre sintaxe, descrição de parâmetros, e exemplos que você pode usar em sua documentação.

Sintaxe da Instrução ENVIRONMENT

Uma instrução ENVIRONMENT tem o seguinte formato:

```
ENVIRONMENT(tipo) nome da biblioteca  ([utilização parâmetro, ...)
```

Cada instrução ENVIRONMENT deve estar em uma única linha.

Os seguintes são os parâmetros que você deve especificar para cada ambiente de linguagem:

- *tipo*

  O nome que associa este ambiente de linguagem com a definição de um bloco FUNCTION em uma macro do Net.Data. Você também deve especificar o tipo do ambiente de linguagem em uma definição de bloco FUNCTION para dizer ao Net.Data qual ambiente de linguagem processa a chamada de função. O nome não pode iniciar pelo prefixo `DTW_`. Este prefixo é reservado para ambientes de linguagem enviados com o Net.Data. Consulte a seção "Bloco de Funções" no *Manual do Net.Data* para mais informações sobre o bloco FUNCTION.

- *library_name*

  O nome do objeto que contém as interfaces de ambiente de linguagem chamadas pelo Net.Data. No Windows NT e OS/2, o nome da DLL é especificado sem a extensão *.dll*. No AIX, o nome do objeto compartilhado é especificado com a extensão *.o*, e no OS/400, o nome do programa de serviço é especificado com a extensão *.SRVPGM*. O OS/390 não tem extensões para arquivos DLL. Verifique o arquivo de inicialização enviado com o Net.Data para seu sistema operacional para ver como especificar este nome. Considere utilizar um nome de caminho completo para certificar-se que o Net.Data encontre a DLL ou biblioteca compartilhada.

- *parameter_list*

  A lista de parâmetros que são passados para o ambiente de linguagem em cada chamada de função, além dos parâmetros especificados na definição do bloco FUNCTION. Eles são passados no campo *parm_data_array* da estrutura *dtw_lei* seguindo os parâmetros especificados na definição do bloco FUNCTION. Você deve definir estes parâmetros como variáveis em sua macro do Net.Data antes de a chamada de função ser feita. Se uma função alterar o valor destes parâmetros, os parâmetros retêm o valor modificado quando o processamento da função for concluído.

Exemplos de Instruções ENVIRONMENT

Os seguintes exemplos mostram instruções ENVIRONMENT para ambientes de linguagem que o Net.Data fornece. Estes exemplos ilustram como especificar parâmetros. As variáveis que você inclui nas instruções ENVIRONMENT são das que você deseja que transcritores da macro do Net.Data possam definir ou substituir em suas macros. Consulte as informações específicas para sistemas operacionais nos apêndices no *Manual do Net.Data* ou no ser arquivo README ou Diretório de Programas do Net.Data para obter exemplos adicionais.

Os seguintes exemplos mostram a sintaxe para 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 )
```

Instruções ENVIRONMENT podem variar em cada sistema operacional; por exemplo o OS/390 difere ligeiramente para acesso a SQL e ODBC:

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

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

Capítulo 4. As Funções Utilitárias da Interface de Programação de Ambientes de Linguagem

O Net.Data fornece uma interface de programação para você utilizar no projeto de um novo ambiente de linguagem. A interface de ambiente de linguagem tem funções utilitárias para acessar serviços do Net.Data que gerenciam variáveis de memória e configuração, e fornecem recursos de manipulação de linhas e tabelas. Apêndice A, “Gabarito de Ambiente de Linguagem” na página 65 fornece um gabarito que você pode usar como modelo no projeto de um ambiente de linguagem.

A seguinte seção explica as funções utilitárias da interface do ambiente de linguagem do Net.Data.

Funções Utilitárias do Ambiente de Linguagem

Os ambientes de linguagem podem usar funções utilitárias para acessar serviços do Net.Data. Estas funções caem em quatro categorias:

- “Funções Utilitárias para Gerenciamento de Memória”
- “Funções Utilitárias para Gerenciamento de Variáveis de Configuração” na página 50
- “Funções Utilitárias para Manipulação de Tabelas” na página 50
- “Funções Utilitárias para Manipulação de Linhas” na página 51

Funções Utilitárias para Gerenciamento de Memória

Os ambientes de linguagem usam funções utilitárias de gerenciamento de memória para alocar memória possuída pelo Net.Data, e para desocupar memória que ele alocou usando a biblioteca de tempo de execução do Net.Data.

O seguinte exemplo ilustra a necessidade destas funções utilitárias. Suponha que o Net.Data seja escrito usando o compilador A, com sua biblioteca de tempo de execução correspondente. Um programador escreve um novo ambiente de linguagem, mas usa o compilador B, que tem uma biblioteca de tempo de execução diferente. O ambiente de linguagem não pode desocupar memória que o Net.Data alocou, e o Net.Data não pode desocupar memória que foi alocada pelo ambiente de linguagem por causa de incompatibilidades potenciais entre as duas bibliotecas de tempo de execução.

Tabela 5. Funções Utilitárias de Gerenciamento de Memória

Função Utilitária	Descrição
“dtw_malloc()” na página 53	Alocar memória da pilha de tempo de execução do Net.Data usando dtw_malloc().
“dtw_free()” na página 52	Desocupar memória alocada da pilha de tempo de execução do Net.Data usando dtw_malloc().
“dtw_strdup()” na página 54	Alocar memória da pilha de tempo de execução do Net.Data e copiar a cadeia especificada para a memória alocada usando dtw_malloc().

Funções Utilitárias para Gerenciamento de Variáveis de Configuração

As funções utilitárias de gerenciamento para as variáveis de configuração permitem que ambientes de linguagem acessem informações de configuração armazenadas no arquivo de inicialização do Net.Data. Usando estas funções, todos os ambientes de linguagem podem compartilhar o arquivo de inicialização do Net.Data e usar as informações dele para configurar ambientes de linguagem.

Tabela 6. Configuração de Funções Utilitárias

Funções Utilitária	Descrição
“dtw_getvar()” na página 52	Recuperar o valor de uma variável de configuração do arquivo de inicialização do Net.Data.

Funções Utilitárias para Manipulação de Tabelas

Use as funções de tabela para manipular quaisquer variáveis de tabela macro do Net.Data que sejam passadas ao ambiente de linguagem.

Números de linhas e colunas começam em (1).

Tabela 7. Funções Utilitárias de Tabelas

Função Utilitária	Descrição
“dtw_table_New()” na página 59	Criar um objeto tabela.
“dtw_table_Delete()” na página 55	Eliminar um objeto tabela.
“dtw_table_SetCols()” na página 61	Definir a largura de uma tabela e alocar memória para os títulos de coluna.
“dtw_table_GetV()” na página 57	Recuperar um valor tabela.
“dtw_table_SetV()” na página 62	Definir um valor tabela.
“dtw_table_GetN()” na página 57	Recuperar um título de coluna de tabela.
“dtw_table_SetN()” na página 61	Definir um título de coluna de tabela.
“dtw_table_Rows()” na página 60	Recuperar o atual número de linhas de uma tabela.
“dtw_table_Cols()” na página 55	Recuperar o atual número de colunas de uma tabela.
“dtw_table_MaxRows()” na página 59	Recuperar o número máximo permitível de linhas de uma tabela.
“dtw_table_QueryColnoNj()” na página 60	Recuperar o número de coluna de uma coluna.
“dtw_table_AppendRow()” na página 54	Incluir uma ou mais linhas no final da tabela.
“dtw_table_InsertRow()” na página 58	Inserir uma ou mais linhas em uma tabela.
“dtw_table_DeleteRow()” na página 56	Eliminar uma ou mais linhas de uma tabela.
“dtw_table_InsertCol()” na página 58	Inserir uma ou mais colunas em uma tabela.
“dtw_table_DeleteCol()” na página 56	Eliminar uma ou mais colunas de uma tabela.

Funções Utilitárias para Manipulação de Linhas

As funções utilitárias de linhas manipulam o objeto linha que é passado para a função dtw_getNextRow() de um ambiente de linguagem durante o processamento de uma linha por vez.

Números de linhas começam em (1).

Tabela 8. Funções Utilitárias de Linhas

Função Utilitária	Descrição
“dtw_row_SetCols()” na página 53	Definir a largura de uma linha.
“dtw_row_SetV()” na página 53	Definir um valor tabela.

Manual de Sintaxe de Funções Utilitárias

Esta seção descreve cada uma das funções utilitárias, seu formato, uso, e parâmetros, e também fornece um exemplo simples.

dtw_free()

Uso

Desocupa memória alocada da pilha de tempo de execução do Net.Data usando dtw_malloc(). O buffer aponta para a memória alocada a desocupar.

Formato

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

Parâmetros

buffer	Um ponteiro para a memória alocada a desocupar.

Exemplos

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

myBuf = (char *)dtw_malloc(nbytes);

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

dtw_getvar()

Uso

Recupera o valor de uma variável de configuração especificada por *var_name* do arquivo de inicialização do Net.Data. O Net.Data é o proprietário da memória retornada por dtw_getvar(); não a altere ou desocupe.

Formato

```
char *dtw_getvar(char *var_name)
```

Parâmetros

var_name	O nome da variável de configuração a recuperar.

Exemplos

```
char *myBindFile;

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

dtw_malloc()

Uso

Retorna uma ponteiro para a memória que foi alocada da pilha de tempo de execução do Net.Data usando dtw_malloc(). A memória tem *nbytes* de comprimento. Se o Net.Data não puder retornar a memória solicitada, ele retorna uma ponteiro NULL.

Formato

```
void *dtw_malloc(long nbytes)
```

Parâmetros

nbytes	O número de bytes a alocar.

Exemplos

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

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

dtw_row_SetCols()

Uso

Designa a largura da linha e aloca memória para os títulos de coluna. Você pode usar a função utilitária dtw_row_SetCols() uma vez para cada linha.

Formato

```
int dtw_row_SetCols(void *row, int cols)
```

Parâmetros

row	Um ponteiro para uma linha recém-criada que ainda não alocou colunas.
cols	O número inicial de colunas a alocar na próxima linha.

Exemplos

```
void *myRow;

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

dtw_row_SetV()

Uso

Designa um valor de tabela. O responsável pela chamada da função utilitária dtw_row_SetV() retém a posse da memória apontada por *src*. Para eliminar o atual valor de tabela, designe o valor a NULL.

Formato

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

Parâmetros

row	Um ponteiro para a linha a modificar.
src	Uma cadeia de caracteres que contém um novo valor a definir.
col	O número da coluna do valor a definir.

Exemplos

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

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

dtw_strdup()

Uso

Aloca memória da pilha de tempo de execução do Net.Data e copia a cadeia especificada por *string* na memória alocada usando dtw_malloc(). Se o Net.Data não puder retornar a memória solicitada, ele retorna uma ponteiro NULL.

Formato

```
char *dtw_strdup(char *string)
```

Parâmetros

string	Um ponteiro para o valor de cadeia a copiar na memória alocada.

Exemplos

```
char *myString = "Esta cadeia será duplicada.";
char *myDupString;

myDupString = dtw_strdup(myString);
```

dtw_table_AppendRow()

Uso

Inclui uma ou mais linhas no final da tabela. Designa os valores de tabela das novas linhas com o utilitário dtw_table_SetV() após as linhas serem anexadas ao final da tabela.

Formato

```
int dtw_table_AppendRow(void *table, int rows)
```

Parâmetros

table	Um ponteiro para a tabela aonde anexar.
rows	O número de linhas a serem anexadas.

Exemplos

```
void *myTable;

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

dtw_table_Cols()

Uso

Retorna o número atual de colunas na tabela.

Formato

```
int dtw_table_Cols(void *table)
```

Parâmetros

table	Um ponteiro para a tabela cujo número atual de colunas é retornado.

Exemplos

```
void *myTable;
int currentColumns;

currentColumns = dtw_table_Cols(myTable);
```

dtw_table_Delete()

Uso

Elimina todos os títulos de coluna, valores de tabela, e o objeto de tabela.

Formato

```
int dtw_table_Delete(void *table)
```

Parâmetros

table	Um ponteiro para a tabela a eliminar.

Exemplos

```
void *myTable;

rc = dtw_table_Delete(myTable);
```

---

dtw_table_DeleteCol()

Uso

Elimina uma ou mais colunas começando pela coluna especificada em *start_col*. para eliminar todas as colunas e linhas de uma tabela, substitua a função utilitária dtw_table_Cols() pelo parâmetro *cols*.

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

Formato

```
int dtw_table_DeleteCol(void *table, int start_col, int cols)
```

Parâmetros

table	Um ponteiro para a tabela a alterar.
start_col	O número de coluna da primeira coluna a eliminar.
rows	O número de colunas a serem eliminadas.

Exemplos

```
void *myTable;

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

---

dtw_table_DeleteRow()

Uso

Elimina uma ou mais linhas começando pela linha especificada em *start_row*.

Formato

```
int dtw_table_DeleteRow(void *table, int start_row, int rows)
```

Parâmetros

table	Um ponteiro para a tabela a alterar.
start_row	O número de linha da primeira linha a eliminar.
rows	O número de linhas a serem eliminadas.

Exemplos

```
void *myTable;

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

dtw_table_GetN()

Uso

Recupera um título de coluna. O Net.Data é proprietário da memória apontada por *dest*; não a altere ou desocupe.

Formato

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

Parâmetros

table	Um ponteiro para a tabela de que um título de coluna é recuperado.
dest	Um ponteiro para a cadeia de caracteres a conter o título da coluna.
col	O número da coluna do título da coluna.

Exemplos

```
void *myTable;
char *myColumnHeading;

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

dtw_table_GetV()

Uso

Recupera um valor de uma tabela. O Net.Data é proprietário da memória apontada por *dest*; não a altere ou desocupe.

Formato

```
int dtw_table_GetV(void *table, char **dest, int row, int col)
```

Parâmetros

table	Um ponteiro para a tabela de que um valor é recuperado.
dest	Um ponteiro para a cadeia de caracteres a conter o valor.
row	O número de linha do valor a recuperar.
col	O número de coluna do valor a recuperar.

Exemplos

```
void *myTable;
char *myTableValue;

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

dtw_table_InsertCol()

Uso

Insere uma ou mais colunas após a coluna especificada.

Formato

```
int dtw_table_InsertCol(void *table, int after_col, int cols)
```

Parâmetros

table	Um ponteiro para a tabela a alterar.
after_col	O número da coluna após a qual as novas colunas devem ser inseridas. Para inserir colunas no início da tabela, especifique 0.
cols	O número de colunas a serem inseridas.

Exemplos

```
void *myTable;

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

dtw_table_InsertRow()

Uso

Insere uma ou mais linhas após a linha especificada.

Formato

```
int dtw_table_InsertRow(void *table, int after_row, int rows)
```

Parâmetros

table	Um ponteiro para a tabela a alterar.
after_row	O número da linha após a qual as novas linhas devem ser inseridas. Para inserir linhas no início da tabela, especifique 0.
rows	O número de linhas a serem inseridas.

Exemplos

```
void *myTable;

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

dtw_table_MaxRows()

Uso

Retorna o máximo número de linhas permitido para a tabela do Net.Data conforme definido no parâmetro *row_lim* da função utilitária dtw_table_New().

Formato

```
int dtw_table_MaxRows(void *table)
```

Parâmetros

table	Um ponteiro para a tabela de que o máximo número de linhas é retornado.

Exemplos

```
void *myTable;
int maximumRows;

maximumRows = dtw_table_MaxRows(myTable);
```

dtw_table_New()

Uso

Cria um objeto tabela do Net.Data e inicializa todos os títulos de coluna e valores de campo como NULL. O responsável pela chamada especifica o número inicial de linhas e colunas, e o número máximo de linhas. Se o número inicial de linhas e colunas é 0, você deve usar a função dtw_table_SetCols() para especificar o número de campos em uma linha antes de quaisquer chamadas a funções de tabela.

Formato

```
int dtw_table_New(void **table, int rows, int cols, int row_lim)
```

Parâmetros

table	O nome da nova tabela.
rows	O número inicial de linhas a alocar na nova tabela.
cols	O número inicial de colunas a alocar na nova tabela.
row_lim	O número máximo de linhas que esta tabela pode conter.

Exemplos

```
void *myTable;

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

dtw_table_QueryColnoNj()

Uso

Retorna o número de coluna associado a um título de coluna.

Formato

```
int dtw_table_QueryColnoNj(void *table, char *name)
```

Parâmetros

table	Um ponteiro para a tabela a consultar.
name	Uma cadeia de caracteres especificando o título da coluna para a qual o número da coluna é retornado. Se o título da coluna não existir na tabela, retorna-se 0.

Exemplos

```
void *myTable;
int columnNumber;

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

dtw_table_Rows()

Uso

Retorna o número atual de linhas na tabela.

Formato

```
int dtw_table_Rows(void *table)
```

Parâmetros

table	Um ponteiro para a tabela cujo número atual de linhas é retornado.

Exemplos

```
void *myTable;
     int currentRows;

     currentRows = dtw_table_Rows(myTable);
```

dtw_table_SetCols()

Uso

Define o número de colunas da tabela e aloca memória para os títulos de coluna. Especifique os títulos de coluna quando a tabela é criada; caso contrário, você deve especificá-los chamando esta função utilitária antes de usar quaisquer outras funções de tabela. Você deve usar a função utilitária dtw_table_SetCols() uma vez para uma tabela. Depois, use as funções utilitárias dtw_table_DeleteCol() ou dtw_table_InsertCol().

Formato

```
int dtw_table_SetCols(void *table, int cols)
```

Parâmetros

table	Um ponteiro para uma nova tabela que não tem colunas ou linhas alocadas.
cols	O número inicial de colunas a alocar na nova tabela.

Exemplos

```
void *myTable;

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

dtw_table_SetN()

Uso

Designa um nome a um título de coluna. O responsável pela chamada da função utilitária dtw_table_SetN() retém a posse da memória apontada pelo parâmetro *src*. Para eliminar o título da coluna, designe o valor do título da coluna a NULL.

Formato

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

Parâmetros

table	Um ponteiro para a tabela cujo título de coluna é designado.
src	Uma cadeia de caracteres sendo designada ao novo título de coluna.
col	O número da coluna.

Exemplos

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

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

dtw_table_SetV()

Uso

Designa um valor em uma tabela. O responsável pela chamada da função utilitária dtw_table_SetV() retém a posse da memória apontada pelo parâmetro *src*. Para eliminar o valor de tabela, designe o valor a NULL.

Formato

```
int dtw_table_SetV(void *table, char *src, int row, int col)
```

Parâmetros

table	Um ponteiro para a tabela cujo valor está sendo designado.
src	Uma cadeia de caracteres designada ao novo valor.
row	O número de linha do novo valor.
col	O número de coluna do novo valor.

Exemplos

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

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

Parte 3. Apêndices

Apêndice A. Gabarito de Ambiente de Linguagem

Use este gabarito para criar seus próprios ambientes de linguagem.

```
/*****************************************************************************/
/*                                                                           */
/* Nome do Arquivo                                                           */
/*                                                                           */
/* Descrição                                                                 */
/*                                                                           */
/* Funções                                                                   */
/*                                                                           */
/* Pontos de Entrada                                                          */
/*                                                                           */
/* Alterar Atividade                                                         */
/*                                                                           */
/* Sinal.  Razão       Data      Desenvolved.  Descrição                     */
/* ------  ----------  --------  ------------  ---------------------- */
/*                                                                           */
/*****************************************************************************/

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

Figura 2 (Parte 1 de 14). Gabartio de Ambiente de Linguagem

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

#ifdef _AIX_
//*-------------------------------------------------------------------*/
/* Função                                                            */
/*    dtw_getFp                                                      */
/*                                                                   */
/* Objetivo                                                          */
/*    Definir ponteiros de funções para todas as rotinas da          */
/*    Interface de Ambiente de Linguagem fornecidas por este         */
/*    Ambiente de Linguagem. Se uma rotina da estrutura não estiver  */
/*    sendo fornecida, defina aquele campo como NULL.                */
/*                                                                   */
/* Formato                                                           */
/*    int dtw_getFp(dtw_fp_t *func_pointer)                          */
/*                                                                   */
/* Parâmetros                                                        */
/*    func_pointer     Um ponteiro para uma estrutura que conterá    */
/*                     ponteiros de função para todas as funções     */
/*                     fornecidas por este ambiente de linguagem.    */
/*                                                                   */
/* Retorna                                                           */
/*    Successo....... 0                                              */
/*    Fracasso....... -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 de 14). Gabartio de Ambiente de Linguagem

```
/*--------------------------------------------------------------------*/
/*                                                                    */
/* Função                                                             */
/*    dtw_initialize                                                  */
/*                                                                    */
/* Objetivo                                                           */
/*                                                                    */
/* Formato                                                            */
/*    int dtw_initialize(dtw_lei_t *le_interface)                     */
/*                                                                    */
/* Parâmetros                                                         */
/*    le_interface     Um ponteiro para uma estrutura que contém      */
/*                     os seguintes campos:                           */
/*                                                                    */
/*      function_name                                                 */
/*      flags                                                         */
/*      exec_statement                                                */
/*      parm_data_array                                               */
/*      default_error_message                                         */
/*      le_opaque_data                                                */
/*      row                                                           */
/*                                                                    */
/* Retorna                                                            */
/*    Successo....... 0                                               */
/*    Fracasso....... 0                                               */
/*--------------------------------------------------------------------*/
int dtw_initialize(dtw_lei_t *le_interface)
{
    return rc;
}
```

Figura 2 (Parte 3 de 14). Gabartio de Ambiente de Linguagem

```
/*--------------------------------------------------------------------*/
/*                                                                    */
/* Função                                                             */
/*    dtw_execute                                                     */
/*                                                                    */
/* Objetivo                                                           */
/*                                                                    */
/* Formato                                                            */
/*    int dtw_execute(dtw_lei_t *le_interface)                        */
/*                                                                    */
/* Parâmetros                                                         */
/*    le_interface     Um ponteiro para uma estrutura que contém      */
/*                     os seguintes campos:                           */
/*                                                                    */
/*      function_name                                                 */
/*      flags                                                         */
/*      exec_statement                                                */
/*      parm_data_array                                               */
/*      default_error_message                                         */
/*      le_opaque_data                                                */
/*      row                                                           */
/*                                                                    */
/* Retorna                                                            */
/*    Successo....... 0                                               */
/*    Fracasso....... 0                                               */
/*--------------------------------------------------------------------*/
int dtw_execute(dtw_lei_t *le_interface)
{
    /*----------------------------------------------------------------*/
    /* Determinar se foi especificada a instrução %exec.              */
    /*----------------------------------------------------------------*/
    if (le_interface->flags & DTW_STMT_EXEC) {
        /*------------------------------------------------------------*/
        /* Analisar a instrução%exec                                  */
        /*------------------------------------------------------------*/
        rc = processExecStmt(le_interface->exec_statement);
    if (rc)
          {
          }
      }
    else {
        /*------------------------------------------------------------*/
        /* Analisar os dados dispostos em seqüência                   */
        /*------------------------------------------------------------*/
        rc = processInlineData(le_interface->exec_statement);
    if (rc)
          {
          }
      }
```

Figura 2 (Parte 4 de 14). Gabartio de Ambiente de Linguagem

```
    /*-------------------------------------------------------------------*/
    /* Analisar os parâmetros de entrada                                 */
    /*-------------------------------------------------------------------*/
    rc = processInputParms(le_interface->parm_data_array);
    if (rc)
      {
      }
    /*-------------------------------------------------------------------*/
    /* Processar a solicitação                                           */
    /*-------------------------------------------------------------------*/
    rc = processRequest();
    if (rc)
      {
      }
    /*-------------------------------------------------------------------*/
    /* Processar os dados de saída                                       */
    /*-------------------------------------------------------------------*/
    rc = processOutputParms(le_interface->parm_data_array);
    if (rc)
      {
      }
    /*-------------------------------------------------------------------*/
    /* Processar o código de retorno e mensagem de erro padrão           */
    /*-------------------------------------------------------------------*/
    if (rc)
      {
        setErrorMessage(rc, &(le_interface->default_error_message));
      }
    /*-------------------------------------------------------------------*/
    /* Fazer limpeza e sair do programa                                  */
    /*-------------------------------------------------------------------*/
    return rc;
}
```

Figura 2 (Parte 5 de 14). Gabartio de Ambiente de Linguagem

```
/*--------------------------------------------------------------------*/
/*                                                                    */
/* Função                                                             */
/*    dtw_getNextRow                                                  */
/*                                                                    */
/* Objetivo                                                           */
/*                                                                    */
/* Formato                                                            */
/*    int dtw_getNextRow(dtw_lei_t *le_interface)                     */
/*                                                                    */
/* Parâmetros                                                         */
/*    le_interface     Um ponteiro para uma estrutura que contém      */
/*                     os seguintes campos:                           */
/*                                                                    */
/*      function_name                                                 */
/*      flags                                                         */
/*      exec_statement                                                */
/*      parm_data_array                                               */
/*      default_error_message                                         */
/*      le_opaque_data                                                */
/*      row                                                           */
/*                                                                    */
/* Retorna                                                            */
/*    Successo....... 0                                               */
/*    Fracasso....... 0                                               */
/*--------------------------------------------------------------------*/
int dtw_getNextRow(dtw_lei_t *le_interface)
{
    return rc;
}
```

Figura 2 (Parte 6 de 14). Gabartio de Ambiente de Linguagem

```
/*--------------------------------------------------------------------*/
/*                                                                    */
/* Função                                                             */
/*    dtw_cleanup                                                     */
/*                                                                    */
/* Objetivo                                                           */
/*                                                                    */
/* Formato                                                            */
/*    int dtw_cleanup(dtw_lei_t *le_interface)                        */
/*                                                                    */
/* Parâmetros                                                         */
/*    le_interface     Um ponteiro para uma estrutura que contém      */
/*                     os seguintes campos:                           */
/*                                                                    */
/*      function_name                                                 */
/*      flags                                                         */
/*      exec_statement                                                */
/*      parm_data_array                                               */
/*      default_error_message                                         */
/*      le_opaque_data                                                */
/*      row                                                           */
/*                                                                    */
/* Retorna                                                            */
/*    Successo....... 0                                               */
/*    Fracasso....... 0                                               */
/*--------------------------------------------------------------------*/
int dtw_cleanup(dtw_lei_t *le_interface)
{
    /*----------------------------------------------------------------*/
    /* Determinar se este é um fim do programa normal ou anormal.     */
    /*----------------------------------------------------------------*/
    if (le_interface->flags & DTW_END_ABNORMAL) {
        /*------------------------------------------------------------*/
        /* Fazer a limpeza do fim anormal.                            */
        /*------------------------------------------------------------*/
      }
    else {
        /*------------------------------------------------------------*/
        /* Fazer a limpeza do fim normal.                             */
        /*------------------------------------------------------------*/
      }

    return rc;
}
```

Figura 2 (Parte 7 de 14). Gabartio de Ambiente de Linguagem

```
/*--------------------------------------------------------------------*/
/*                                                                    */
/* Função                                                             */
/*    processInputParms                                               */
/*                                                                    */
/* Objetivo                                                           */
/*                                                                    */
/* Formato                                                            */
/*    unsigned long processInputParms(dtw_parm_data_t *parm__data)    */
/*                                                                    */
/* Parâmetros                                                         */
/*    dtw_parm_data_t *parm_data                                      */
/*                                                                    */
/* Retorna                                                            */
/*    Successo ..... 0                                                */
/*    Fracasso .....                                                  */
/*                                                                    */
/*--------------------------------------------------------------------*/
unsigned long processInputParms(dtw_parm_data_t *parm_data)
{
    /*----------------------------------------------------------------*/
    /* Fazer um loop através de todas as variáveis na matriz de dados */
    /* de parâmetros. A matriz é terminada por uma entrada NULL, que  */
    /* significa que o campo pram_name é definido como NULL, o campo  */
    /* parm_value é definido como NULL e o campo parm_descriptor é    */
    /* definido como 0. Entretanto, a única verificação válida para o */
    /* final da matriz de dados de parâmetro é verificar se           */
    /* parm_descriptor == 0, já que quando uma cadeia literal é       */
    /* passada para dentro, o campo parm_field fica NULL, e o campo   */
    /* parm_value fica NULL quando passa-se uma variável não-declarada*/
    /*----------------------------------------------------------------*/
    for (; parm_data->parm_descriptor != 0; ++parm_data) {
```

Figura 2 (Parte 8 de 14). Gabartio de Ambiente de Linguagem

```
        /*-------------------------------------------------------------*/
        /* Determinar a utilização de cada parâmetro de entrada.       */
        /*-------------------------------------------------------------*/
        switch(parm_data->parm_descriptor & DTW_USAGE) {

            case(DTW_IN):
                /*-----------------------------------------------------*/
                /* Determinar o tipo de cada parâmetro de entrada.     */
                /*-----------------------------------------------------*/
                switch (parm_data->parm_descriptor & DTW_TYPE) {
                case DTW_STRING:
            break;
                case DTW_TABLE:
            break;
                    default:
                        /*-------------------------------------------*/
                    /* Erro interno - tipo de dados desconhecido      */
                        /*-------------------------------------------*/
            break;
                  }
            break;

            case(DTW_OUT):
            break;

            case(DTW_INOUT):
            break;

                    default:
                /*-----------------------------------------------------*/
                /* Erro interno - utilização desconhecida              */
                /*-----------------------------------------------------*/
            break;
           }
      }
    return rc;
}
```

Figura 2 (Parte 9 de 14). Gabartio de Ambiente de Linguagem

```
/*--------------------------------------------------------------------*/
/*                                                                    */
/* Função                                                             */
/*    processOutputParms()                                            */
/*                                                                    */
/* Objetivo                                                           */
/*                                                                    */
/* Formato                                                            */
/*    unsigned long processOutputParms(dtw_parm_data_t *parm_data)    */
/*                                                                    */
/* Parâmetros                                                         */
/*    dtw_parm_data_t *parm_data                                      */
/*                                                                    */
/* Retorna                                                            */
/*    Successo .......  0                                             */
/*    Fracasso ....... -1                                             */
/*                                                                    */
/*--------------------------------------------------------------------*/
unsigned long processOutputParms(dtw_parm_data_t *parm_data) {
    /*----------------------------------------------------------------*/
    /* Pegar dados de saída de alguma maneira específica para o       */
    /* ambiente de linguage,. Isto depende totalmente de com que o    */
    /* ambiente está fazendo interface, e como o AL opta por fazer    */
    /* esta interface.                                                */
    /*----------------------------------------------------------------*/
```

Figura 2 (Parte 10 de 14). Gabartio de Ambiente de Linguagem

```
/    /*-------------------------------------------------------------*/
    /* Fazer um loop através de todos os parâmetros da matriz de   */
    /* dados de parâmetro, procurando parâmetros de saída.         */
    /*-------------------------------------------------------------*/
    for (; parm_data->parm_descriptor != 0; ++parm_data) {

        /*---------------------------------------------------------*/
        /* Determinar a utilização de cada parâmetro.              */
        /*---------------------------------------------------------*/
        if (pd_i->parm_descriptor & DTW_OUT) {
            /*-----------------------------------------------------*/
                /* Determinar o tipo de cada parâmetro de entrada. */
            /*-----------------------------------------------------*/
            switch (pd_i->parm_descriptor & DTW_TYPE) {
                case DTW_STRING:
                    /*-------------------------------------------*/
                    /* Dar um novo valor a um parâmetro de cadeia. Se */
                    /* o valor do parâmetro não for atualmente NULL,  */
                    /* a memória deve ser livrada usando uma função   */
                    /* utilitária de interface de AL se ela tiver sido*/
                    /* alocada pelo 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:
                    /*-------------------------------------------*/
                    /* Altere o tamanho de um parâmetro de tabela. Use*/
                    /* as funções utilitárias de interface de AL para */
                    /* alterar o objeto tabela.                  */
                    /*-------------------------------------------*/
                    /*-------------------------------------------*/
                    /* Primeiro pegue o ponteiro para o objeto tabela.*/
                    /*-------------------------------------------*/
                    void *myTable = (void *) parm_data->parm_value;
```

Figura 2 (Parte 11 de 14). Gabartio de Ambiente de Linguagem

```
/*-----------------------------------------------*/
/* Depois pegue o tamanho atual da tabela.       */
/*-----------------------------------------------*/
cols = dtw_table_Cols(myTable);
rows = dtw_table_Rows(myTable);
/*-----------------------------------------------*/
/* Agora defina o novo tamanho (assuma que os    */
/* novos valores de tamanho são válidos).        */
/*-----------------------------------------------*/

/*-----------------------------------------------*/
/* Defina primeiro as colunas.                   */
/*-----------------------------------------------*/
if (cols > newColValue)
  {
    dtw_table_DeleteCol(myTable,
                        newColValue + 1,
                        cols - newColValue);
  }
else if (cols < new_col_value)
  {
    dtw_table_InsertCol(myTable,
                        cols,
                        newColValue - cols);
  }

/*-----------------------------------------------*/
/* Agora defina as linhas.                       */
/*-----------------------------------------------*/
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 de 14). Gabartio de Ambiente de Linguagem

```
                        /*-----------------------------------------------*/
                        /* Agora pegue o valor da última linha/coluna.   */
                        /*-----------------------------------------------*/
                        dtw_table_GetV(myTable,
                                       &myValue;,
                                       newRowValue,
                                       newColValue);

                        /*-----------------------------------------------*/
                        /* Elimine o valor da última linha/coluna.       */
                        /*-----------------------------------------------*/
                        dtw_table_SetV(myTable,
                                       NULL,
                                       newRowValue,
                                       newColValue);

                        /*-----------------------------------------------*/
                        /* Defina o valor da última linha/coluna.        */
                        /*-----------------------------------------------*/
                        dtw_table_SetV(myTable,
                                       dtw_strdup(myNewValue),
                                       newRowValue,
                                       newColValue);

                    break;
                        default:
                        /*-----------------------------------------------*/
                        /* Erro interno - tipo de dados desconhecido     */
                        /*-----------------------------------------------*/
                    break;
                      }
                  }
              }

        return 0;
    }
```

Figura 2 (Parte 13 de 14). Gabartio de Ambiente de Linguagem

```
/*--------------------------------------------------------------------*/
/*                                                                    */
/* Função                                                             */
/*    setErrorMessage()                                               */
/*                                                                    */
/* Objetivo                                                           */
/*                                                                    */
/* Formato                                                            */
/*    unsigned long setErrorMessage(int returnCode,                   */
/*                                  char **defaultErrorMessage)       */
/*                                                                    */
/* Parâmetros                                                         */
/*    int    returnCode                                               */
/*    char **defaultErrorMessage                                      */
/*                                                                    */
/* Retorna                                                            */
/*    Successo .......  0                                             */
/*    Fracasso ....... -1                                             */
/*                                                                    */
/*--------------------------------------------------------------------*/
unsigned long setErrorMessage(int returnCode,
                              char **defaultErrorMessage)
{

    /*----------------------------------------------------------------*/
    /* Defina a mensagem de erro padrão baseada no código de retorno. */
    /*----------------------------------------------------------------*/
    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 de 14). Gabartio de Ambiente de Linguagem

Apêndice B. Avisos

Estas informações foram desenvolvidas para produtos e serviços oferecidos nos EUA. A IBM pode não oferecer os produtos, serviços ou recursos discutidos neste documento em outros países. Consulte seu representante IBM local para mais informações sobre os produtos e serviços atualmente disponíveis em sua área. Qualquer referência a um produto, programa ou serviço IBM não pretende afirmar ou implicar que apenas aquele produto, programa ou serviço IBM possa ser utilizado. Qualquer produto, programa ou serviço equivalente que não infrinja um direito a propriedade intelectual da IBM pode ser usado no seu lugar. Entretanto, é de responsabilidade do usuário avaliar e verificar a operação de um produto, programa ou serviço não-IBM.

A IBM pode ter patentes ou solicitações de patentes pendentes relativas a assuntos tratados neste documento. O fornecimento deste documento não lhe concede nenhuma licença sobre estas patentes. Você pode enviar consultas sobre licenças, por escrito, ao:

Gerente de Relações Comerciais e Industriais
da IMB do Brasil
Av. Pasteur, 138/146
Botafogo - Rio de Janeiro - RJ
Cep: 22290-240
BRASIL.

O seguinte parágrafo não se aplica ao Reino Unido ou a qualquer outro país onde tais providências sejam inconsistentes com as leis locais: A INTERNATIONAL BUSINESS MACHINES CORPORATION FORNECE ESTA PUBLICAÇÃO “COMO ESTÁ” SEM GARANTIA DE NENHUM TIPO, EXPRESSA OU IMPLÍCITA, INCLUSIVE, MAS SEM SE LIMITAR A, AS GARANTIAS IMPLICADAS DE NÃO INFRINGIMENTO, COMERCIABILIDADE OU ADEQUAÇÃO A UM FIM EM PARTICULAR. Alguns estados não permitem um aviso de isenção de garantias expressas ou implícitas em certas transações, e portanto esta declaração pode não se aplicar a você.

Estas informações poderiam conter imprecisões técnicas ou erros tipográficos. As alterações são periodicamente feitas às informações aqui contidas; estas alterações serão incorporadas a novas edições da publicação. A IBM pode fazer melhorias e/ou alterações no(s) produto(s) e/ou programa(s) descritos nesta publicação a qualquer momento sem aviso.

Os licenciados deste programa que desejam obter informações sobre este, com o propósito de possibilitar: (i) a troca de informações entre programas criados independentemente e outros programas (incluindo este), e (ii) o uso mútuo das informações que foram trocadas, deveriam entrar em contato com:

Centro de Atendimento IBM Brasil
Av. Pasteur, 138-146 - Botafogo
CEP: 22290-240
RJ - Brasil.

Tais informações podem estar disponíveis, sujeitas aos termos e condições adequados, incluindo em alguns casos o pagamento de uma taxa.

O programa licenciado descrito nestas informações e todo o material licenciado disponível para ele são fornecidos sob os termos do Contrato com o Cliente IBM ou qualquer outro contrato equivalente entre nós.

As informações a respeito de produtos que não os da IBM foram obtidas dos fornecedores destes produto, seu anúncios publicados ou outras fontes publicamente disponíveis. A IBM não testou estes produtos e não pode confirmar a precisão de seu desempenho, compatibilidade ou quaisquer outras afirmações relacionadas a produtos não-IBM. Perguntas sobre as capacidades de produtos não-IBM devem ser endereçadas aos fornecedores dos produtos.

LICENÇA DE DIREITOS AUTORAIS:

Estas informações contêm programas aplicativos de exemplo em linguagem-fonte, que ilustram técnicas de programação em várias plataformas operacionais. Você pode copiar, modificar, e distribuir estes programas de exemplo de qualquer forma sem pagar à IBM, para fins de desenvolvimento, uso, marketing ou distribuição de programas aplicativos em conformação com a interface de programação de aplicativos para a plataforma operacional para que os programas de exemplo forma escritos. Estes exemplos não foram completamente testados sob todas as condições. A IBM, portanto, não pode garantir ou implicar confiabilidade, manutenção ou função destes programas. Você pode copiar, modificar, e distribuir estes programas de exemplo de qualquer forma sem pagar à IBM, para fins de desenvolvimento, uso, marketing ou distribuição de programas aplicativos em conformação com a interfaces de programação de aplicativos.

Marcas

Os termos a seguir são marcas da IBM Corporation nos Estados Unidos, em outros países ou em ambos:

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

Os termos a seguir são marcas de outras companhias, como segue:

Java e HotJava são marcas da Sun Microsystems, Inc.

Microsoft, Windows, Windows NT®, e o logotipo do Windows 95 são marcas registradas da Microsoft Corporation.

UNIX é uma marca registrada nos Estados Unidos e em outros países, licenciada exclusivamente através da X/Open Company Limited

Os nomes de outros serviços, produtos e empresas podem ser de marcas ou de serviço de terceiros.

Glossário

ambiente de linguagem. Um módulo que fornece acesso de uma macro do Net.Data a uma fonte de dados externa como o DB2 ou uma linguagem de programação como Perl. Alguns ambientes de linguagem são fornecidos com o Net.Data como REXX, Perl e Oracle. Você também pode criar seus próprios ambientes de linguagem.

API. Application Programming Interface (Interface de Programação de Aplicativos).

applet. Um programa Java incluído em uma página HTML. Os applets trabalham com navegadores habilitados para Java, tais como Netscape, e são carregados quando a página HTML é carregada.

banco de dados. Uma coleção de tabelas, ou uma coleção de espaços de tabela e espaços de índice.

BLOB. Objeto grande binário.

cache. Um tipo de memória que contém dados recentemente acessados, feita para para acelerar acessos subseqüentes aos mesmos dados. A cache é comumente usada para guardar uma cópia local de dados freqüentemente usados que são acessíveis através da rede.

caching. Os processos de armazenamento de resultados utilizados freqüentemente em um servidor Web localmente para rápida recuperação, até que seja hora de atualizar as informações.

caminho. Uma rota de pesquisa usada para localizar arquivos.

CGI. Common Gateway Interface (Interface de Gateway Comum)

cliette. Um processo de longa execução que serve solicitações de um servidor Web. O Connection Manager agenda processos de cliettes para servirem a estas solicitações.

CLOB. Objeto grande de caractere.

Common Gateway Interface (Interface de Gateway Comum). Uma maneira padronizada para um servidor Web passar controle para um programa aplicativo e receber dados de volta.

Connection Manager (Gerenciador de Conexões). Um arquivo executável, `dtwcm`, no Net.Data, que é necessário para suportar o Live Connection.

cookie. Um pacote de informações enviado por um servidor HTTP para um navegador Web e então enviado de volta pelo navegador cada vez que ele acessa o servidor. Cookies podem conter qualquer informação arbitrária pela qual o servidor optar e são usados para manter o estado entre transações HTTP normalmente sem estado.*Free Online Dictionary of Computing*

DBMS. Sistema de gerenciamento de banco de dados.

firewall. Um computador com software que protege uma rede interna de acessos externos não-autorizados.

Gerenciador de Cache. O programa que gerencia uma cache para uma máquina. Ele pode gerenciar múltiplas caches.

HTML. Hypertext markup language (linguagem de marcação de hipertexto).

HTTP. Hypertext transfer protocol (protocolo de transferência de hipertexto).

hypertext markup language (linguagem de marcação de hipertexto). Uma linguagem de marcação para gravar documentos Web.

hypertext transfer protocol (protocolo de transferência de hipertexto). O protocolo de comunicação usado entre um servidor Web e o navegador.

ICAPI. Internet Connection API (API de Conexão à Internet).

ICS. Internet Connection Server (Servidor de Conexão à Internet).

ICSS. Internet Connection Secure Server (Servidor Seguro de Conexão à Internet).

interface de arquivo plano. Um conjunto de funções internas do Net.Data que lhe permitem ler e escrever dados de arquivos de texto corrido.

interface de programação de aplicativos (application programming interface - API). Uma interface funcional fornecida pelo sistema operacional ou por um programa licenciado e obtenível separadamente que permite que um programa aplicativo, escrito em uma linguagem de alto nível, use dados ou funções específicas do sistema operacional ou do programa licenciado. O Net.Data suporta as seguintes APIs de servidor Web proprietárias para melhorar o desempenho em processos CGI: ICAPI, GWAPI, ISAPI, e NSAPI.

Internet. Uma rede de computadores TCP/IP pública internacional.

Internet Connection Secure Server. Servidor seguro Web da IBM.

Internet Connection Server. Servidor inseguro Web da IBM.

Intranet. Uma rede TCP/IP dentro de um firewall de companhia.

ISAPI. API do Servidor da Internet da Microsoft.

Java. Uma linguagem de programação orientada por objeto independente de sistema operacional, especialmente útil para aplicações da Internet.

Live Connection. Uma configuração do Net.Data que funciona com o Connection Manager e com a API do servidor Web. O Live Connection permite que conexões a bancos de dados sejam reutilizadas.

LOB. Objeto grande.

middleware. Software que intermedeia entre um programa aplicativo e uma rede. Ele gerencia a interação entre aplicativos díspares através de sistemas operacionais de computação heterogêneos. *Free Online Dictionary of Computing*

NSAPI. Netscape API.

nulo. Um valor especial que indica a ausência de informações.

Perl. Uma linguagem de programação interpretada.

porta. Um número de 16 bits usado para comunicação entre TCP/IP e um protocolo ou aplicativo de alto nível.

Servidor Web. Um computador executando software de servidor http, tal como Internet Connection.

sistema de gerenciamento de banco de dados (database management system - DBMS). Um sistema de software que controle a criação, organização, e modificação de um banco de dados e acesso os dados armazenados dentro dele.

TCP/IP. Transmission Control Protocol / Internet Protocol (Protocolo de Controle de Transmissão / Protocolo da Internet)

tipo de dados. Um atributo de colunas e literais.

Transmission Control Protocol / Internet Protocol (Protocolo de Controle de Transmissão / Protocolo da Internet). Um conjunto de protocolos de comunicação que suporta funções de conectividade ponto a ponto tanto para redes de área estendida quanto locais.

uniform resource locator (localizador de recursos uniforme). Um endereço que nomeia um servidor HTTP e opcionalmente um diretório e nome de arquivo, por exemplo: `http://www.software.ibm.com/data/net.data/index.html`.

URL. Uniform resource locator (Localizador uniforme de recursos).

Índice Remissivo

Numéricos

0System
  ambiente de linguagem 30
  passagem de variáveis 31

A

alocação de memória dinâmica 43
ambientes de linguagem
  Aplicativos Java 18
  Applet Java 12
  configuração 46
  criando 35
  estruturas 40
    Veja também ?
  funções de interface 43
  funções utilitárias 49
    Veja também ?
  gabarito da interface 65
  IMS Web 10
  inicialização 43
  instruções, execução 43
  interface de arquivo plano 7
  introdução 49
  limpeza após processamento 44, 45
  ODBC 22
  Oracle 23
  Perl 26
  Registro Web 32
  REXX 27
  sistema 30
  SQL 28
  sumário 3
  Sybase 28
Aplicativos Java
  Ambiente de Linguagem 18
  chamada para execução 22
  configuração especial 21
  criação de cliettes 20
  criação de funções 20
apontamentos para a memória 53
Applet Java
  ambiente de linguagem 12
  chamada para execução 12
  classes 17
  criando 12
  exemplo 16
  geração de tags 12
Avisos 79

B

Bloco FUNCTION
  execução de instruções 43, 44
  nome 40

C

chamada para execução de applets 12
colunas
  determinação do total na tabela 55
  eliminação 56
  especificação do número em uma tabela 61
  inserção 58
condições anormais
  mensagens de erro 41
  sinalizador dtw_lei 41, 46
condições de erro 39
configuração de ambientes 46
criação de tabelas 59

D

DB2, ambiente de linguagem SQL 28
dtw_ structures 40
  Veja também ?
dtw_ utilities 49
  Veja também funções utilitárias
DTW_APPLET 12
DTW_FFI 7
DTW_IMS 10
DTW_JAVAPPS 18
DTW_LE_CONTINUE 45
dtw_lei
  campos
    default_error_messages 41
    exec_statement 41
    le_opaque_data 41
    linha 42
    nome de função 40
    parm_data_array 41
    sinalizadores 40
  estrutura 40
DTW_ODBC 22
DTW_ORA 23
dtw_parm_data
  campos
    parm_descriptor 42
    parm_name 43
    parm_value 43
  estrutura 42
DTW_PERL 26

DTW_REXX 27
DTW_SQL 28
DTW_SYB 28
DTW_SYSTEM 30
DTW_WEBREG 32

E

erros graves, sinalizador dtw_lei 41, 46
estruturas parm_data_array, designação de nomes 41
estruturas, ambiente de linguagem
  dtw_lei 40
  dtw_parm_data 42
execução de instruções de ambiente de linguagem 43, 44

F

funções de interface
  ambiente de linguagem, descrição 43
  dtw_cleanup() 45
  dtw_execute() 44
  dtw_getNextRow() 45
  dtw_initialize() 44
  ordem de processamento 43
funções de interface dtw_ 43
funções utilitárias
  ambiente de linguagem 49
  dtw_free() 52
  dtw_getvar() 52
  dtw_malloc() 53
  dtw_row_SetCols() 53
  dtw_row_SetV() 53
  dtw_strdup() 54
  dtw_table_AppendRow() 54
  dtw_table_Cols() 55
  dtw_table_Delete() 55
  dtw_table_DeleteCol() 56
  dtw_table_DeleteRow() 56
  dtw_table_GetN() 57
  dtw_table_GetV() 57
  dtw_table_InsertCol() 58
  dtw_table_InsertRow() 58
  dtw_table_MaxRows() 59
  dtw_table_New() 59
  dtw_table_QueryColnoNj() 60
  dtw_table_Rows() 60
  dtw_table_SetCols() 61
  dtw_table_SetN() 61
  dtw_table_SetV() 62
  gerenciamento de memória 49
  manipulação de linhas 51
  manipulação de tabelas 50
  variável de configuração 50
funções utilitárias de gerenciamento de memória 49
funções utilitárias de manipulação de linhas 51

G

gabarito, ambiente de linguagem 65
glossário 81

I

IMS Web
  ambiente de linguagem 10
  Ferramenta de estúdio 11
inicialização de tarefas, ambientes de linguagem 43, 44
instruções ENVIRONMENT
  exemplos 48
  para novos ambientes de linguagem 46
  sintaxe 46
instruções exec, sinalizador dtw_lei 41
interface de arquivo plano
  ambiente de linguagem 7
  considerações sobre segurança 7
  funções internas 8

L

limpeza
  após processamento 44, 45
  sinalizador dtw_lei 41, 46
  sinalizador para finalização anormal 41, 46
linhas
  anexação 54
  designação de largura 53
  eliminação 56
  função de interface dtw_getNextRow() 42
  inserção 58
  recuperação do número atual de 60
  retorno 42, 44, 45
  retorno do máximo número permitido 59

M

máximo número de colunas 59
memória
  alocação 53, 54, 61
  desocupação 41, 43, 52
  sinalizador dtw_lei 41
mensagens de condições de erro 41

O

ODBC
  ambiente de linguagem 22
  instruções SQL no arquivo de macro 22
Oracle
  acesso 23
  ambiente de linguagem 23

Oracle *(continuação)*
configuração especial 23
procedimento armazenado 23

P

parâmetros
especificação 42
nomeação 43
parm_name 43
passando 41, 42
passando
parâmetros 41
variáveis 41
Perl
ambiente de linguagem 26
Variáveis Net.Data nos scripts 26
pilha de tempo de execução do Net.Data 43
processamento linha-por-vez
dtw_getNextRow() 44, 45
DTW_LE_CONTINUE 41
sinalizador dtw_lei 41

R

Registro Web
ambiente de linguagem 32
descrição 32
REXX
ambiente de linguagem 27
substituição de variável 27

S

SQL
ambiente de linguagem 28
instruções suportadas 28
utilização do DataJoiner 28
Sybase
acesso 28
ambiente de linguagem 28
configuração especial 28
objetos grandes 28

T

tabelas
anexação de linhas 54
criação de novas 59
eliminação 55
funções utilitárias de manipulação de 50
títulos de coluna
alocação de memória 53, 61
designação de nomes 61
eliminação 55, 57, 61
recuperação 57
retorno do número de coluna 60

V

valores de tabela
designação 53, 62
eliminação 55, 57, 62
recuperação 57
variáveis
desocupação 45
passando 41
variáveis de configuração
funções utilitárias para gerenciar 50
recuperação de valores de variáveis 52

Comentários do Leitor

Manual de Ambiente de Linguagens do Net.Data.

Neste formulário, faça-nos saber sua opinião sobre este manual. Utilize-o se encontrar algum erro, ou se quiser externar qualquer opinião a respeito (tal como organização, assunto, aparência ...) ou fazer sugestões para melhorá-lo.

Para pedir publicações extras, fazer perguntas ou tecer comentários sobre as funções de produtos ou sistemas da IBM, fale com o seu representante IBM.

Quando você envia seus comentários, concede direitos, não exclusivos, à IBM para usá-los ou distribuí-los da maneira que achar conveniente, sem que isso implique em qualquer compromisso ou obrigação para com você.

Não se esqueça de preencher seu nome e seu endereço abaixo, se desejar resposta.

Nome

Endereço

Companhia ou Empresa

Telefone

Comentários do Leitor

Corte ou dobre ao longo da linha

Dobre e cole com fita **Não grampeie** Dobre e cole com fita

COLE SELO POSTAL AQUI

Centro Industrial IBM Brasil
Centro de Traduções
Caixa Postal 71
13001-970 Campinas, SP
BRASIL

Dobre e cole com fita **Não grampeie** Dobre e cole com fita

Corte ou dobre ao longo da linha