

IBM[®] Net.Data
para OS/2[®], Windows NT[®] y UNIX[®]



Guía de administración y programación

Versión 7

IBM[®] Net.Data
para OS/2[®], Windows NT[®] y UNIX[®]



Guía de administración y programación

Versión 7

Antes de utilizar esta información y el producto al que da soporte, asegúrese de leer la información general incluida en el “Avisos” en la página 277.

Este documento contiene información sobre productos patentados de IBM. Se proporciona de acuerdo con un contrato de licencia y está protegido por la ley de la propiedad intelectual. La presente publicación no incluye garantías del producto y las declaraciones que contiene no deben interpretarse como tales.

Puede solicitar publicaciones a través del representante de IBM o sucursal de IBM de su localidad, o bien llamando a los números de teléfono 1-800-879-2755, en los Estados Unidos, o 1-800-IBM-4YOU, en Canadá.

Cuando envía información a IBM, otorga a IBM un derecho no exclusivo para utilizar o distribuir dicha información en la forma en que IBM considere adecuada, sin contraer por ello ninguna obligación con el remitente.

© Copyright International Business Machines Corporation 1997, 2000. Reservados todos los derechos.

Contenido

Prefacio	vii
Acerca de Net.Data	vii
Novedades de la Versión 7.	viii
Acerca de este manual.	ix
Quién debe leer este manual	ix
Acerca de los ejemplos de este manual	ix

Capítulo 1. Introducción	1
¿Qué es Net.Data?	1
¿Por qué utilizar Net.Data?	2

Capítulo 2. Configuración de Net.Data	5
Acerca del archivo de inicialización de Net.Data	7
Acerca de los archivos de configuración de Net.Data para los componentes opcionales	7
Archivo de configuración de Live Connection.	8
Archivo de configuración de Cache Manager	8
Secciones comunes de los archivos de macros, de control y de inicialización de Net.Data	9
Personalización del archivo de inicialización de Net.Data	12
Sentencias de variables de configuración	13
Sentencias de configuración de vía de acceso	22
Sentencias de configuración de entorno	28
Configuración de los entornos de lenguaje	31
Configuración del entorno de lenguaje IMS Web.	31
Configuración del entorno de lenguaje Java	32
Configuración del entorno de lenguaje Oracle	33
Configuración de Live Connection	36
Configuración del servidor Web para utilizarlo con CGI	42
Configuración de Net.Data para FastCGI	42
Configuración de Net.Data para utilizarlo con servlets Java y beans Java	46
Configuración de Net.Data para utilizarlo con las API del servidor Web.	46
Configuración de Net.Data con la herramienta de administración de Net.Data.	50

Antes de empezar	50
Inicio de la herramienta de administración	51
Configuración de las sentencias de vía de acceso	51
Configuración de puertos	53
Configuración de cliettes.	54
Configuración de entornos de lenguaje	59
Definición de variables de configuración	63
Cómo otorgar derechos de acceso a los archivos a los que accede Net.Data	65

Capítulo 3. Cómo mantener seguros sus activos	67
Utilización de cortafuegos	67
Cifrado de los datos en la red	70
Utilización de la autenticación	70
Utilización de la autorización	71
Utilización de los mecanismos de Net.Data	71
Variables de configuración de Net.Data	71
Técnicas de desarrollo de macros	73

Capítulo 4. Invocación de Net.Data	79
Tipos de peticiones de invocación.	79
Invocación de Net.Data con una macro (Petición de macro)	81
Invocación de Net.Data sin ninguna macro (petición directa)	86
Invocación de Net.Data mediante las API de servidor Web.	92
Invocación de Net.Data con servlets Java y JavaBeans	95
Servlets de Net.Data	95
JavaBeans de Net.Data	102

Capítulo 5. Desarrollo de macros de Net.Data	107
Anatomía de una macro de Net.Data	108
Bloque DEFINE	110
Bloque FUNCTION	111
Bloques HTML.	112
Bloques XML	114
Variables de macro de Net.Data	118
Ámbito del identificador	119
Definición de variables	120
Cómo hacer referencia a variables	122

Tipos de variables.	124	Restricciones del almacenamiento en antememoria de Net.Data	219
Funciones de Net.Data	133	Interfaces de almacenamiento en antememoria de Net.Data	219
Definición de funciones.	134	Planificación del Cache Manager.	220
Llamada de funciones	140	Configuración de las antememorias del Cache Manager y de Net.Data	222
Llamada de funciones incorporadas de Net.Data.	140	Inicio y detención del Cache Manager	230
Generación de marcación de documentos	145	Almacenamiento de páginas Web en antememoria	231
Bloques HTML y XML	145	Mandato CACHEADM	235
Bloques de informe	147	Anotación cronológica de antememoria	238
Lógica condicional y repetición en bucle en una macro	153	Establecimiento del nivel de anotación cronológica de errores	241
Lógica condicional: Bloques IF	154	Optimización de los entornos de lenguaje	241
Construcciones de repetición en bucle: Bloques WHILE	156	Entorno de lenguaje REXX.	241
Capítulo 6. Utilización de entornos de lenguaje	159	Entorno de lenguaje SQL	242
Visión general de los entornos de lenguaje proporcionados por Net.Data	161	Entornos de lenguaje System y Perl.	244
Llamada de un entorno de lenguaje.	162	Capítulo 8. Anotaciones cronológicas de Net.Data	245
Manejo de condiciones de error	162	Anotación cronológica de mensajes de error de Net.Data.	245
Seguridad	163	Planificación de la anotación cronológica de errores de Net.Data	246
Entornos de lenguaje de datos	163	Control del nivel de anotación cronológica de Net.Data.	247
Entornos de lenguaje de bases de datos relacionales	163	Tipos de mensajes de error de Net.Data no anotados cronológicamente	247
Entorno de lenguaje Flat File Interface	182	Tamaño y rotación del archivo de anotaciones cronológicas de errores de Net.Data.	247
Entorno de lenguaje Web Registry	184	Formato de la anotación cronológica de errores de Net.Data	248
Entornos de lenguaje de programación.	187	Anotación cronológica de mensajes de error y de cliette de Live Connection	248
Entorno de lenguaje Java Applet.	187	Planificación de la anotación cronológica de Live Connection	249
Entorno de lenguaje Java Application	195	Control del nivel de anotación cronológica de Live Connection	250
Entorno de lenguaje Perl	198	Tipos de mensajes de Live Connection no anotados cronológicamente	250
Entorno de lenguaje REXX.	201	Nombres de archivos de anotaciones cronológicas de Live Connection.	250
Entorno de lenguaje System	205	Tamaño y rotación del archivo de anotaciones cronológicas de Live Connection	251
Capítulo 7. Mejora del rendimiento	209	Formato de anotación cronológica de Live Connection	252
Utilización de las API del servidor Web	209	Apéndice A. Bibliografía	255
Utilización de FastCGI	210		
Gestión de conexiones	210		
Acerca de Live Connection.	211		
Ventajas de Live Connection	212		
¿Debo utilizar Live Connection?	212		
Inicio de Connection Manager	212		
Flujo de proceso de Net.Data y Live Connection	214		
Almacenamiento en antememoria de Net.Data.	214		
Acerca del almacenamiento de páginas Web en antememoria	215		
Acerca del almacenamiento en antememoria de Net.Data	216		

Biblioteca técnica de Net.Data.	255	Acerca del plug-in de NetObjects Fusion . . .	267
Apéndice B. Net.Data para AIX.	257	Instalación del plug-in de NetObjects Fusion	268
Carga de bibliotecas compartidas para los		Configuración del plug-in de Net.Data para	
entornos de lenguaje.	257	NetObjects Fusion	268
Mejora del rendimiento en el entorno REXX	258	Modificación de las propiedades del plug-in	269
Consideraciones acerca de NLS	258	Publicación de servlets con el plug-in NOF	272
 Apéndice C. Asistentes de Net.Data	 261	 Apéndice F. Macro de ejemplo de	
Antes de empezar.	262	Net.Data	273
Ejecución de los Asistentes.	262	 Avisos	 277
 Apéndice D. Creación de sentencias de		Marcas registradas	280
SQL con Net.Data SQL Assist	265	 Glosario	 283
Antes de empezar.	265	 Índice	 287
Ejecución de Net.Data SQL Assist	266	 Cómo ponerse en contacto con IBM . . .	 295
 Apéndice E. Utilización de los plug-in de		Información del producto	295
NetObjects Fusion (NOF) con servlets de			
Net.Data	267		

Prefacio

Gracias por seleccionar Net.Data[®], la herramienta de desarrollo de IBM[™] para crear páginas Web dinámicas. Con Net.Data, puede desarrollar de forma rápida páginas Web con contenido dinámico mediante la incorporación de datos de diversas fuentes y mediante la utilización de la potencia de los lenguajes de programación que ya conoce.

Acerca de Net.Data

Con el producto Net.Data de IBM, puede crear páginas Web dinámicas utilizando datos de sistemas de gestión de bases de datos (DBMS) relacionales y no relacionales, incluidas bases de datos habilitadas para ODBC, IMS y, y utilizando aplicaciones escritas en lenguajes de programación tales como Java, JavaScript, Perl, C, C++ y REXX.

Net.Data es un procesador de macros que se ejecuta como middleware en una máquina servidor Web. Puede escribir programas de aplicación Net.Data, llamados *macros*, que Net.Data interpreta para crear páginas Web dinámicas con contenido personalizado, basándose en la entrada del usuario, el estado actual de la base de datos, otras fuentes de datos, la lógica de negocio existente y otros factores que se diseñan en la macro.

Una petición de un navegador, por ejemplo Netscape Navigator o Internet Explorer, fluye en forma de URL (uniform resource locator - localizador uniforme de recursos) a un servidor Web que reenvía la petición a Net.Data para que la ejecute. Net.Data localiza y ejecuta la macro y crea una página Web, que personaliza basándose en las funciones que se han escrito. Estas funciones pueden:

- Encapsular la lógica de negocio dentro de scripts Perl, aplicaciones C y C++ o programas REXX
- Acceder a bases de datos tales como DB2
- Acceder a otras fuentes de datos como, por ejemplo, archivos planos.

Net.Data pasa esta página Web al servidor Web, que a su vez reenvía la página a través de la red para que se visualice en el navegador.

Net.Data puede utilizarse en entornos de servidor que estén configurados para utilizar interfaces tales como HTTP (HyperText Transfer Protocol) y CGI (Common Gateway Interface). HTTP es una interfaz estándar de la industria para la interacción entre un navegador y un servidor Web, y CGI es una interfaz estándar de la industria para la invocación, por parte del servidor

Web, de aplicaciones de pasarela como Net.Data. Estas interfaces permiten seleccionar el navegador o servidor Web favorito para utilizarlo con Net.Data. Net.Data también soporta diversas Interfaces de programación de aplicaciones (API) de servidor Web para mejorar el rendimiento. La familia de productos Net.Data proporciona posibilidades similares en los sistemas operativos OS/400, OS/390, Windows NT, AIX, OS/2, HP-UX, Sun Solaris, Linux y Dynix/PTX. Net.Data también soporta FastCGI y las principales Interfaces de programación de aplicaciones (API) de servidor Web en múltiples sistemas operativos.

Una herramienta gráfica de administración le ayuda a administrar los valores de configuración de Net.Data para los sistemas operativos AIX, Windows NT y OS/2. La herramienta de administración también le ayuda a especificar seguridad para las conexiones a las bases de datos que utilizan Live Connection.

Para ayudarle a acceder fácilmente a los datos de la base de datos, Net.Data proporciona diversas herramientas, que incluyen los plug-in NetObjects Fusion y los asistentes para el desarrollo basado en Java. Estas herramientas funcionan con las servlets Java de Net.Data en el entorno Java, permitiéndole crear aplicaciones que son portables entre sistemas operativos. Los plug-in NetObjects Fusion le permiten utilizar la herramienta de desarrollo Web NetObjects Fusion para crear aplicaciones sofisticadas con datos dinámicos de fuentes de datos relacionales. Los asistentes de Net.Data proporcionan una herramienta gráfica para guiarle en la creación de macros de Net.Data básicas.

Novedades de la Versión 7

Net.Data Versión 7 ofrece toda la funcionalidad de los releases anteriores de Net.Data y mucho más. Net.Data para OS/2, Windows NT y UNIX proporciona las características adicionales siguientes en la Versión 7:

- Utilice Net.Data para visualizar tanto XML como soporte el navegador; los datos se visualizan automáticamente como XML utilizando el nuevo Bloque de informes XML. Net.Data viene con unas hojas de estilo XSL de ejemplo (ndTable.xsl, ndObject.xsl, ndRecord.xsl). Las encontrará en el directorio raíz del servidor Web.
- Los navegadores de cliente pueden subir archivos al servidor.
- Mediante la utilización de procedimientos almacenados, Net.Data puede importar archivos de texto a DB2.
- Nuevas funciones incorporadas de Net.Data: DTW_REPLACE(), DTW_HEXTOCHAR(), DTW_CHARTOHEX() y DTWF_READFILE().
- Rendimiento mejorado en DB2 utilizando DTW_USE_DB2_PREPARE_CACHE, que aprovecha la antememoria de paquete y las sentencias preparadas de DB2.

- Mejor eliminación de espacio en blanco obsoleto con DTW_REMOVE_WS.
- Proporcionar un mensaje de error de identificación general con DTW_DEFAULT_ERROR_MESSAGE.

Acerca de este manual

Este manual describe conceptos de administración y programación para Net.Data, así como el modo de configurar Net.Data y sus componentes, planificar la seguridad y mejorar el rendimiento.

Basándose en sus conocimientos de los lenguajes de programación y de base de datos, aprenderá a utilizar el lenguaje de las macros de Net.Data o servlets Java para desarrollar macros. Aprenderá a utilizar los entornos de lenguajes proporcionados por Net.Data que acceden a bases de datos DB2 y las transacciones IMS utilizando IMS Web, así como a utilizar Java, REXX, Perl, y otros lenguajes de programación para acceder a los datos.

Este manual puede hacer referencia a productos o características que se han anunciado pero que aún no están disponibles.

Puede encontrar más información, incluyendo macros de Net.Data de ejemplo, programas de demostración y la copia más reciente de este manual, en el sitio World Wide Web siguiente:

<http://www.ibm.com/software/data/net.data>

Quién debe leer este manual

Este manual está destinado a las personas implicadas en la planificación y escritura de aplicaciones Net.Data. Para comprender los conceptos descritos en este manual, deberá estar familiarizado con el modo en que funciona un servidor Web, deberá saber interpretar sentencias de SQL simples y deberá conocer los códigos HTML, incluidos los códigos de formularios HTML.

En el manual *Net.Data Guía de consulta*, se describen el lenguaje de las macros, las variables y las funciones incorporadas de Net.Data, así como las diferencias entre sistemas operativos.

Acerca de los ejemplos de este manual

En este manual se utilizan ejemplos simples con el fin de ilustrar conceptos específicos, por lo que dichos ejemplos no muestran todos los modos en que se pueden usar las construcciones de Net.Data. Algunos ejemplos son fragmentos que, para funcionar, necesitan código adicional.

Capítulo 1. Introducción

La mayoría de las páginas Web de Internet son páginas Web estáticas; en otras palabras, páginas que no cambian a no ser que se editen. Para dar “vida” a los datos y las aplicaciones de la Web (por ejemplo, las estadísticas de ventas actuales), los desarrolladores de sitios Web escriben generalmente programas que se ejecutan como middleware en el servidor Web para crear dinámicamente páginas Web. La escritura de estos tipos de programas no es fácil.

Net.Data simplifica la escritura de aplicaciones Web interactivas mediante las *macros*.

Este capítulo describe Net.Data y las razones por las que puede que desee utilizarlo para las aplicaciones Web.

- “¿Qué es Net.Data?”
- “¿Por qué utilizar Net.Data?” en la página 2

¿Qué es Net.Data?

Mediante la utilización de las macros de Net.Data, puede ejecutar lógica de programación, acceder a variables y manipularlas, llamar a funciones y utilizar herramientas de generación de informes. Una macro es un archivo de texto que contiene construcciones de lenguaje de macro de Net.Data, códigos HTML, Javascript y sentencias de entorno de lenguaje, por ejemplo SQL y Perl. Net.Data procesa la macro para producir salida que se puede visualizar mediante un navegador Web. Las macros combinan la sencillez de HTML con la funcionalidad dinámica de los programas servidor Web, haciendo que sea fácil añadir datos vivos a las páginas Web estáticas. Los datos vivos pueden extraerse de bases de datos locales o remotas y de archivos planos o pueden generarlos aplicaciones y servicios de sistema.

La Figura 1 en la página 2 ilustra la relación entre Net.Data, el servidor Web y los entornos de lenguaje de programación y de datos soportados.

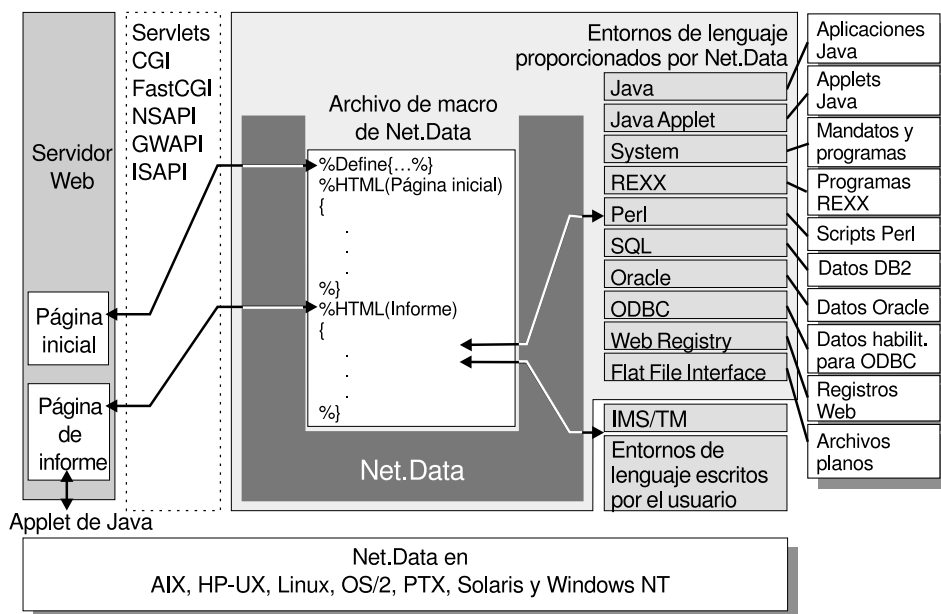


Figura 1. Relación entre Net.Data, el servidor Web y las fuentes de programas y datos soportadas

El servidor Web invoca Net.Data como una interfaz de programación de aplicaciones (API) CGI, FastCGI o de servidor Web llamando a Net.Data como una DLL o biblioteca compartida cuando recibe un URL que solicita servicios de Net.Data. El URL incluye información específica de Net.Data, que comprende la macro que se debe procesar o la sentencia de SQL o el programa que debe invocarse directamente. Cuando Net.Data termina de procesar la petición, envía la página Web resultante al servidor Web. El servidor la transmite al cliente Web, donde el navegador la visualiza.

¿Por qué utilizar Net.Data?

Net.Data es una buena elección para crear páginas Web dinámicas porque la utilización del lenguaje de macro es más simple que la escritura de aplicaciones de servidor Web propias y porque Net.Data le permite utilizar lenguajes que ya conoce, por ejemplo HTML, SQL, Perl, REXX y JavaScript. Net.Data también proporciona entornos de lenguaje que acceden a bases de datos DB2, ejecutan transacciones IMS utilizando IMS Web o utilizan REXX, Perl y otros lenguajes para las aplicaciones. Además, los cambios efectuados en una macro pueden verse instantáneamente en un navegador.

Net.Data complementa posibilidades de gestión de datos que ya existen en el sistema operativo habilitando para la Web los datos y la lógica de negocio relacionada. Más específicamente, Net.Data:

- Proporciona un lenguaje de macro simple, aunque potente, que permite el rápido desarrollo de aplicaciones de Internet e Intranet. El entorno de aplicaciones Web de Net.Data proporciona las características siguientes:
- Permite la separación de la lógica de generación de datos de la lógica de presentación de los mismos dentro de las aplicaciones Web. Net.Data no impone restricciones en el método con el que se presentan los datos (por ejemplo HTML o Javascript). Esta separación permite a los usuarios cambiar fácilmente la presentación de los datos utilizando las técnicas de presentación más recientes.
- Permite utilizar los conocimientos existentes y la lógica de negocio para generar páginas Web proporcionando la posibilidad de intercambiar información con programas escritos en C, C++, REXX, Java u otros lenguajes.
- Proporciona la posibilidad de desarrollar de forma rápida aplicaciones de Internet complejas, utilizando un lenguaje de macro simple .
- Proporciona acceso de alto rendimiento a los datos que están almacenados en DB2 y en cualquier base de datos remota habilitada para DRDA.
- Proporciona facilidad de migración de las macros entre todos los sistemas operativos soportados por la familia de productos Net.Data.

Lenguaje de macro interpretado

El lenguaje de macro de Net.Data es un lenguaje interpretado. Cuando se invoca Net.Data para procesar una macro, Net.Data interpreta directamente cada sentencia de lenguaje de una forma secuencial, empezando por el principio del archivo. Mediante el uso de esta propuesta, cualquier cambio que efectúe en una macro podrá verse inmediatamente cuando, a continuación, especifique el URL que ejecuta la macro. No es necesario realizar ninguna recompilación.

Peticiones directas

Las peticiones simples que requieren la ejecución de una sola sentencia de SQL, un procedimiento almacenado DB2, un programa REXX, un programa C o C++ o un script Perl, no necesitan que se cree una macro. Dichas peticiones pueden especificarse directamente en el URL que fluye del navegador al servidor Web.

Formato libre

El lenguaje de macro de Net.Data sólo tiene unas pocas normas para el formato de programación. Esta sencillez proporciona libertad y flexibilidad a los programadores. Una sola instrucción puede fragmentarse en muchas líneas o se pueden entrar varias instrucciones en una sola línea. Las instrucciones pueden empezar en cualquier columna. Se pueden saltar espacios o líneas enteras. Se pueden utilizar comentarios en cualquier lugar.

Variables sin tipo

Net.Data considera todos los datos como series de caracteres. Net.Data utiliza funciones incorporadas para realizar operaciones aritméticas en una serie que representa un número válido, incluyendo las que están en formatos exponenciales. Las variables de lenguaje de macro se describen detalladamente en el apartado “Variables de macro de Net.Data” en la página 118.

Funciones incorporadas

Net.Data proporciona funciones incorporadas que realizan diversas operaciones de proceso, búsqueda y comparación para texto y números. Otras funciones incorporadas proporcionan posibilidades de formato y cálculos aritméticos.

Manejo de errores

Cuando Net.Data detecta un error, se devuelven mensajes con explicaciones al cliente. Los mensajes de error pueden personalizarse antes de que se devuelvan a un usuario en un navegador. Consulte el manual *Net.Data Guía de consulta* para obtener más información.

Capítulo 2. Configuración de Net.Data

Puede instalar Net.Data para el sistema operativo correspondiente utilizando las instrucciones del archivo README que acompañaba al producto. La mayoría de los pasos de configuración se realizan durante la instalación; esto varía en función del sistema operativo.

Después de instalar Net.Data para el sistema operativo, deberá configurar Net.Data y modificar la configuración para el servidor Web. Las tareas de configuración incluyen:

- Personalización del archivo de inicialización (INI) de Net.Data
- Configuración de Net.Data para utilizarlo con FastCGI o una de las API de servidor Web soportadas (opcional)
- Personalización de los archivos de variables de entorno y de configuración de servidor Web
- Configuración del Cache Manager (opcional)
- Configuración de Live Connection (opcional)
- Preparación de los entornos de lenguaje Net.Data
- Especificación de los derechos de acceso

Para configurar Net.Data se utilizan las herramientas siguientes:

- Un editor de texto

Utilice un editor de texto para editar el archivo de inicialización y los archivos de configuración de Live Connection y Cache Manager en todos los sistemas operativos. Utilice también un editor de texto para actualizar los archivos de configuración de servidor Web. Es aconsejable hacer una copia de seguridad de los archivos antes de realizar cambios.

- La herramienta de administración de Net.Data

La herramienta de administración proporciona una interfaz gráfica para personalizar el archivo de inicialización y el archivo de configuración de Live Connection. Puede utilizar la herramienta de administración para configurar Net.Data en los sistemas operativos OS/2, Windows NT y AIX.

El método a utilizar dependerá de los componentes que sea necesario configurar y del sistema operativo en el que se esté ejecutando Net.Data, como se describe en la Tabla 1 en la página 6. Si empieza a utilizar un método determinado para una tarea de configuración, deberá continuar utilizando dicho método a fin de obtener los mejores resultados.

Tabla 1. Comparación de métodos de configuración con tareas y sistemas operativos. **A** - Puede configurarse con la herramienta de administración o manualmente. **M** - Sólo se puede configurar manualmente.

Tarea	Sistemas operativos:			
	AIX	NT	OS/2	HP SUN PTX
Configurar el archivo INI de Net.Data	A	A	A	M
Definir puertos de cliettes	A	A	A	M
Definir cliettes	A	A	A	M
Activar cifrado de contraseña de cliette	A	N/D	N/D	N/D
Activar anotación cronológica de errores	A	A	A	M
Configurar servidor Web para FastCGI, CGI y las API*	M	M	M	M
Definir puertos de Cache Manager	M	M	N/D	N/D
Configurar Cache Manager	M	M	N/D	N/D

***Consejo:** Muchos servidores Web tienen herramientas de administración que puede utilizar para configurar el servidor Web.

Este capítulo describe cómo configurar Net.Data y cómo modificar la configuración del servidor Web para utilizarlo con Net.Data. Adicionalmente, describe cómo configurar componentes opcionales.

- “Personalización del archivo de inicialización de Net.Data” en la página 12
- “Configuración de los entornos de lenguaje” en la página 31
- “Configuración de Live Connection” en la página 36
- “Configuración del servidor Web para utilizarlo con CGI” en la página 42
- “Configuración de Net.Data para FastCGI” en la página 42
- “Configuración de Net.Data para utilizarlo con servlets Java y beans Java” en la página 46
- “Configuración de Net.Data para utilizarlo con las API del servidor Web” en la página 46
- “Configuración de Net.Data con la herramienta de administración de Net.Data” en la página 50
- “Cómo otorgar derechos de acceso a los archivos a los que accede Net.Data” en la página 65

Acerca del archivo de inicialización de Net.Data

Net.Data utiliza el archivo de inicialización para establecer los valores de diversas variables de configuración y para configurar entornos de lenguaje y vías de acceso de búsqueda. Los valores de las variables de configuración controlan diversos aspectos del funcionamiento de Net.Data como, por ejemplo, los siguientes:

- La codificación de datos de tipo carácter como Unicode
- La determinación de si se han habilitado para MBCS las funciones de series y de palabras
- El nombre de la instancia de DB2 para acceder a datos de la base de datos
- El modo en que Net.Data se conecta y se comunica con los entornos de lenguaje, las bases de datos, la gestión de conexiones y el almacenamiento en antememoria
- La determinación de si se ha activado la anotación cronológica de errores

Las sentencias de entorno de lenguaje definen los entornos de lenguaje de Net.Data que están disponibles e identifican los valores de parámetros de entrada y salida especiales que fluyen hacia y desde los entornos de lenguaje. Los entornos de lenguaje permiten a Net.Data acceder a diferentes fuentes de datos, por ejemplo servicios del sistema y bases de datos DB2. Las sentencias de vía de acceso especifican las vías de acceso de directorio a los archivos que Net.Data utiliza, tales como macros, programas REXX y scripts Perl.

El archivo de inicialización de Net.Data, db2www.ini, está ubicado en el directorio de documentos del servidor Web. Si desea obtener más información, consulte el archivo README para el sistema operativo correspondiente.

Consejo respecto a la autorización: Asegúrese de que el ID de usuario bajo el que se ejecuta el servidor Web tiene autorización para leer este archivo. Consulte el apartado “Cómo otorgar derechos de acceso a los archivos a los que accede Net.Data” en la página 65 para obtener más información.

Acerca de los archivos de configuración de Net.Data para los componentes opcionales

Las secciones siguientes describen los archivos de configuración para los componentes opcionales de Net.Data.

“Archivo de configuración de Live Connection” en la página 8

“Archivo de configuración de Cache Manager” en la página 8

“Secciones comunes de los archivos de macros, de control y de inicialización de Net.Data” en la página 9

Archivo de configuración de Live Connection

Live Connection proporciona gestión de conexión en los sistemas operativos Windows NT, OS/2, AIX y Sun Solaris para mejorar el rendimiento mediante la eliminación de la actividad general del arranque. El archivo de configuración de Live Connection de Net.Data contiene información acerca de una o varias cliettes con nombre. Una cliette es un proceso de larga ejecución que mantiene una conexión a una base de datos o a una aplicación Java que perdura a lo largo de las invocaciones de macros de Net.Data de múltiples usuarios. Después de iniciarse, una cliette continúa existiendo hasta que termina la conexión de Live Connection de Net.Data. Se pueden conectar múltiples cliettes a una sola base de datos.

Como parte de la información de cliette en el archivo de configuración, deberá especificar un nombre de cliette, puertos exclusivos y el número mínimo y máximo de procesos. Para cliettes de bases de datos, también puede especificar el nombre de base de datos, el inicio de sesión y la contraseña para cada entrada de cliette.

Consejo respecto a la autorización: Asegúrese de que el ID de usuario que inicia Connection Manager tiene autorización para leer este archivo. Consulte el apartado “Cómo otorgar derechos de acceso a los archivos a los que accede Net.Data” en la página 65 para obtener más información.

Archivo de configuración de Cache Manager

El archivo de configuración de Cache Manager contiene las definiciones para el Cache Manager y para cada una de las antememorias. El almacenamiento en antememoria de Net.Data se describe en el apartado “Almacenamiento en antememoria de Net.Data” en la página 214. La configuración del Cache Manager se describe en el apartado “Configuración de las antememorias del Cache Manager y de Net.Data” en la página 222. La estructura del archivo consiste en una serie de secciones o stanzas:

Stanza de Cache Manager

Esta stanza define los parámetros del propio Cache Manager e incluye información de red, el estado de las anotaciones cronológicas y el estado de rastreo. La stanza es necesaria y debe llevar la etiqueta cache-manager.

Stanzas de definición de antememoria

Estas stanzas definen los parámetros para cada antememoria; existe una stanza de definición de antememoria en el archivo de configuración para cada antememoria gestionada por el Cache Manager; esta sección contiene la información de red, los requisitos de memoria y espacio, el estado de las anotaciones cronológicas y el estado de las estadísticas. La stanza de definición de antememoria es necesaria para cada antememoria gestionada por el Cache Manager.

El archivo de configuración de Cache Manager no se gestiona mediante la herramienta de administración y puede actualizarse con cualquier editor de texto. Consulte el apartado “Almacenamiento en antememoria de Net.Data” en la página 214 para obtener información sobre cómo definir este archivo.

Consejo respecto a la autorización: Asegúrese de que el ID de usuario que inicia el Cache Manager tiene derechos de acceso a este archivo. Consulte el apartado “Cómo otorgar derechos de acceso a los archivos a los que accede Net.Data” en la página 65 para obtener más información.

Secciones comunes de los archivos de macros, de control y de inicialización de Net.Data

Determinadas partes de los archivos de macros, configuración e inicialización de Net.Data deben ser coherentes para que todos los componentes de Net.Data funcionen como un conjunto. La tabla siguiente resume las áreas de cada uno de estos archivos que deben coincidir.

Tabla 2. Requisitos de coherencia para los archivos de configuración de Net.Data y la macro

Archivo	Secciones comunes	Notas
Archivo INI de Net.Data	Sentencia de entorno	Los entornos de lenguaje que utilizan Live Connection deben especificar el nombre de cliette de base de datos en la sentencia de entorno
	Variables de configuración de Live Connection	Al utilizar Live Connection de Net.Data, especifique el puerto de Live Connection, DTW_CM_PORT. Este valor de variable debe coincidir con el valor de MAIN_PORT del archivo de configuración de Live Connection.
	Variables de configuración de antememoria	Al utilizar el almacenamiento en antememoria de Net.Data, incluya opcionalmente variables de número de puerto y nombre de máquina. Estos valores deben coincidir con los empleados en el archivo de configuración de Cache Manager, en caso de utilizarse.
Archivo de configuración de Live Connection	Definiciones de cliette	Cada definición de cliette debe coincidir con una definición correspondiente en el archivo INI. Adicionalmente, el valor de MAIN_PORT debe coincidir con el valor de la variable DTW_CM_PORT del archivo INI.

Tabla 2. Requisitos de coherencia para los archivos de configuración de Net.Data y la macro (continuación)

Archivo	Secciones comunes	Notas
Archivo de configuración de Cache Manager	Variables de configuración de Cache Manager	Al utilizar el almacenamiento en antememoria de Net.Data, puede incluir opcionalmente variables de número de puerto y nombre de máquina. Estos valores deben coincidir con los empleados en el archivo INI, si se utilizan.

Los fragmentos siguientes ilustran la relación entre una macro, un archivo de inicialización de Net.Data y un archivo de configuración de Live Connection. La macro (DTW_SQL:SAMPLE, DTW_SQL:CELDIAL) utiliza dos cliettes y éstas acceden a dos bases de datos DB2, llamadas SAMPLE y CELDIAL. El archivo de configuración de Live Connection contiene los nombres y las definiciones de cliettes. La sentencia ENVIRONMENT del archivo de inicialización de Net.Data hace referencia al nombre de cliette. Los valores de LOGIN y PASSWORD se especifican en el archivo de configuración de Live Connection.

La Figura 2 en la página 11 muestra un fragmento de la macro que contiene la sentencia @DTW_ASSIGN que define qué cliette se debe utilizar para acceder a una base de datos.

```

<3*****>
<3** Esto es un comentario HTML **>
<3** Acceder a la base de datos SAMPLE utilizando **>
<3** el cliette DTW_SQL:SAMPLE **>
<3*****>
@DTW_ASSIGN (DATABASE, " SAMPLE ")
@insert_customer
(customer_name, customer_street, customer_city, customer_state,
customer_country, customer_zip, customer_credit, customer_expiry)

<3*****>
<3** Esto es un comentario HTML **>
<3** Procesar base de datos CELDIAL utilizando **>
<3** el cliette DTW_SQL:CELDIAL **>
<3*****>
@DTW_ASSIGN (DATABASE, " CELDIAL ")
@insert_customer
(customer_name, customer_street, customer_city, customer_state,
customer_country, customer_zip, customer_credit, customer_expiry)

```

Figura 2. Fragmento de macro de Net.Data

Tenga en cuenta que la variable de configuración DATABASE se sustituye en la sentencia ENVIRONMENT del archivo de inicialización para generar el nombre de cliette. Esto le permite acceder a múltiples bases de datos desde la misma macro.

La Figura 3 muestra un fragmento del archivo de inicialización de Net.Data que contiene la sentencia ENVIRONMENT y el tipo de cliette asociado. Existe una sentencia ENVIRONMENT para cada tipo de cliette en el archivo de inicialización. Para cada tipo de cliette de base de datos, la sentencia ENVIRONMENT especifica un nombre de cliette. El nombre está compuesto por el tipo de cliette y una referencia de variable, \$(DATABASE), que se resuelve en tiempo de ejecución. Cada entorno de lenguaje que utilice Live Connection deberá tener una definición de cliette en la sentencia ENVIRONMENT.

```

ENVIRONMENT (DTW_SQL)
(IN DATABASE, LOGIN, PASSWORD, TRANSACTION_SCOPE, SHOWSQL,
ALIGN, START_ROW_NUM, DTW_SET_TOTAL_ROWS)
CLLETTE "DTW_SQL:$(DATABASE)"

```

Figura 3. Fragmento de archivo de inicialización de Net.Data

La Figura 4 muestra un fragmento del archivo de configuración de Live Connection, que contiene las definiciones de cliette para DTW_SQL:CELDIAL y DTW_JAVAPPS.

```

CONNECTION_MANAGER{
  MAIN_PORT=7100
  ENCRYPTION=OFF
}

#####
# Esto es un comentario de un archivo de configuración de Live
# Connection. Los comentarios empiezan por el carácter de almohadilla.
# Los comentarios finalizan al final de la línea y no continúan en la
# línea siguiente a menos que aparezca otro carácter de almohadilla.
# Puede incluir comentarios al final de las líneas que contengan
# palabras clave de Live Connection, excepto en las líneas de contraseña.
# No puede incluir comentarios en líneas que contengan la palabra clave
# de contraseña.
# No puede incluir espacios y caracteres de almohadilla en ningún nombre,
# como el nombre de cliette o contraseñas de cliette de una base de datos.
#####
CLIETTE DTW_SQL:CELDIAL{
  MIN_PROCESS=1
  MAX_PROCESS=5
  EXEC_NAME=./dtwcdb2
  DATABASE=CELDIAL
  LOGIN=marshall
  PASSWORD=stlpwd
}

CLIETTE DTW_JAVAPPS{
  MIN_PROCESS=1
  MAX_PROCESS=5
  EXEC_NAME=./javaapp
}

```

Figura 4. Fragmento de archivo de configuración de Live Connection

Personalización del archivo de inicialización de Net.Data

La información contenida en el archivo de inicialización se especifica utilizando tres tipos de sentencias de configuración, descritas en las secciones siguientes:

- “Sentencias de variables de configuración” en la página 13
- “Sentencias de configuración de vía de acceso” en la página 22
- “Sentencias de configuración de entorno” en la página 28

El archivo de inicialización de ejemplo mostrado en la Figura 5 contiene ejemplos de estas sentencias y es válido para OS/2 y Windows NT.

El texto de cada sentencia de configuración individual debe estar todo en una sola línea. Asegúrese de que el archivo de inicialización contiene una sentencia ENVIRONMENT para cada entorno de lenguaje al que se llama desde las macros. Si califica totalmente todas las referencias a los archivos en la macro, no necesita especificar ninguna de las sentencias de configuración de vía de acceso.

<pre> 1 DTW_CM_PORT 7128 2 DTW_INST_DIR c:\db2www 3 DTW_LOG_DIR c:\db2www\logs 4 DB2INSTANCE DB2 5 DTW_DIRECT_REQUEST NO 6 DTW_SHOWSQL NO 7 DTW_UNICODE NO 8 DTW_MBMODE NO 9 MACRO_PATH c:\DB2WWW\Macro 10 HTML_PATH c:\www\html 11 INCLUDE_PATH c:\db2www\Macro 12 EXEC_PATH c:\db2www\Macro 13 FFI_PATH c:\pub\ffi;pub\ffi\data 14 ENVIRONMENT (DTW_SQL) [DLL path] [Parameter list] 15 ENVIRONMENT (DTW_ORA) [DLL path] [Parameter list] 16 ENVIRONMENT (DTW_ODBC) [DLL path] [Parameter list] 17 ENVIRONMENT (DTW_DEFAULT) [DLL path] [Parameter list] 18 ENVIRONMENT (DTW_APPLET) [DLL path] [Parameter list] 19 ENVIRONMENT (DTW_REXX) [DLL path] [Parameter list] 20 ENVIRONMENT (DTW_PERL) [DLL path] [Parameter list] 21 ENVIRONMENT (DTW_SYSTEM) [DLL path] [Parameter list] 22 ENVIRONMENT (DTW_FILE) [DLL path] [Parameter list] 23 ENVIRONMENT (DTW_WEBREG) [DLL path] [Parameter list] 24 ENVIRONMENT (DTW_JAVAPPS) [DLL path] [Parameter list] 25 ENVIRONMENT (HWS_LE) [DLL path] [Parameter list] </pre>	<ul style="list-style-type: none"> • Las líneas 1 a 8 definen variables de configuración • Las líneas 9 a 13 definen vías de acceso a archivos necesarios para procesar la macro • Las líneas 14 a 25 definen las sentencias de entorno que están disponibles.
--	---

Figura 5. Archivo de inicialización de Net.Data. Para obtener descripciones completas de la vía de acceso de DLL y la lista de parámetros, consulte el propio archivo db2www.ini y el apartado “Sentencias de configuración de entorno” en la página 28.

Las secciones siguientes describen cómo personalizar las sentencias de configuración del archivo de inicialización.

Sentencias de variables de configuración

Las sentencias de variables de configuración Net.Data establecen los valores de las variables de configuración. Las variables de configuración se utilizan para diversos fines. Algunas variables son necesarias para que un entorno de lenguaje funcione correctamente u opere en una modalidad alternativa. Otras variables controlan la codificación de caracteres o el contenido de la página

Web que se está creando. Adicionalmente, puede utilizar sentencias de variables de configuración para definir variables específicas de aplicaciones.

Las variables de configuración que utilice dependerán de los entornos de lenguaje y de las bases de datos que esté utilizando, así como de otros factores que son específicos de la aplicación.

Para actualizar las sentencias de variables de configuración:

Personalice el archivo de inicialización con las variables de configuración necesarias para la aplicación. Una variable de configuración tiene la sintaxis siguiente:

NOMBRE [=] serie-valor

El signo igual es opcional, como indican los corchetes.

Las subsecciones siguientes describen las sentencias de variables de configuración que se pueden especificar en el archivo de inicialización:

- “Variables de configuración de Cache Manager” en la página 15
- “DB2INSTANCE: Variable de instancia de DB2” en la página 16
- “DTW_CM_PORT: Variable de número de puerto de Live Connection” en la página 16
- “DTW_DEFAULT_ERROR_MESSAGE: Especificar mensajes de error genéricos” en la página 17
- “DTW_DIRECT_REQUEST: Habilitar variable de petición directa” en la página 17
- “DTW_INST_DIR: Variable de directorio de instalación de Net.Data” en la página 18
- “DTW_LOG_DIR y DTW_LOG_LEVEL: Variables de anotación cronológica de errores” en la página 18
- “DTW_LOG_LEVEL: Variable de nivel de anotación cronológica de errores” en la página 18
- “DTW_MBMODE: Variable de soporte de idioma nativo” en la página 18
- “DTW_REMOVE_WS: Variable para eliminar espacio en blanco adicional” en la página 19
- “DTW_SHOWSQL: Habilitar o inhabilitar la variable de configuración SHOWSQL” en la página 19
- “DTW_SMTP_SERVER: Variable de servidor SMTP de correo electrónico” en la página 20
- “DTW_UNICODE: Variable de Unicode” en la página 20
- “DTW_VARIABLE_SCOPE: Variable de ámbito de variable” en la página 22

Variables de configuración de Cache Manager

Se utilizan dos variables de configuración opcionales si el Cache Manager se ejecuta en una máquina distinta de la máquina donde se ejecuta la macro Net.Data:

- DTW_CACHE_PORT especifica qué número de puerto utiliza Net.Data para conectarse al Cache Manager.
- DTW_CACHE_HOST especifica el nombre de sistema principal TCP/IP de la máquina local o remota.

Si Cache Manager se ejecuta en la máquina local, se utilizan conexiones con nombre o sockets de dominio UNIX para la comunicación y no es necesaria ninguna configuración.

Cache Manager se ejecuta sólo en máquinas AIX y Windows NT. Consulte el apartado “Almacenamiento en antememoria de Net.Data” en la página 214 para obtener información sobre el almacenamiento en antememoria de Net.Data.

DTW_CACHE_PORT: Variable de puerto de Cache Manager

Especifica el puerto TCP/IP al que está conectado Cache Manager. Este número de puerto debe coincidir con el número de puerto especificado en el archivo de configuración de Cache Manager, de modo que Net.Data pueda comunicarse con Cache Manager. Si no se especifica, Cache Manager utiliza el puerto por omisión 7175.

Sintaxis:

DTW_CACHE_PORT [=] *número_puerto*

Parámetro:

número_puerto

Número de puerto exclusivo asignado a Cache Manager para atender las peticiones de antememoria. El valor por omisión es 7175.

La Tabla 3 describe las opciones para especificar los ID de máquina y los números de puerto para estas variables.

Tabla 3. Variables de configuración de Cache Manager: Opciones de configuración

Valores por omisión de Connection Manager	Si se especifica la máquina de antememoria ...	Si no se especifica la máquina de antememoria ...
Si se especifica el puerto de antememoria ...	Net.Data se conecta a Cache Manager en la máquina especificada utilizando el puerto especificado.	Net.Data se conecta a Cache Manager en la máquina local utilizando el puerto especificado.

Tabla 3. Variables de configuración de Cache Manager: Opciones de configuración (continuación)

Si no se especifica el puerto de antememoria ...	Net.Data se conecta a Cache Manager en la máquina especificada utilizando el puerto por omisión de 7175.	Net.Data se conecta a Cache Manager en la máquina local utilizando el puerto por omisión de 7175.
---	--	---

DTW_CACHE_HOST: Variable de ID de máquina de Cache Manager

Especifica la máquina donde reside Cache Manager. Si no se especifica, Net.Data supone que la máquina correcta es la máquina local.

Sintaxis:

`DTW_CACHE_HOST [=] nombre_sistpral`

Parámetro:

nombre_sistpral

Nombre de sistema principal TCP/IP calificado de la máquina local o remota donde se ejecuta Cache Manager. El valor por omisión es el nombre de sistema principal de la máquina local.

DB2INSTANCE: Variable de instancia de DB2

Especifica la instancia de DB2 utilizada por el entorno de lenguaje SQL. El valor de la variable es necesario cuando Net.Data se conecta a DB2 ejecutándose en los sistemas operativos Windows NT, OS/2 y UNIX.

DB2 en los sistemas operativos OS/2, Windows NT y UNIX necesita que se defina DB2INSTANCE como una variable de entorno. Si Net.Data detecta que DB2INSTANCE no se ha definido como una variable de entorno, establece la variable de entorno DB2INSTANCE en el valor de DB2INSTANCE que se encuentra en el archivo de inicialización antes de intentar conectarse a DB2.

Sintaxis:

`DB2INSTANCE [=] nombre_instancia`

DTW_CM_PORT: Variable de número de puerto de Live Connection

Especifica un número de puerto exclusivo que Net.Data utiliza para Live Connection.

Sintaxis:

`DTW_CM_PORT [=] número_puerto`

Donde *número_puerto* especifica el número de puerto exclusivo utilizado para Live Connection.

DTW_DEFAULT_ERROR_MESSAGE: Especificar mensajes de error genéricos

Utilice la variable de configuración DTW_DEFAULT_ERROR_MESSAGE para especificar un mensaje de error genérico para aplicaciones en producción. Esta variable proporciona un mensaje genérico para condiciones de error que no se capturan en ningún bloque MESSAGE.

Si aún desea ver los mensajes de error reales generados por Net.Data, utilice la anotación cronológica de mensajes de error para capturar los mensajes. Consulte el “Capítulo 8. Anotaciones cronológicas de Net.Data” en la página 245 para obtener información sobre la utilización de la anotación cronológica de errores.

Si no se especifica la variable de configuración, Net.Data visualiza su propio mensaje proporcionado para la condición de error.

Sintaxis:

```
DTW_DEFAULT_ERROR_MESSAGE [=] "mensaje"
```

Ejemplo: Especifica un mensaje genérico

```
DTW_DEFAULT_ERROR_MESSAGE "Este sitio no está disponible temporalmente."
```

DTW_DIRECT_REQUEST: Habilitar variable de petición directa

Habilita o inhabilita la invocación de petición directa. Por omisión, la petición directa está inhabilitada.

El método de petición directa para invocar Net.Data permite a un usuario especificar la ejecución de una sentencia de SQL o un programa Perl, REXX o C directamente en un URL. Cuando la petición directa está inhabilitada, el usuario debe invocar Net.Data utilizando el método de petición de macro, que sólo permite a los usuarios ejecutar las funciones y las sentencias de SQL definidas o llamadas en una macro. Consulte el apartado “Utilización de los mecanismos de Net.Data” en la página 71 para ver las recomendaciones relacionadas con la seguridad al utilizar DTW_DIRECT_REQUEST.

Sintaxis:

```
DTW_DIRECT_REQUEST [=] YES|NO
```

Donde:

YES Habilita la petición directa de Net.Data.

NO Inhabilita la petición directa de Net.Data. NO es el valor por omisión.

DTW_INST_DIR: Variable de directorio de instalación de Net.Data

Localiza determinados archivos durante la ejecución de Net.Data. Esta variable se establece durante la instalación para especificar el directorio inicial, `<dir_inst>`, donde se instala Net.Data. No cambie este valor después de la instalación.

DTW_LOG_DIR y DTW_LOG_LEVEL: Variables de anotación cronológica de errores

DTW_LOG_DIR especifica el directorio donde se almacenan las anotaciones cronológicas de errores. La anotación cronológica no se producirá a no ser que se establezcan esta variable y la variable DTW_LOG_LEVEL.

Consulte el apartado “Anotación cronológica de mensajes de error de Net.Data” en la página 245 para obtener más información sobre estas variables y sobre la anotación cronológica de mensajes de error con Net.Data.

Sintaxis:

```
DTW_LOG_DIR [=] \dir_inst\viaacceso
```

Ejemplo: Configuración de archivo de inicialización

```
DTW_LOG_DIR \dir_inst\mis_arch_anot_cron\
```

DTW_LOG_LEVEL: Variable de nivel de anotación cronológica de errores

DTW_LOG_LEVEL especifica el nivel de los errores que se deben registrar en las anotaciones cronológicas de errores. La anotación cronológica no se producirá a no ser que se establezcan esta variable y la variable DTW_LOG_DIR.

Consulte el apartado “Anotación cronológica de mensajes de error de Net.Data” en la página 245 para obtener más información sobre estas variables y sobre la anotación cronológica de mensajes de error con Net.Data.

Sintaxis:

```
DTW_LOG_LEVEL [=] off|warning|error
```

Ejemplo: Configuración de archivo de inicialización

```
DTW_LOG_LEVEL error
```

DTW_MBMODE: Variable de soporte de idioma nativo

Activa el soporte de idioma nacional para funciones de palabras y series. Cuando el valor de esta variable es YES, todas las funciones de series y palabras procesan correctamente los caracteres MBCS dentro de las series tratando a éstas como datos mixtos (es decir, como series que contienen potencialmente caracteres de los juegos de caracteres de un solo byte y de los juegos de caracteres de doble byte). El valor por omisión es NO. Puede alterar

temporalmente el valor establecido en el archivo de inicialización estableciendo la variable DTW_MBMODE en una macro de Net.Data.

Esta variable de configuración funciona con la variable de configuración DTW_UNICODE. Si DTW_UNICODE utiliza el valor por omisión de NO, se utiliza el valor de DTW_MBMODE. Si se establece DTW_UNICODE en un valor distinto de NO, se utiliza su valor. La Tabla 4 ilustra cómo los valores de estas dos variables determinan el modo en que las funciones incorporadas procesan las series:

Tabla 4. Relación entre los valores de DTW_UNICODE y DTW_MBMODE

Si DTW_UNICODE se establece en	Si DTW_MBMODE=YES	Si DTW_MBMODE=NO
NO	Soporta MBCS combinado con SBCS	Sólo soporta SBCS
UTF8	Soporta UTF-8	Soporta UTF-8

Sintaxis:

DTW_MBMODE [=] NO|YES

DTW_REMOVE_WS: Variable para eliminar espacio en blanco adicional

Cuando esta variable se establece en YES, Net.Data elimina los espacios en blanco innecesarios de la salida HTML. Al comprimir el espacio en blanco, esta variable reduce la cantidad de datos enviados al navegador Web, mejorando de este modo el rendimiento. El valor por omisión es NO.

Puede alterar temporalmente esta variable en la macro utilizando la sentencia DEFINE.

Consejo: No establezca DTW_REMOVE_WS en YES si está utilizando DTW_PRINT_HEADER para generar sus propias cabeceras (DTW_PRINT_HEADER "NO").

Sintaxis:

DTW_REMOVE_WS [=] YES|NO

DTW_SHOWSQL: Habilitar o inhabilitar la variable de configuración SHOWSQL

Altera temporalmente el efecto de establecer SHOWSQL en las macros de Net.Data.

Sintaxis:

DTW_SHOWSQL [=] YES|NO

Donde:

- YES** Habilita SHOWSQL en cualquier macro que establezca el valor de SHOWSQL en YES.
- NO** Inhabilita SHOWSQL en las macros, incluso si la variable SHOWSQL está establecida en YES. NO es el valor por omisión.

La Tabla 5 describe cómo los valores del archivo de inicialización de Net.Data y de la macro determinan si la variable SHOWSQL se habilita o se inhabilita para una macro determinada.

Tabla 5. Relación entre los valores del archivo de inicialización de Net.Data y de la macro para SHOWSQL

Valor de DTW_SHOWSQL	Valor SHOWSQL	Se visualiza la sentencia de SQL
NO	NO	NO
NO	YES	NO
YES	NO	NO
YES	YES	SÍ

DTW_SMTP_SERVER: Variable de servidor SMTP de correo electrónico

Especifica el servidor SMTP a utilizar para enviar mensajes de correo electrónico utilizando la función incorporada DTW_SENDMAIL. El valor de esta variable puede ser un nombre de sistema principal o una dirección IP. Si no se establece este variable, Net.Data utiliza el sistema principal local como servidor SMTP.

Sintaxis:

DTW_SMTP_SERVER [=] *nombre_servidor*

Donde *nombre_servidor* es el nombre de sistema principal o la dirección IP del servidor SMTP que se deberá utilizar para enviar mensajes de correo electrónico.

Consejo respecto al rendimiento: Especifique una dirección IP para este valor a fin de impedir que Net.Data se conecte a un servidor de nombres de dominio al recuperar la dirección IP del servidor SMTP especificado.

Ejemplo:

DTW_SMTP_SERVER us.ibm.com

DTW_UNICODE: Variable de Unicode

Especifica si Net.Data soporta Unicode en:

- Las macros
- Los datos de formulario

- Los datos recuperados de una base de datos DB2
- Las series procesadas por funciones incorporadas de Net.Data

Net.Data soporta el formato UTF-8 Unicode en las macros, los datos de formulario y las funciones incorporadas, y la salida es siempre en UTF-8. Net.Data puede acceder a una base de datos que contenga datos UTF-16 y convertirlos a UTF-8.

Cuando se establece en UTF8, DTW_UNICODE indica a Net.Data que se ejecute en un entorno Unicode. Entonces Net.Data genera páginas en UTF-8 y espera que los datos de entrada estén en formato UTF-8 (o, en el caso de datos de base de datos DB2, se acepta UCS-2). Los datos de entrada incluyen el contenido del archivo de macro, los datos de formulario enviados desde el navegador y todos los demás datos que vienen de fuentes de datos externas.

Requisito de base de datos DB2 Unicode: Además de establecer la variable DTW_UNICODE, establezca también la variable de entorno específica de DB2, DB2CODEPAGE, en 1208 en el entorno en el que se ejecuta Net.Data. Por ejemplo, para el servidor Web Apache, añada la línea siguiente al archivo HTTPD.CONF:

```
SetEnv DB2CODEPAGE 1208
```

Consulte la documentación del servidor Web para determinar cómo establecer variables de entorno para scripts CGI, API de servidor Web, programas Fast-CGI o servlets.

Net.Data utiliza el catálogo de mensajes en inglés cuando se ejecuta en un entorno Unicode.

La variable de configuración DTW_UNICODE funciona con la variable de configuración DTW_MBMODE. El valor de la variable de configuración DTW_UNICODE prevalece sobre el valor de la variable DTW_MBMODE cuando se procesan funciones incorporadas de palabra y de serie. Pero, si DTW_UNICODE se establece en NO o no se establece, se utiliza el valor de DTW_MBMODE. La Tabla 4 en la página 19 ilustra cómo los valores de estas dos variables determinan el modo en que las funciones incorporadas procesan las series:

Sintaxis:

```
DTW_UNICODE [=] NO|UTF8
```

Donde:

NO Especifica diferir al valor de la variable DTW_MBMODE. La Tabla 4 en la página 19 describe el soporte de Net.Data basándose en el valor de DTW_MBMODE.

UTF8 Especifica soportar la página de códigos UTF-8 e ignorar el valor de la variable de configuración DTW_MBMODE. UTF-8 representa los caracteres mediante un número variable de bytes y permite trabajar con seguridad bajo ASCII.

DTW_VARIABLE_SCOPE: Variable de ámbito de variable

Especifica cómo trata Net.Data el ámbito de variables locales: las variables locales siguen siendo locales o las variables locales pueden utilizarse fuera del bloque de función en el que se han creado. Esta variable se proporciona por compatibilidad con versiones anteriores de Net.Data y no está disponible con las versiones de Net.Data para OS/390 o para OS/400.

Sintaxis:

DTW_VARIABLE_SCOPE = LOCAL|GLOBAL

Donde:

LOCAL

Especifica que las variables locales permanecen locales. Este comportamiento se introdujo con Net.Data Versión 2.0 y es el valor por omisión.

GLOBAL

Especifica que las variables locales pueden utilizarse fuera del bloque de función en el que se han creado. Se proporciona por compatibilidad con versiones anteriores de Net.Data; LOCAL es el valor recomendado.

Sentencias de configuración de vía de acceso

Net.Data determina la ubicación de los archivos y los programas ejecutables utilizados por las macros de Net.Data a partir de los valores de las sentencias de configuración de vía de acceso. Las sentencias de vía de acceso son:

- “DTW_UPLOAD_DIR” en la página 23
- “EXEC_PATH” en la página 23
- “FFI_PATH” en la página 24
- “HTML_PATH” en la página 25
- “INCLUDE_PATH” en la página 25
- “MACRO_PATH” en la página 27

Estas sentencias de vía de acceso identifican uno o más directorios en los que Net.Data busca al intentar localizar macros, archivos ejecutables, archivos de texto, archivos LOB y archivos de inclusión. Las sentencias de vía de acceso que necesite dependerán de las posibilidades de Net.Data que utilicen las macros.

Directrices para la actualización:

Se aplican una directrices generales a las sentencias de vía de acceso. Las excepciones se indican en la descripción de cada sentencia de vía de acceso.

- Separe cada directorio especificado de la sentencia de vía de acceso con un punto y coma (;).
- Cada sentencia de vía de acceso puede especificar varias vías de acceso, excepto para HTML_PATH, que sólo puede tener una sentencia de vía de acceso. Las búsquedas en las vías de acceso se realizan de izquierda a derecha en el orden especificado. Esta posibilidad de múltiples vías de acceso le permite organizar los archivos en múltiples directorios. Por ejemplo, puede poner cada una de las aplicaciones Web en su propio directorio.
- Se recomienda utilizar sentencias de vía de acceso absoluta.

Las secciones siguientes describen la finalidad y la sintaxis de cada sentencia de vía de acceso y proporcionan ejemplos de sentencias de vía de acceso válidas. Los ejemplos pueden diferir de lo que usted verá en la aplicación, en función del sistema operativo y de la configuración.

DTW_UPLOAD_DIR

Esta sentencia de configuración de directorio especifica en qué lugar del servidor se deben almacenar los archivos subidos cuando un navegador de cliente envía una petición directa que contiene un tipo de archivo. Cuando no se establece esta variable, Net.Data no acepta los archivos para subirlos.

Sintaxis:

DTW_UPLOAD_DIR [=] víaacceso

Ejemplo:

DTW_UPLOAD_DIR /usr/lpp/internet/server_root/pub/upload

EXEC_PATH

Esta sentencia de configuración de vía de acceso identifica uno o más directorios en los que Net.Data busca un programa externo que se invoca mediante la sentencia EXEC o una variable ejecutable. El orden de los directorios de la sentencia de vía de acceso determina el orden en que Net.Data busca en los directorios. Si se encuentra el programa, se añade el nombre de programa externo a la especificación de vía de acceso, produciendo un nombre de archivo totalmente calificado que se pasa al entorno de lenguaje para su ejecución.

Sintaxis:

EXEC_PATH [=] víaacceso1;víaacceso2;...;víaacceson

Ejemplo: El ejemplo siguiente muestra la sentencia EXEC PATH en el archivo de inicialización y la sentencia EXEC en la macro que invoca un programa externo.

Archivo de inicialización de Net.Data:

```
EXEC_PATH /u/user1/prgms;/usr/lpp/netdata/prgms;
```

Macro de Net.Data:

```
%FUNCTION(DTW_REXX) myFunction() {  
    %EXEC{ myFunction.cmd %}  
%}
```

Si se encuentra el archivo myFunction.cmd en el directorio /usr/lpp/netdata/prgms, el nombre calificado del programa es /usr/lpp/netdata/prgms/myFunction.cmd.

Si el archivo no se encuentra en los directorios especificados en la sentencia EXEC_PATH:

- Si la vía de acceso especificada es absoluta, Net.Data busca el archivo en la vía de acceso especificada. Por ejemplo, si se somete el URL siguiente:

```
http://myserver/cgi-bin/db2www/usr/user1/prgms/myFunction.cmd
```

Net.Data busca el archivo en la vía de acceso de directorio /u/user1/prgms/myFunction.cmd.

- Si la vía de acceso especificada es relativa, Net.Data busca en el directorio de trabajo actual. Por ejemplo, si se somete el URL siguiente:

```
http://myserver/cgi-bin/db2www/myFunction.cmd/report
```

y no se ha encontrado el archivo myFunction.cmd en ninguno de los directorios especificados en EXEC_PATH, Net.Data intenta encontrar el archivo en el directorio de trabajo actual.

FFI_PATH

Esta sentencia de configuración de vía de acceso identifica uno o más directorios en los que Net.Data busca, en el orden en el que se han especificado, un archivo plano al que hace referencia una función de interfaz de archivo plano (FFI).

Sintaxis:

```
FFI_PATH [=] víaacceso1;víaacceso2;...;víaacceson
```

Ejemplo: El ejemplo siguiente muestra una sentencia FFI_PATH del archivo de inicialización.

Archivo de inicialización de Net.Data:

```
FFI_PATH /u/user1/ffi;/usr/lpp/netdata/ffi;
```

Cuando se llama al entorno de lenguaje FFI, Net.Data busca en la vía de acceso especificada en la sentencia FFI_PATH.

Dado que la sentencia FFI_PATH se utiliza para proporcionar seguridad a los archivos que no están en directorios de la sentencia de vía de acceso, existen condiciones especiales para los archivos FFI que no se encuentran. Consulte la sección de funciones incorporadas FFI en el manual *Net.Data Guía de consulta*.

HTML_PATH

Esta sentencia de configuración de directorio especifica en qué directorio graba Net.Data los objetos grandes (LOB). Esta sentencia de vía de acceso sólo acepta una vía de acceso de directorio.

Durante la instalación, Net.Data crea un directorio llamado tmplobs, bajo el directorio especificado en la variable de configuración de vía de acceso HTML_PATH. Net.Data almacena todos los archivos LOB en este directorio. Si cambia el valor de HTML_PATH, cree un subdirectorio nuevo bajo el nuevo directorio.

Sintaxis:

```
HTML_PATH [=] víaacceso
```

Ejemplo: El ejemplo siguiente muestra la sentencia HTML_PATH del archivo de inicialización.

Archivo de inicialización de Net.Data:

```
HTML_PATH /db2/lobs
```

Cuando una consulta devuelve un LOB, Net.Data lo guarda en el directorio especificado en la sentencia de configuración HTML_PATH.

Consejo respecto al rendimiento: Tenga en cuenta las limitaciones del sistema cuando utilice los LOB porque éstos pueden consumir recursos rápidamente. Consulte el apartado “Utilización de objetos grandes” en la página 168 para obtener más información.

INCLUDE_PATH

Esta sentencia de configuración de vía de acceso identifica uno o más directorios en los que Net.Data realiza la búsqueda, en el orden en que se han especificado, para encontrar un archivo especificado en una sentencia INCLUDE en una macro de Net.Data. Cuando encuentra el archivo, Net.Data añade el nombre del archivo de inclusión (include) a la especificación de vía de acceso para producir el nombre de archivo de inclusión calificado.

Sintaxis:

```
INCLUDE_PATH [=] víaacceso1;víaacceso2;...;víaacceson
```

Ejemplo 1: El ejemplo siguiente muestra la sentencia INCLUDE_PATH del archivo de inicialización y la sentencia INCLUDE que especifica el archivo de inclusión.

Archivo de inicialización de Net.Data:

```
INCLUDE_PATH /u/user1/includes;/usr/lpp/netdata/includes
```

Macro de Net.Data:

```
%INCLUDE "myInclude.txt"
```

Si el archivo *myInclude.txt* se encuentra en el directorio /u/user1/includes, el nombre totalmente calificado del archivo de inclusión es /u/user1/includes/myInclude.txt.

Ejemplo 2: El ejemplo siguiente muestra la sentencia INCLUDE_PATH y un archivo INCLUDE con un nombre de subdirectorio.

Archivo de inicialización de Net.Data:

```
INCLUDE_PATH /u/user1/includes;/usr/lpp/netdata/includes
```

Macro de Net.Data:

```
%INCLUDE "OE/oeheader.inc"
```

El archivo de inclusión se busca en los directorios /u/user1/includes/OE y /usr/lpp/netdata/includes/OE. Si se encuentra el archivo en /usr/lpp/netdata/includes/OE, el nombre totalmente calificado del archivo de inclusión es /usr/lpp/netdata/includes/OE/oeheader.inc.

Si el archivo no se encuentra en los directorios especificados en la sentencia INCLUDE_PATH:

- Si la vía de acceso especificada es absoluta, Net.Data busca el archivo en la vía de acceso especificada. Por ejemplo, si se somete el URL siguiente:

```
http://myserver/cgi-bin/db2www/u/user1/includes/oeheader.inc
```

Net.Data busca el archivo en la vía de acceso de directorio /u/user1/includes/oeheader.inc.

- Si la vía de acceso especificada es relativa, Net.Data busca en el directorio de trabajo actual. Por ejemplo, si se somete el URL siguiente:

```
http://myserver/cgi-bin/db2www/my.cmd/report
```

y no se ha encontrado el archivo `myFunction.cmd` en ninguno de los directorios especificados en `INCLUDE_PATH`, `Net.Data` intenta encontrar el archivo en el directorio de trabajo actual.

MACRO_PATH

Esta sentencia de configuración de vía de acceso identifica los directorios en los que `Net.Data` busca las macros de `Net.Data`. Por ejemplo, la especificación del URL siguiente solicita la macro de `Net.Data` con la vía de acceso y el nombre de archivo `/macro/sqlm.d2w`:

```
http://servidor/cgi-bin/db2www/macro/sqlm.d2w/report
```

Sintaxis:

```
MACRO_PATH [=] víaacceso1;víaacceso2;...;víaacceson
```

El signo igual (=) es opcional, como indican los corchetes.

`Net.Data` añade la vía de acceso `/macro/sqlm.d2w` a las vías de acceso de la sentencia de configuración `MACRO_PATH`, de izquierda a derecha hasta que `Net.Data` encuentra la macro o busca en todas las vías de acceso. Consulte el “Capítulo 4. Invocación de `Net.Data`” en la página 79 para obtener información sobre cómo invocar macros de `Net.Data`.

Ejemplo: El ejemplo siguiente muestra la sentencia `MACRO_PATH` del archivo de inicialización y el enlace relacionado que invoca `Net.Data`.

Archivo de inicialización de `Net.Data`:

```
MACRO_PATH /u/user1/macros;/usr/lpp/netdata/macros
```

Enlace HTML:

```
<a href="http://servidor/cgi-bin/db2www/query.d2w/input">Someter otra consulta.</a>
```

Si se encuentra el archivo `query.d2w` en el directorio `/u/user1/macros`, la vía de acceso totalmente calificada es `/u/user1/macros/query.d2w`.

Si el archivo no se encuentra en los directorios especificados en la sentencia `MACRO_PATH`:

- Si la vía de acceso especificada es absoluta, `Net.Data` busca el archivo en la vía de acceso especificada. Por ejemplo, si se somete el URL siguiente:

```
http://servidor/cgi-bin/db2www/u/user1/macros/myfile.txt/report
```

`Net.Data` busca el archivo en la vía de acceso de directorio `/u/user1/macros/myfile.txt`.

- Si la vía de acceso especificada es relativa, `Net.Data` busca el archivo en todos los directorios, empezando por el directorio raíz (/). Por ejemplo, si se somete el URL siguiente:

`http://servidor/cgi-bin/db2www/myfile.txt/report`

y el archivo `myfile.txt` no se ha encontrado en ninguno de los directorios especificados en `MACRO_PATH`, Net.Data intentará encontrar el archivo en el directorio raíz (/): `/myfile.txt`

Sentencias de configuración de entorno

Una sentencia `ENVIRONMENT` configura un entorno de lenguaje. Un entorno de lenguaje es un componente de Net.Data que éste utiliza para acceder a una fuente de datos, por ejemplo una base de datos DB2, o para ejecutar un programa escrito en un lenguaje como, por ejemplo, REXX. Net.Data proporciona un conjunto de entornos de lenguaje, así como una interfaz que permite crear entornos de lenguaje propios. Estos entornos de lenguaje se describen en el “Capítulo 6. Utilización de entornos de lenguaje” en la página 159 y la interfaz de entorno de lenguaje se describe en el manual *Net.Data Language Environment Interface Reference*.

Net.Data necesita que exista una sentencia `ENVIRONMENT` para un entorno de lenguaje determinado para que se pueda invocar dicho entorno de lenguaje.

Puede asociar variables con un entorno de lenguaje especificando las variables como parámetros en la sentencia `ENVIRONMENT`. Net.Data pasa implícitamente al entorno de lenguaje los parámetros especificados en una sentencia `ENVIRONMENT` como variables de macro. Para cambiar el valor de un parámetro especificado en una sentencia `ENVIRONMENT` de la macro, asigne un valor a la variable utilizando la función `DTW_ASSIGN()` o defina la variable en una sección `DEFINE`. **Importante:** Si una variable de macro se define en una macro pero no se especifica en la sentencia `ENVIRONMENT`, la variable de macro no se pasará al entorno de lenguaje.

Por ejemplo, una macro puede definir una variable `DATABASE` para especificar el nombre de una base de datos en la que debe ejecutarse una sentencia de SQL de una función `DTW_SQL`. El valor de `DATABASE` debe pasarse al entorno de lenguaje SQL (`DTW_SQL`) para que el entorno de lenguaje SQL pueda conectarse a la base de datos designada. Para pasar la variable al entorno de lenguaje, deberá añadir la variable `DATABASE` a la lista de parámetros de la sentencia de entorno para `DTW_SQL`.

El archivo de inicialización de Net.Data de ejemplo realiza varias suposiciones respecto a la personalización del valor de las sentencias de configuración de entorno de Net.Data. Puede que estas suposiciones no sean correctas para su entorno. Modifique las sentencias de forma apropiada para el entorno correspondiente.

Añadir o actualizar una sentencia ENVIRONMENT:

Las sentencias ENVIRONMENT tienen la sintaxis siguiente:

```
ENVIRONMENT(tipo)  
nombre_biblioteca (lista_parámetros, ...) [CLIETTE "nombre_cliette"]
```

Parámetros:

- *tipo*

Nombre por el que Net.Data asocia este entorno de lenguaje con un bloque FUNCTION definido en una macro de Net.Data. Deberá especificar el tipo del entorno de lenguaje en una definición de bloque FUNCTION para identificar el entorno de lenguaje que Net.Data deberá utilizar para ejecutar la función.

- *nombre_biblioteca*

Nombre de la DLL o biblioteca compartida que contiene las interfaces de entorno de lenguaje a las que llama Net.Data.

- En AIX, el nombre de la biblioteca compartida se especifica con la extensión *.o*.
- En HP-UX, el nombre de la biblioteca compartida se especifica con la extensión *.sl*
- En SUN, PTX y LINUX el nombre de la biblioteca compartida se especifica con la extensión *.so*
- En OS/2 y Windows NT el nombre de la biblioteca compartida se especifica con la extensión *.dll*.

- *lista_parámetros*

Lista de parámetros que se pasan al entorno de lenguaje en cada llamada de función, además de los parámetros que se especifican en la definición de bloque FUNCTION.

Para establecer y pasar las variables de la lista de parámetros, defina la variable en la macro.

Deberá definir estos parámetros como variables de configuración o como variables en la macro antes de ejecutar una función que vaya a ser procesada por el entorno de lenguaje. El ejemplo siguiente especifica las variables en la sentencia ENVIRONMENT:

```
ENVIRONMENT(DTW_SQL) C:\WINNT\System32\nddb2.dll (IN  
DATABASE, TRANSACTION_SCOPE, USERID, PASSWORD)
```

Si una función modifica cualquiera de sus parámetros de salida, los parámetros mantienen su valor modificado después de que se haya completado la función.

- *nombre_cliette*

Nombre de la cliette. El *nombre_cliette* puede hacer referencia a la cliette de entorno de lenguaje Java Application o puede ser una cliette de base de datos. El parámetro *nombre_cliette* se utiliza con la palabra clave CLIETTE, y ambos se utilizan solamente con Live Connection. CLIETTE y *nombre_cliette* son opcionales y sólo se pueden especificar para entornos de lenguaje de base de datos y de aplicaciones Java.

Cliette de Java Application

Este nombre de cliette especifica el entorno de lenguaje Java Application.

Sintaxis:

```
CLIETTE "DTW_JAVAPPS"
```

Cliette de base de datos

Este nombre de cliette especifica una cliette que está asociada con una base de datos.

Sintaxis:

```
CLIETTE "tipo:nombre_bd"
```

Parámetros:

tipo Entorno de lenguaje de base de datos asociado con la cliette. Consulte la página 62 para obtener una lista de tipos válidos.

nombre_bd

Nombre de cliette de base de datos. Este nombre es generalmente el mismo que el de la base de datos con la que está asociada la cliette, por ejemplo MYDBASE, pero también puede ser otro nombre. *nombre_bd* es opcional cuando se utiliza el entorno de lenguaje Oracle.

Cuando Net.Data procesa el archivo de inicialización, no carga las DLL o las bibliotecas compartidas del entorno de lenguaje. Net.Data carga una DLL o una biblioteca compartida de entorno de lenguaje cuando ejecuta por primera vez una función que identifica dicho entorno de lenguaje. Entonces la DLL o la biblioteca compartida permanece cargada durante el tiempo que Net.Data está cargado.

Ejemplo: Sentencias ENVIRONMENT para entornos de lenguaje proporcionados por Net.Data

Cuando personalice las sentencias ENVIRONMENT para la aplicación, añada las variables a las sentencias ENVIRONMENT que necesitan pasarse del archivo de inicialización a un entorno de lenguaje o que los escritores de macros de Net.Data necesitan establecer o alterar temporalmente en las macros.

```

ENVIRONMENT (DTW_SQL) /net.data/lib/dtwsq1.so ( IN DATABASE, LOGIN, PASSWORD,
TRANSACTION SCOPE, ALIGN, START_ROW_NUM, DTW_SET_TOTAL_ROWS)
CLLETTE "DTW_SQL:MYDBASE"
ENVIRONMENT (DTW_ORA) /net.data/lib/dtwora.so ( IN DATABASE, LOGIN, PASSWORD,
TRANSACTION SCOPE, ALIGN, START_ROW_NUM, DTW_SET_TOTAL_ROWS)
ENVIRONMENT (DTW_ODBC) /net.data/lib/dtwodbc.so ( IN DATABASE, LOGIN, PASSWORD,
TRANSACTION SCOPE, ALIGN, DTW_SET_TOTAL_ROWS)
ENVIRONMENT (DTW_APPLET) /net.data/lib/dtwjava.so ( )
ENVIRONMENT (DTW_JAVAPPS) /net.data/lib/dtwjavapps.so ( OUT RETURN_CODE )
CLLETTE "DTW_JAVAPPS"
ENVIRONMENT (DTW_PERL) /net.data/lib/dtwperl.so ( OUT RETURN_CODE )
ENVIRONMENT (DTW_REXX) /net.data/lib/dtwrexx.so ( OUT RETURN_CODE )
ENVIRONMENT (DTW_SYSTEM) dtwsys.so ( OUT RETURN_CODE )
ENVIRONMENT (HWS_LE) dtwhws.so ( OUT RETURN_CODE )

```

Necesario: Cada sentencia ENVIRONMENT debe estar en una sola línea.

Configuración de los entornos de lenguaje

Después de modificar las variables de configuración y las sentencias de configuración ENVIRONMENT para los entornos de lenguaje de Net.Data, es necesario realizar una configuración adicional para que los entornos de lenguaje siguientes puedan funcionar correctamente. Las secciones siguientes describen los pasos necesarios para configurar los entornos de lenguaje:

- “Configuración del entorno de lenguaje IMS Web”
- “Configuración del entorno de lenguaje Java” en la página 32
- “Configuración del entorno de lenguaje Oracle” en la página 33

Configuración del entorno de lenguaje IMS Web

Para utilizar el entorno de lenguaje IMS Web, deberá realizar los pasos siguientes:

1. Instale el componente IMS Web Runtime en el servidor Web que ejecuta Net.Data. Para obtener información sobre cómo instalar y utilizar el componente IMS Web Runtime, consulte el manual *IMS Web User's Guide*:
<http://www.ibm.com/software/data/ims/about/imsweb/document/>
2. Cree la DLL de transacciones.
 - a. Genere el código C++, el archivo make (DTWproj.mak) y los archivos de macros de Net.Data (DTWproj.d2w) para la transacción con la herramienta IMS Web Studio.
 - b. Si está ejecutando Net.Data en un sistema operativo diferente del sistema operativo en el que se ejecuta la herramienta IMS Web Studio, mueva los archivos fuente DLL a una máquina de desarrollo de IMS Web para el sistema operativo de destino. Por ejemplo, si ejecuta la herramienta IMS Web Studio en Windows NT y la plataforma de destino es AIX u OS/390, mueva la fuente para la DLL de transacciones a una máquina de desarrollo de IMS Web que se ejecute bajo AIX u OS/390, respectivamente.

- c. Cree el formulario ejecutable de la DLL de transacciones utilizando el archivo make generado.
3. Copie el archivo DLL de transacciones (DTWproj.dll) y la macro de Net.Data (DTWproj.d2w) en el servidor Web.
 - a. Ponga la macro en un directorio del que Net.Data recupere macros. (Consulte el apartado “MACRO_PATH” en la página 27 para obtener más información).
 - b. Ponga la DLL o la biblioteca compartida de transacciones en un directorio del que el servidor Web recupere las DLL o las bibliotecas compartidas.
4. Utilice el enlace del archivo de ejemplo generado por la herramienta IMS Web Studio, DTWproj.htm, para modificar un archivo HTML en el árbol HTML del servidor Web. Entonces puede utilizar el enlace para invocar Net.Data y visualizar el formulario HTML de entrada para invocar el entorno de lenguaje IMS Web. A continuación, el entorno de lenguaje IMS Web llama a la DLL de transacciones IMS, que utiliza los servicios probados por las DLL de IMS Web Runtime para ejecutar la transacción y devolver su salida al navegador Web.

Las DLL de IMS Web Runtime crean y envían un mensaje de petición mediante IMS TOC a OTMA, lo cual, a su vez, hace que la transacción apropiada se ponga en cola. A continuación, OTMA devuelve a IMS Web la salida de la transacción mediante IMS TOC. Entonces se vuelve a pasar la transacción a Net.Data mediante el entorno de lenguaje IMS Web para visualizarla en el navegador Web.

Configuración del entorno de lenguaje Java

El entorno de lenguaje Java requiere una configuración adicional para poder llamar a las funciones desde una macro:

1. Cree un archivo de proceso por lotes para arrancar la aplicación Java porque Net.Data no puede iniciar directamente una aplicación Java. Net.Data utiliza este archivo para arrancar la Java Virtual Machine (Máquina virtual Java), que ejecuta la función de Java. El archivo de proceso por lotes debe incluir la sentencia de vía de acceso de clase java (java-classpath) para asegurar que se pueden encontrar los paquetes Java necesarios (los paquetes estándares y específicos de aplicación). Por ejemplo, el archivo de proceso por lotes, launchjv.bat, contiene la sentencia de vía de acceso de clase java siguiente:


```
java -classpath %CLASSPATH%;C:\DB2WWW\Javaclas dtw_samp %1 %2 %3 %4 %5 %6
```
2. Defina una cliette para trabajar con el entorno de lenguaje Java en el archivo de configuración de Live Connection, dtwcm.cnf. Especifique números de puerto exclusivos para la cliette y el nombre de archivo de proceso por lotes relacionado con la variable de configuración EXEC_NAME. En el ejemplo siguiente, el nombre de cliette Java se define

como DTW_JAVAPPS y la variable de configuración EXEC_NAME se establece en el nombre del archivo de proceso por lotes, launchjv.bat:

```
CLIETTE DTW_JAVAPPS{
MIN_PROCESS=1           <= Necesario: este valor debe ser 1 porque
                           la cliette JAVAPPS es de múltiples hebras.
MAX_PROCESS=1           <= Necesario: este valor debe ser 1 porque
                           la cliente JAVAPPS es de múltiples hebras.
EXEC_NAME=launchjv.bat  <= Nombre del archivo de proceso por lotes
                           que incluye las sentencias classpath
}
```

Al iniciar el Net.Data Connection Manager, Net.Data inicia la cliette Java especificada en el archivo de configuración. La cliette queda disponible para procesar las peticiones de entorno de lenguaje Java de las aplicaciones de macro Net.Data.

3. Actualice la sentencia de vía de acceso DTW_JAVAPPS ENVIRONMENT del archivo de inicialización de Net.Data, db2www.ini, añadiendo cada
4. nombre de cliette a la sentencia. Por ejemplo:

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

Configuración del entorno de lenguaje Oracle

Utilice los pasos siguientes para acceder a bases de datos Oracle desde una macro de Net.Data:

1. Asegúrese de que los componentes apropiados de Oracle estén instalados y funcionen del modo siguiente:
 - a. Si SQL*Net aún no está instalado, instálelo en la máquina donde está instalado Net.Data. Para obtener más información, consulte el URL siguiente:
http://www.oracle.com/products/networking/html/stdn_sqlnet.html
 - b. Verifique que la función *tnsping* de Oracle pueda utilizarse con la misma autorización de seguridad que utiliza el servidor Web. Para realizar la verificación, conéctese con el tipo y el ID de usuario del servidor Web:
tnsping nombre-instancia-oracle

Donde *nombre-instancia-oracle* es el nombre del sistema Oracle al que acceden las macros de Net.Data.

Es posible que no pueda verificar la función *tnsping* en Windows NT si el servidor Web se ejecuta bajo la autorización de sistema. Si es así, sáltese este paso.

- c. Verifique que se puede acceder a las tablas de Oracle con la misma autorización de seguridad que utiliza el servidor Web. Para realizar la verificación, entre una sentencia SELECT de SQL, utilizando la

herramienta de mandato de línea SQL*Plus, para acceder a una tabla Oracle con una sentencia SELECT de SQL con la autorización del servidor Web. Por ejemplo:

```
SELECT * FROM nombretabla
```

Es posible que no pueda verificar el acceso a tabla en Windows NT si el servidor Web se ejecuta bajo la autorización de sistema. Si es así, sáltese este paso.

Resolución de problemas: No continúe si fallan los pasos anteriores. Si falla alguno de los pasos anteriores, compruebe la configuración de Oracle.

2. Asegúrese de que las variables de entorno de Oracle estén establecidas correctamente en el proceso de servidor Web.

- Para AIX, ponga las líneas siguientes en el archivo `/etc/environment` o en el archivo `.profile` para el ID de usuario bajo el que se está ejecutando el servidor Web:

```
ORACLE_SID=nombre-instancia-oracle  
ORACLE_HOME=directorio-biblioteca-ejecución-oracle
```

- Para Windows NT, utilice el Panel de control de Propiedades del sistema para añadir las variables de entorno siguientes:

```
ORACLE_SID=nombre-instancia-oracle  
ORACLE_HOME=directorio-biblioteca-ejecución-oracle
```

Consejo: Puede que necesite líneas adicionales para otras variables de entorno de Oracle, en función de los recursos de Oracle que piense utilizar, por ejemplo soporte de idioma nacional y compromiso de dos fases. Consulte la documentación de administración de Oracle para obtener más información sobre estas variables de entorno.

3. Pruebe la conexión a Oracle desde Net.Data. En la macro de Net.Data, especifique los valores apropiados en las variables LOGIN y PASSWORD. No defina la variable DATABASE de Net.Data al acceder a las bases de datos Oracle. A continuación se muestra un ejemplo de sentencia de conexión en una macro:

```
%DEFINE LOGIN=ID_usuario@nombre-instancia-oracle-remota  
%DEFINE PASSWORD=contraseña
```

Instancias Oracle locales:

Si sólo accede a la instancia Oracle local, no especifique el nombre de instancia-oracle-remota como parte del ID de usuario de inicio de sesión, como en el ejemplo siguiente:

```
%DEFINE LOGIN=ID_usuario  
%DEFINE PASSWORD=contraseña
```

Live Connection:

Si utiliza Live Connection, puede especificar LOGIN y PASSWORD en el archivo de configuración de Live Connection, aunque no se recomienda hacerlo por razones de seguridad. Por ejemplo:

```
CLIETTE DTW_ORA:{  
MIN_PROCESS=1  
MAX_PROCESS=3  
EXEC_NAME=./dtwora8  
DATABASE=not_used  
LOGIN=idusuario@nombre_instancia_oracle_remota  
PASSWORD=contraseña  
}
```

Consejo: No especifique la variable DATABASE para Oracle.

4. Pruebe la configuración ejecutando un script de shell CGI para asegurarse de que se puede acceder a la instancia de Oracle desde el servidor Web, como en el ejemplo siguiente:

```
#!/bin/sh  
echo "content-type; text/html  
echo  
echo "<html><pre>"  
set  
echo "</pre><p>&nbsp;</p><pre>"  
tnsping nombre-instancia-oracle  
echo
```

Alternativamente, puede ejecutar *tnsping* directamente desde una macro Net.Data, como en el ejemplo siguiente:

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

Resolución de problemas:

Si falla el paso de verificación, compruebe que todos los pasos anteriores hayan sido satisfactorios verificando los elementos siguientes:

- Compruebe la configuración de Oracle.
- Verifique que la sintaxis de las variables de entorno de Oracle sea correcta y que no falte ninguna variable.
- Compruebe la conexión Oracle, asegurándose de que ha entrado el ID de usuario y la contraseña correctos.

Si el paso de verificación aún falla, póngase en contacto con el Servicio de IBM.

Ejemplo:

Después de haber realizado los pasos de verificación de acceso, puede efectuar llamadas al entorno de lenguaje de Oracle con funciones de la macro, como en el ejemplo siguiente:

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

Configuración de Live Connection

Live Connection gestiona las conexiones de bases de datos y aplicaciones Java para mejorar el rendimiento de Net.Data en los sistemas operativos Windows NT, OS/2, AIX y Sun Solaris. Mediante el uso de un Connection Manager y de cliettes, procesos que mantienen conexiones abiertas, Live Connection elimina la actividad general de arranque de la conexión a una base de datos o de inicio de una Java Virtual Machine.

Live Connection utiliza un archivo de configuración, dtwcm.cnf, para determinar qué cliettes es necesario iniciar. Contiene información de administración y definiciones para cada una de las cliettes utilizadas con Live Connection. Consulte el apartado “Gestión de conexiones” en la página 210 para obtener más información sobre Live Connection.

El archivo de configuración de ejemplo mostrado en la Figura 6 en la página 37 contiene los tipos siguientes de información:

- Información de puertos de Connection Manager
- Información de cliettes de SQL para una conexión DB2
- Información de cliettes de aplicaciones Java


```

1 CONNECTION_MANAGER{
2   MAIN_PORT=7100
3 }
4
5 CLIETTE DTW_SQL:CELDIAL{
6   MIN_PROCESS=1
7   MAX_PROCESS=5
8   EXEC_NAME=./dtwcdb2
9   DATABASE=CELDIAL
10  LOGIN=marshall
11  PASSWORD=st1pwd
12 }
13
14 CLIETTE DTW_JAVAPPS{
15   MIN_PROCESS=1
16   MAX_PROCESS=5
17   EXEC_NAME=./javaapp
18 }

```

- Las líneas 1 a 3 son necesarias para el archivo de configuración y definen números de puerto exclusivos utilizados con Live Connection.
- Las líneas 5 a 12 definen todas las cliettes de base de datos, identificando el nombre de cliette, el número de procesos a ejecutar, el nombre de base de datos y el archivo ejecutable (exec) de cliette. Puede incluir información adicional, por ejemplo un ID de usuario y una contraseña para conectarse a una base de datos DB2.
- Las líneas 14 a 18 definen todas las cliettes para las aplicaciones Java, identificando el nombre de cliette, el número de procesos a ejecutar, los números de puerto exclusivos y el archivo ejecutable de cliette.

Figura 6. Archivo de configuración de Live Connection

Antes de empezar: Lea la sección de sugerencias y consejos a continuación de estos pasos antes de personalizar el archivo de configuración de Live Connection.

Configuración de puertos de Live Connection:

El valor que elija para MAIN_PORT es el número de puerto que se utilizará en primer lugar. Los números de puerto que Live Connection puede utilizar pueden calcularse utilizando el valor de MAIN_PORT y MAX_PROCESSES de cada cliette. Cuando se carga, Live Connection asigna puertos que empiezan en el número especificado en MAIN_PORT y aumentan hasta que se alcanza el valor de MIN_PROCESSES acumulativo. Entonces cargará puertos, según sea necesario, hasta que se alcance el valor de MAX_PROCESSES. El número máximo de puertos utilizados es la suma de los valores MAX_PROCESSES.

Por ejemplo, en la configuración de la Figura 6, los números de puerto asignados serán 7100, 7101 y 7102 y luego hasta el 7110 según sea necesario.

Importante:

- Consulte con el administrador del sistema para asegurarse de que los números de puerto que piensa utilizar están disponibles.

- Asegúrese de que el valor de MAIN_PORT coincide con el valor de DTW_CM_PORT en el archivo de inicialización de Net.Data.

Configuración de cliettes de base de datos:

1. Escriba la sentencia de entorno de cliette.

CLLETTE tipo:nombre_bd

Parámetros:

tipo Nombre que asocia un entorno de lenguaje con una cliette. Consulte una lista de tipos válidos en la página 62.

nombre_bd

Nombre de cliette de base de datos, que es normalmente el mismo que el de la base de datos con la que está asociada la cliette, por ejemplo MYDATABASE; sin embargo, el *nombre_bd* también puede ser otro nombre. *nombre_bd* es opcional cuando se utiliza el entorno de lenguaje Oracle.

2. Determine los valores para MIN_PROCESS y MAX_PROCESS. MIN_PROCESS especifica el número de procesos que se deben iniciar cuando se inicia Connection Manager. Después, si llegan peticiones simultáneas adicionales, Connection Manager inicia más cliettes, añadiéndolas de una en una según sean necesarias, hasta que se alcanza el valor especificado para MAX_PROCESS.

Escriba las sentencias MIN_PROCESS y MAX_PROCESS:

MIN_PROCESS=núm_mín

MAX_PROCESS=núm_máx

Parámetros:

núm_mín

Número de procesos de cliette que se deben iniciar cuando se inicia Connection Manager. Deberá tener suficientes números de puerto exclusivos disponibles para este número de cliettes.

núm_máx

Número máximo de cliettes que se pueden ejecutar simultáneamente. Deberá tener suficientes números de puerto exclusivos disponibles para este número de cliettes.

3. Especifique el nombre del archivo ejecutable de cliette. Este nombre de archivo se especifica como:

EXEC_NAME=./dtwcidtipobd

Donde *idtipobd* es el identificador de tipo de base de datos. Consulte la Tabla 6 en la página 39 para obtener nombres de archivos ejecutables válidos:

Tabla 6. Nombres de archivos ejecutables de cliette

Descripción de cliette	Tipo de cliette	Nombres		Disponibilidad de plataforma					
		UNIX	Windows NT u OS/2	AIX	NT	OS/2	HP	SUN	PTX
Cliette de proceso DB2	DTW_SQL	dtwcdb2	dtwcdb2.exe	S	S	S	S	S	N
Cliette de proceso ODBC	DTW_ODBC	dtwcodbc	dtwcodbc.exe	S	S	N	N	N	N
Cliette de proceso Oracle	DTW_ORA	dtwcora	dtwcora.exe	S	S	N	N	N	N

4. Especifique el nombre de la base de datos con la que está asociada la cliette:

`DATABASE=nombre_bd`

Donde *nombre_bd* es el nombre de la base de datos con la que está asociada la cliette; por ejemplo, MYDBASE.

5. Opcional: Cambie los valores por omisión para las variables LOGIN y PASSWORD a *USE_DEFAULT para que Net.Data utilice el mismo ID de usuario que ha iniciado el Connection Manager para conectarse a la base de datos DB2. Si especifica estos valores por omisión, se evitará tener que poner esta información en el archivo de configuración. Por ejemplo, sustituya las líneas 14 y 15, en el archivo de configuración de ejemplo de la Figura 6 en la página 37, por estas líneas:

```
LOGIN=*USE_DEFAULT
PASSWORD=*USE_DEFAULT
```

Consejo: Si define varias entradas de cliette en el archivo de configuración, puede especificar diversos inicios de sesión y contraseñas de base de datos para una base de datos determinada.

Configuración de cliettes de aplicación Java:

1. Escriba la sentencia de entorno de cliette:

```
CLIETTE DTW_JAVAPPS
```

2. Determine los valores para MIN_PROCESS y MAX_PROCESS. MIN_PROCESS especifica el número de procesos que se deben iniciar cuando se inicia Connection Manager. Después, si llegan peticiones simultáneas, Connection Manager inicia más cliettes, añadiéndolas de una en una según sea necesario, hasta que se alcanza el valor especificado para MAX_PROCESS.

Escriba las sentencias MIN_PROCESS y MAX_PROCESS.

MIN_PROCESS=núm_mín

MAX_PROCESS=núm_máx

Parámetros:

núm_mín

Número de procesos de cliente iniciados cuando se inicia Connection Manager. Deberá tener suficientes números de puerto exclusivos disponibles para este número de clientes.

núm_máx

Número máximo de clientes adicionales que se pueden ejecutar simultáneamente. Deberá tener suficientes números de puerto exclusivos disponibles para este número de clientes.

Sugerencias y consejos para configurar Live Connection:

- Connection Manager utiliza los nombres de cliente para identificar de forma exclusiva un conjunto de clientes.
- Para las clientes de base de datos, deberá tener un conjunto de clientes con nombre para cada base de datos a la que piensa acceder. Para bases de datos a las que se accede raras veces, puede establecer el número MIN y MAX de clientes en 1. Alternativamente, también puede establecer MIN en 0, lo que significa que los procesos no se inician hasta que se realiza una petición de Net.Data para la cliente.
- El nombre (NAME) de la cliente debe ser coherente con el nombre de cliente al que se hace referencia en la sentencia ENVIRONMENT para el tipo de cliente del archivo de inicialización. El nombre de cliente puede contener variables y, en el caso de clientes de base de datos, deberá incluir la referencia de variable \$(DATABASE). El valor por omisión para el nombre de cliente en la sentencia ENVIRONMENT es DTW_SQL:\$(DATABASE). Puede utilizar una referencia de variable en el archivo de inicialización, pero no en el archivo de configuración de Live Connection.

La variable DATABASE se define en la macro de Net.Data. Cuando se encuentra una sentencia de SQL en la macro, la referencia de variable \$(DATABASE) del archivo de inicialización de Net.Data se sustituye por el valor actual de DATABASE.

Puede utilizar este método para acceder a varias bases de datos. Si tiene tres bases de datos a las que desea acceder en la macro de Net.Data (por ejemplo, D1, D2 y D3) y el archivo de inicialización tiene la línea estándar CLIETTE "DTW_SQL:\$(DATABASE)", necesitará tres secciones en el archivo de configuración de Live Connection tales como:

```
CLIETTE DTW_SQL:D1{ ...}  
CLIETTE DTW_SQL:D2{....}  
CLIETTE DTW_SQL:D3{....}
```

- Los procesos se inician pero no se detienen. Si establece el número máximo de procesos en M y en cualquier momento se utilizan simultáneamente M procesos, éstos permanecerán activos hasta que concluya el Connection Manager; por consiguiente, no es aconsejable que el valor de MAX_PROCESS sea tan alto como para que se agoten todos los recursos del sistema iniciando procesos que se utilizan raras veces.

Recomendación: Intente utilizar valores diferentes para MIN_PROCESS y MAX_PROCESS para ver qué funciona mejor para el sistema. Si el Connection Manager recibe un número de peticiones mayor que el valor máximo especificado, la última petición se pone en cola hasta que termina de procesarse una cliette. Cuando queda disponible una cliette, entonces se procesa la petición en cola. Este proceso de poner en cola las peticiones es transparente para el usuario de la aplicación.

- Puede utilizar el mismo tipo de cliette para diferentes secciones con nombre. Por ejemplo, todas las secciones de base de datos DB2 del archivo de configuración utilizan el mismo tipo de cliette. No puede tener dos secciones con el mismo nombre.

Si está utilizando CGI y desea que sólo algunas bases de datos utilicen Live Connection, simplemente liste las bases de datos que desea en el archivo de configuración. Cuando Net.Data está procesando una macro de Net.Data y encuentra una función de SQL, solicita al Connection Manager una cliette específica. Si el Connection Manager no tiene dicho tipo de cliette, responde con un mensaje NO_CLIETTE_AVAIL. En lugar de ello, Net.Data procesa la petición con una versión de DLL.

Configuración del servicio Connection Manager para que se inicie automáticamente:

En Windows NT, puede especificar que Connection Manager se inicie como un servicio de Windows NT, en lugar de hacerlo desde la línea de mandatos. La ejecución de Connection Manager como un servicio de Windows NT permite a Connection Manager iniciarse automáticamente cada vez que se inicia la máquina.

Importante: Inicie Connection Manager desde la línea de mandatos antes de configurarlo para que se inicie automáticamente a fin de asegurarse de que el archivo de configuración de Live Connection es correcto.

- En la barra de tareas de Windows NT, seleccione **Inicio->Configuración->Panel de control ->Servicios**.
- Seleccione **Net.Data Connection Manager** y, a continuación, pulse el botón **Inicio**.
- Seleccione **Tipo de inicio automático** y, a continuación, pulse **Aceptar**.

Configuración del servidor Web para utilizarlo con CGI

La CGI (Common Gateway Interface - Interfaz de pasarela común) es una interfaz estándar de la industria que permite a un servidor Web invocar un programa de aplicación como, por ejemplo, Net.Data. El soporte de Net.Data para CGI le permite utilizar Net.Data con el servidor Web favorito.

Configure el servidor Web para invocar Net.Data añadiendo las directivas Map, Exec y Pass al archivo de configuración HTTP de forma que se invoque Net.Data.

Recomendación: Organice las directivas en el orden siguiente en el archivo de configuración HTTP para evitar que se ignoren las directivas: Map, Exec, Pass. Por ejemplo, si la directiva Pass siguiente precede a una directiva Map o Exec, se ignoran las directivas Map y Exec:

Pass /*

Directivas Map

Las directivas Map correlacionan las entradas utilizando el formato /cgi-bin/db2www/* en la biblioteca donde reside el programa Net.Data en el sistema. (El asterisco (*) al final de la serie hace referencia a cualquier elemento que siga a la serie). Se incluyen las sentencias de correlación de mayúsculas y minúsculas, porque las directivas son sensibles a las mayúsculas y minúsculas.

Directivas Exec

La directiva Exec permite al servidor Web ejecutar cualquier programa CGI de la biblioteca CGI. Especifique en la directiva la biblioteca donde reside el programa (no el programa propiamente dicho).

Configuración de Net.Data para FastCGI

Net.Data puede ejecutarse como un proceso FastCGI en Apache Web Server y Domino Go Webserver. FastCGI proporciona un rendimiento similar a los demás programas API de Web con el aislamiento de aplicación de CGI. FastCGI se soporta en los sistemas operativos AIX y Sun Solaris.

Antes de utilizar FastCGI, asegúrese de que ha instalado Apache Web Server 1.2.0 o posterior o Domino Go Webserver 4.6.2.5 o posterior.

Para configurar Net.Data para FastCGI:

1. Configure el archivo de configuración de FastCGI y del servidor Web para el sistema operativo:

Para Apache Web Server:

Actualice el archivo http.conf.

- Declare la nueva aplicación:

```
AppClass dir_inst
-processes núm_proc
-initial-env LIBPATH=víaacceso_bibl
-initial-env ORACLE_HOME=víaacceso_oracle
-initial-env ORACLE_SID=instancia_oracle
-initial-env DB2INSTANCE=instancia_db2
-initial-env RXQUEUE_OWNER_PID=var_rend_REXX
-initial-env LANG=entorno_nacional
```

- Declare el módulo FastCGI:

```
<location /fcgi-bin>
SetHandler fastcgi-script
</location>
```

Para Domino Go Webserver:

Actualice los archivos httpd.conf y fcgi.conf:

- En el archivo httpd.conf, declare la sección de servicio:

```
ServerInit /u/mydir/http/fcgi-bin/fcgi.o:FCGIInit
/u/mydir/http/fcgi.conf service/fcgi-bin/*
/u/mydir/http/fcgi-bin/fcgi.o:FCGIDispatcher*ServerTerm
/u/mydir/http/fcgi-bin/fcgi.o:FCGIStop
```

- En el archivo fcgi.conf, declare la aplicación:

```
Local {
Exec dir_inst
Role Responder
URL /fcgi-bin/db2www
BindPath /tmp/db2www.ibm
NumProcesses núm_proc
Environ LIBPATH=víaacceso_bibl
Environ ORACLE_HOME=víaacceso_oracle
Environ ORACLE_SID=instancia_oracle
Environ DB2INSTANCE=instancia_db2
Environ RXQUEUE_OWNER_PID=var_rend_REXX
Environ LANG=entorno_nacional
Environ NLSPATH=víaacceso_catálogo_msj
Environ MAXREQUEST=núm_peticiones
}
```

Parámetros:

dir_inst

Vía de acceso y nombre de directorio para los archivos ejecutables de Net.Data.

Para Apache:

```
AppClass /u/mydir/apache/fcgi-bin/db2www
```

Para Domino Go Webserver:

```
Exec /u/mydir/http/fcgi-bin/db2www
```

Role Responder

Palabra clave necesaria para Domino Go Webserver solamente.

URL Palabra clave necesaria y dirección de URL para Domino Go Webserver solamente. El URL apunta a la vía de acceso especificada para la sentencia EXEC_PATH.

BindPath

Palabra clave necesaria y sentencia de vía de acceso para Domino Go Webserver en AIX solamente. Vía de acceso del socket UNIX exclusivo utilizado por Net.Data y FastCGI.

num_proc

Número de peticiones que se pueden manejar simultáneamente. El valor por omisión es 1, pero deberá aumentarse para mejorar el rendimiento, basándose en los requisitos de la aplicación. Consulte el apartado “Utilización de FastCGI” en la página 210 para obtener información de ajuste.

Para Apache:

`-processes 7`

Para ICS o Domino Go Webserver:

`NumProcesses 7`

víaacceso_bibl

Sentencias LIBPATH (biblioteca compartida o DLL) declaradas en cada sentencia ENVIRONMENT del archivo de inicialización de Net.Data. Cuando se accede a DB2, la sentencia LIBPATH debe contener la vía de acceso al directorio de bibliotecas de DB2. Por ejemplo:

`/usr/lpp/db2_05_00/lib`

Para Apache:

`-initial-env LIBPATH=/u/mydir/apache/lib:/u/mydir/apache/usr/lib`

Para Domino Go Webserver:

`Environ LIBPATH=/u/mydir/http/lib:/u/mydir/http/usr/lib`

víaacceso_oracle

Necesario cuando se utiliza Oracle. Vía de acceso y directorio de los archivos ejecutables de base de datos de Oracle.

Para Apache:

`-initial-env ORACLE_HOME=/home.native/oracle/product/7.2`

Para Domino Go Webserver:

`Environ ORACLE_HOME=/home.native/oracle/product/7.2`

instancia_oracle

Necesario cuando se utiliza Oracle. Instancia de la base de datos Oracle. Deberá utilizar Live Connection para Oracle.

Para Apache:

-initial-env ORACLE_SID=mvpdb2

Para Domino Go Webserver:

Environ ORACLE_SID=mvpdb2

instancia_db2

Necesario cuando se utiliza DB2. Instancia de la base de datos DB2.

Para Apache:

-initial-env DB2INSTANCE=wwwinst

Para Domino Go Webserver:

Environ DB2INSTANCE=wwwinst

var_rend_REXX

Necesario cuando se utiliza REXX en AIX. La variable de rendimiento se utiliza con FastCGI y REXX en el sistema operativo AIX. El valor por omisión es 0. Para otros productos y sistemas operativos, declare esta variable en la macro Net.Data. Consulte el “Apéndice B. Net.Data para AIX” en la página 257 para obtener más información sobre esta variable.

Para Apache:

-initial-env RXQUEUE_OWNER_PID=0

Para Domino Go Webserver:

Environ RXQUEUE_OWNER_PID=0

entorno_nacional

Variable de entorno nacional de UNIX. Utilice En_US para el inglés americano.

Para Apache:

-initial-env LANG=En_US

Para Domino Go Webserver:

Environ LANG=En_US

NLSPATH

Especifica la ubicación de directorio del catálogo de mensajes.

Para Apache:

-initial-env NLSPATH=/usr/lib/nls/msg/%L/%N

Para Domino Go Webserver:

Environ NLSPATH=/usr/lib/nls/msg/%L/%N

MAXREQUEST

Especifica el número de peticiones que el proceso Fast-CGI de Net.Data atenderá antes de que el servidor Web recicle el proceso e inicie uno nuevo.

Para Apache:

-initial-env MAXREQUEST=5000

Para Domino Go Webserver:

Environ MAXREQUEST=5000

2. **Para Apache:** Añada el directorio fgi-bin como un alias de script nuevo en el archivo srm.conf: `ScripAlias /fcgi-bin/ /u/mydir/apache/fci-bin`
3. Migre los hiperenlaces de las páginas Web generadas estática o dinámicamente de CGI-BIN a FCGI-BIN. Por ejemplo:

```
<a href="http://servidor/fcgi-bin/db2www/nombreachivo.ext/bloque/  
[?name=val&...]">cualquier texto</a>
```
4. Modifique la documentación de usuario final para las invocaciones de URL de Net.Data con FCGI-BIN en lugar de CGI-BIN. Por ejemplo:
`http://servidor/fcgi-bin/db2www/nombreachivo.ext/bloque/[?name=val&...]`

Configuración de Net.Data para utilizarlo con servlets Java y beans Java

Consulte la documentación del servidor Web para obtener instrucciones sobre cómo registrar y utilizar las servlets. Las servlets de Net.Data están contenidas en el archivo `NetDataServlets.jar`. El servidor Web necesitará que añada `dir_inst/servlet-lib/NetDataServlets.jar` y `dir_inst/servlet-lib` a `CLASSPATH`.

Para obtener más información sobre la instalación del servidor Web y sobre las directivas de archivos de configuración del servidor Web, consulte la documentación del servidor Web.

Configuración de Net.Data para utilizarlo con las API del servidor Web

La utilización de una interfaz de programación de aplicaciones (API) de servidor Web en lugar de la CGI puede mejorar el rendimiento de Net.Data considerablemente. Net.Data soporta las API de servidor siguientes:

- API de Lotus Domino Go Webserver (GWAPI)
- API de Internet Server (ISAPI) de Microsoft
- API de Netscape (NSAPI)

Para obtener más información sobre cada API, consulte el apartado “Utilización de las API del servidor Web” en la página 209 y el archivo README de la versión correspondiente de Net.Data.

Requisito: Para ejecutar Net.Data en modalidad GWAPI, ISAPI o NSAPI, deberá reconfigurar el servidor Web para utilizar las DLL o las bibliotecas compartidas de Net.Data como directivas de servicio. Después de realizar la reconfiguración, deberá reiniciar el servidor Web para que entren en vigor los cambios efectuados en el archivo de inicialización de Net.Data. Por omisión, Net.Data se ejecuta en modalidad CGI.

Las secciones siguientes describen cómo configurar Net.Data y el servidor Web para ejecutar la modalidad de API del servidor Web. Los pasos y ejemplos generales que se proporcionan pueden ser diferentes para su sistema operativo. Consulte el archivo README de Net.Data para el sistema operativo correspondiente para obtener instrucciones específicas.

Para configurar GWAPI:

1. Detenga el servidor Web.
2. Asegúrese de que la biblioteca compartida o la DLL GWAPI esté en el directorio CGI-BIN o ICAPI-LIB del servidor.
Consulte el directorio de programa o el archivo README de Net.Data para el sistema operativo correspondiente a fin de conocer los nombres de directorios y archivos específicos.
3. Añada una sentencia de servicio (service) al archivo de configuración del servidor Web (httpd.conf o httpd.cnf) para llamar a la API.

Por ejemplo:

```
Service /cgi-bin/db2www* /home/http/cgi-bin/dtwicapi.o:dtw_icipi*
```

Consulte el archivo README de Net.Data para su sistema operativo para conocer los nombres de directorios y archivos específicos.

4. Reinicie el servidor Web.

GWAPI tiene compatibilidad completa para soportar las aplicaciones existentes. Utilice los mismos métodos que utiliza para CGI para invocar un URL, un formulario o un enlace con GWAPI. Cualquier macro que se ejecute satisfactoriamente utilizando CGI se ejecutará satisfactoriamente utilizando GWAPI. No es necesario realizar modificaciones en estas macros.

Para configurar ISAPI:

1. Detenga el servidor Web.
2. Copie la DLL para ISAPI que viene con Net.Data en el subdirectorio del servidor. Por ejemplo:

```
/inetsrv/scripts/dtwisapi.tipoarchivo
```

Donde *tipoarchivo* es .dll para Windows NT y OS/2 y .o para los sistemas operativos UNIX.

Consulte el archivo README de Net.Data para su sistema operativo para conocer los nombres de directorios y archivos específicos.

3. Dado que ISAPI ignora el proceso de CGI, no necesita tener la parte cgi-bin/db2www/ del URL en los formularios y los enlaces. En su lugar, utilice dtwisapi.*tipoarchivo*. Por ejemplo, si el URL siguiente invoca Net.Data como el programa CGI:

`http://server1.stl.ibm.com/cgi-bin/db2www/test1.d2w/report`

Deberá invocar Net.Data como el plug-in ISAPI con el URL siguiente:

`http://server1.stl.ibm.com/scripts/dtwisapi.dll/test1.d2w/report`

4. Si ha almacenado la macro test1.d2w en el subdirectorio /order/ bajo uno de los directorios especificados en MACRO_PATH o el directorio actual del servidor Web, invoque Net.Data en modalidad CGI utilizando el URL siguiente:

`http://server1.stl.ibm.com/cgi-bin/db2www/orders/test1.d2w/report`

El URL equivalente para invocar Net.Data en modalidad ISAPI es:

`http://server1.stl.ibm.com/scripts/dtwisapi.dll/orders/test1.d2w/report`

5. Reinicie el servidor Web.

Para configurar NSAPI:

1. Detenga el servidor Web.
2. Copie la DLL para NSAPI que viene con Net.Data en el directorio de servidor. Por ejemplo:

`/netscape/server/bin/httpd/dtwnsapi.tipoarchivo`

Donde *tipoarchivo* es .dll para Window NT y OS/2 y .o para los sistemas operativos UNIX.

Consulte el archivo README de Net.Data para su sistema operativo para conocer los nombres de directorios y archivos específicos.

3. Modifique el archivo de configuración de servidor con los cambios listados más abajo. Consulte el directorio de programa o el archivo README de Net.Data para el sistema operativo correspondiente para conocer las diferencias entre sistemas operativos.

obj.conf

Añada al principio del archivo:

```
Init fn="load-modules" shlib="<víaacceso>dtwnsapi.dll" funcs=dtw_nsapi
```

obj.conf	Añada a la directiva de servicios (Service): <pre>Service fn="dtw_nsapi" method=(GET HEAD POST) type="magnus-internal/d2w"</pre>
mime.types	Añada este tipo, donde <i>d2w</i> es la extensión por omisión de la macro. Puede especificar cualquier combinación de tres caracteres. <pre>type=magnus-internal/d2w exts=d2w</pre>

4. Mueva las macros de Net.Data del directorio `netdata/macro` al directorio raíz de documentos del servidor:
`/netscape/server/docs/`
5. Añada el directorio raíz de documentos del servidor a la sentencia `MACRO_PATH`, en el archivo de inicialización. Este cambio indica a Net.Data dónde debe buscar las macros.
6. Dado que NSAPI ignora el proceso de CGI, no necesita tener la parte `cgi-bin/db2www/` del URL en los formularios y los enlaces. El servidor sabe que los archivos con un tipo de archivo `d2w` son macros de Net.Data porque lo ha definido al cambiar los archivos de configuración de Netscape. Por ejemplo, el URL siguiente invoca Net.Data como el programa CGI:
`http://server1.stl.ibm.com/cgi-bin/db2www/test1.d2w/report`

mientras que el URL siguiente invoca Net.Data como el plug-in NSAPI:
`http://server1.stl.ibm.com/test1.d2w/report`

7. Reinicie el servidor Web.

Si mantiene las macros de Net.Data en varios directorios, los tres últimos pasos cambian:

1. Mueva los directorios con las macros de Net.Data que contienen al directorio raíz de documentos del servidor.
2. Actualice la variable `MACRO_PATH` en el archivo de inicialización para incluir todos los directorios y subdirectorios donde están ubicadas las macros.
3. Modifique los enlaces y los formularios que apuntan a estas macros de Net.Data, manteniendo los nombres de directorio. Por ejemplo, al ejecutar en modalidad CGI, el URL siguiente llama a una macro de Net.Data que se almacena en el directorio `/orders/`:
`http://server1.stl.ibm.com/cgi-bin/db2www/orders/test1.d2w/report`

El URL actualizado utilizado para invocar Net.Data en modalidad NSAPI es más corto, pero mantiene el nombre de directorio:
`http://server1.stl.ibm.com/orders/test1.d2w/report`

Configuración de Net.Data con la herramienta de administración de Net.Data

La herramienta de administración de Net.Data le ayuda a configurar y gestionar el archivo de inicialización de Net.Data (DB2WWW.INI) y el archivo de configuración para Live Connection (dtwcm.cnf) en los sistemas operativos Windows NT, AIX y OS/2. Mediante el uso de esta herramienta, puede realizar las tareas siguientes:

- “Inicio de la herramienta de administración” en la página 51
- “Configuración de las sentencias de vía de acceso” en la página 51
- “Configuración de puertos” en la página 53
- “Configuración de cliettes” en la página 54
- “Configuración de entornos de lenguaje” en la página 59
- “Definición de variables de configuración” en la página 63

Consulte el apartado “Antes de empezar” para aprender a configurar la herramienta de administración y para asegurarse de que tiene los prerequisites correctos de software.

Antes de empezar

1. Planifique la configuración de los entornos de lenguaje, las bases de datos, las cliettes, los puertos y las variables de configuración de Net.Data.
2. Instale Net.Data desde el CD-ROM.
3. Instale las bibliotecas de ejecución de Java (JDK 1.1 y las versiones subsiguientes para cada sistema operativo). Consulte el archivo README de Net.Data para el sistema operativo correspondiente para obtener más información.

Asegúrese de que tiene `classes.zip` en `CLASSPATH` después de instalar JDK.

4. Si ha instalado el controlador IBM JDBC que se envía con DB2 Universal Database, añada el directorio de controlador a la sentencia `CLASSPATH` de Java para habilitar la prueba de inicio de sesión de DB2.
5. Vaya al directorio donde se almacena el programa de la herramienta de administración de Net.Data:

Para OS/2 y Windows NT:

El directorio `dir_inst\connect\directorio_admin`, donde `dir_inst` es el directorio que ha especificado para Net.Data durante la instalación y `directorio_admin` es el directorio donde están los archivos de la herramienta de administración.

Para AIX:

El directorio `/usr/lpp/internet/db2www/db2.v2/directorio_admin`, donde `directorio_admin` es el directorio donde están los archivos de la herramienta de administración

Inicio de la herramienta de administración

El sistema operativo que se utilice determina el modo en que se inicia la herramienta de administración.

Para OS/2 y Windows NT:

En la carpeta de IBM Net.Data Versión 2, seleccione el icono **Net.Data Admin Tool**.

Para AIX:

Vaya al directorio de instalación de Net.Data (dir_inst). Desde la línea de mandatos, entre ndadmin para iniciar la herramienta.

La herramienta de administración arranca y se visualiza el cuaderno Net.Data Administration.

Configuración de las sentencias de vía de acceso

Utilice la página **Path** para añadir, modificar o suprimir las sentencias de vía de acceso a fin de localizar los archivos que Net.Data necesita para procesar las macros de Net.Data. Estas sentencias se describen en el apartado “Sentencias de configuración de vía de acceso” en la página 22. La Figura 7 en la página 52 muestra la página **Path**.

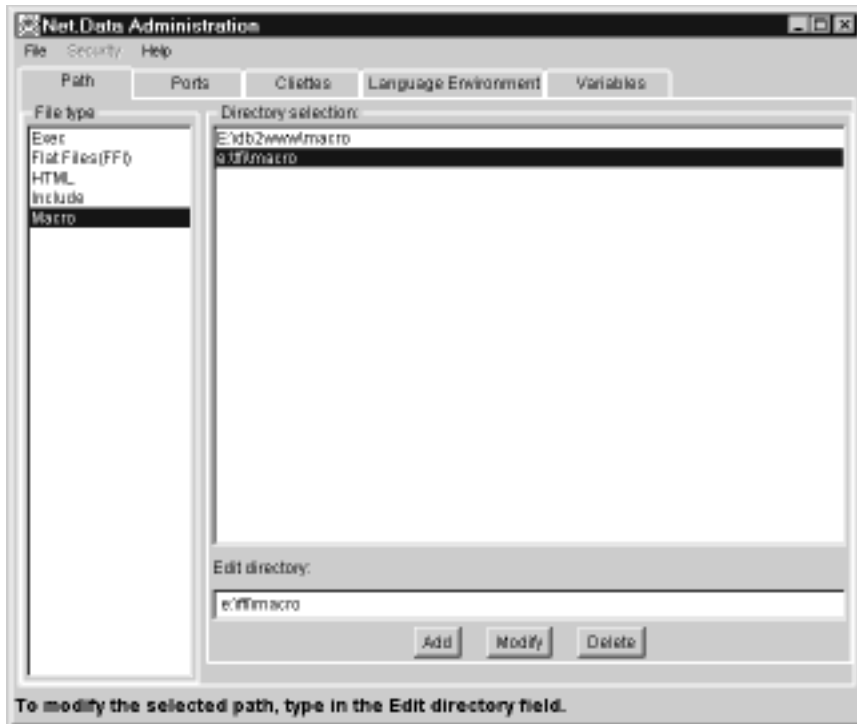


Figura 7. Página Path de la herramienta de administración de Net.Data. Utilice esta página para añadir, modificar o suprimir sentencias de vía de acceso.

Consejo respecto a la configuración: El tipo de archivo HTML sólo puede tener una vía de acceso.

Para añadir una sentencia de vía de acceso:

1. Inicie la herramienta de administración.
2. En la página **Path**, seleccione un tipo de archivo en **File type**, por ejemplo, seleccione Exec.
3. En el campo **Edit directory**, escriba la nueva vía de acceso y pulse el pulsador **Add**.

Si la vía de acceso que ha especificado no existe, se abrirá una ventana de aviso. Si no se selecciona ningún directorio, se añadirá el nuevo directorio como el último elemento de la lista.

4. Cierre la herramienta de administración o pulse otra pestaña para realizar tareas de configuración adicionales.

Para modificar una sentencia de vía de acceso:

1. Inicie la herramienta de administración.

2. En la página **Path**, seleccione el tipo de archivo que desea cambiar en la lista **File type**.
3. Seleccione la vía de acceso que desea modificar en la lista **Directory selection**. La vía de acceso seleccionada se abre en el campo **Edit directory**.
4. Modifique la vía de acceso en el campo **Edit directory** y pulse el pulsador **Modify**. Si la vía de acceso que ha entrado no existe, se abrirá una ventana de aviso.
5. Cierre la herramienta de administración o pulse otra pestaña para realizar tareas de configuración adicionales.

Para suprimir una sentencia de vía de acceso:

1. Inicie la herramienta de administración.
2. En la página **Path**, seleccione en la lista **File type** el tipo de archivo que desea suprimir.
3. En el campo **Directory selection**, seleccione la vía de acceso que desea suprimir. La vía de acceso seleccionada se abre en el campo **Edit directory**.
4. Pulse el pulsador **Delete**.
5. Cierre la herramienta de administración o pulse otra pestaña para realizar tareas de configuración adicionales.

Configuración de puertos

Utilice la página **Ports** para especificar los números de puerto TCP/IP utilizados por Net.Data. La Figura 8 en la página 54 muestra la página **Ports**.



Figura 8. Página Ports de la herramienta de administración de Net.Data. Utilice esta página para especificar puertos.

Para especificar números de puerto TCP/IP:

1. Inicie la herramienta de administración.
2. En la página **Ports**, escriba un número de puerto exclusivo en cada uno de los campos de puerto. La herramienta de administración verifica el número de puerto que se escribe en cada campo cuando se utiliza el tabulador para ir al campo siguiente.
3. Cierre la herramienta de administración o pulse otra pestaña para realizar tareas de configuración adicionales.

Configuración de cliettes

Utilice la página **Cliettes** para añadir, modificar o suprimir cliettes de base de datos de Live Connection, y también puede utilizarla para gestionar los ID de usuario y las contraseñas de base de datos y de administrador para las cliettes de base de datos. En el apartado “Gestión de conexiones” en la página 210, se proporciona más información acerca de las cliettes. La Figura 9 en la página 55 muestra la página **Cliettes**.

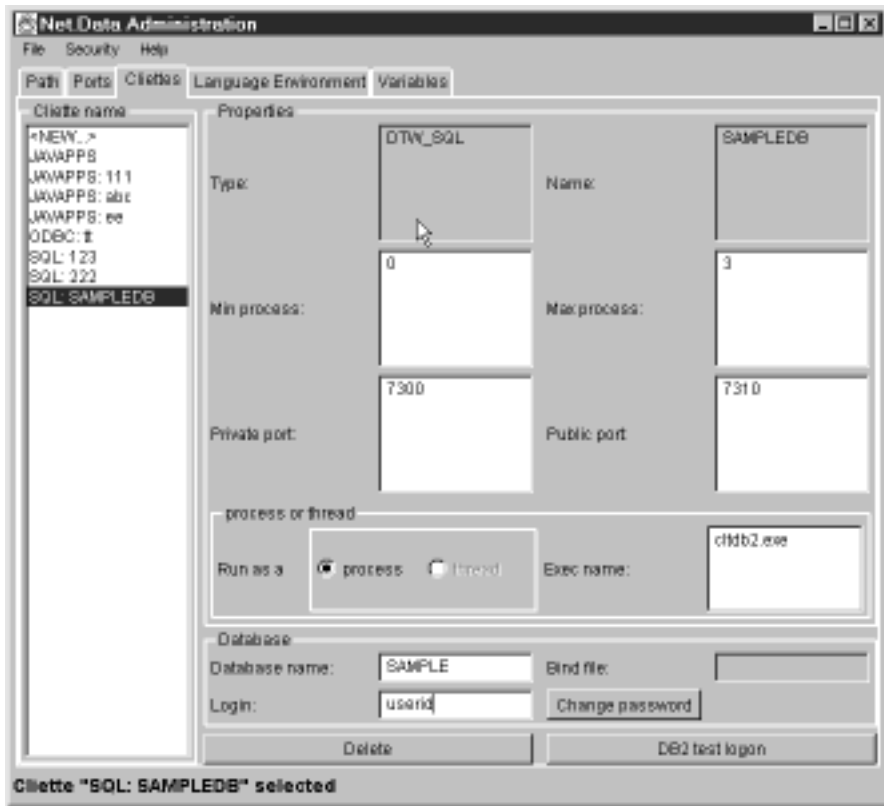


Figura 9. Página Cliettes de la herramienta de administración de Net.Data. Utilice esta página para añadir, modificar o suprimir cliettes.

Para añadir una cliette:

1. Inicie la herramienta de administración.
2. En la página **Cliettes**, seleccione **<new...>** en la lista **Cliette name**. Se abre la ventana **Add a cliette**.



Figura 10. Ventana Add a Cliente de la herramienta de administración de Net.Data. Utilice esta página para añadir clientes.

- Si ha habilitado el cifrado, se le solicitará la contraseña de cifrado la primera vez que cree o modifique una cliente. Esta contraseña se guarda y no tendrá que volver a entrarla nunca.
3. Seleccione un tipo de cliente en la lista **Type**.
4. Escriba un nombre para la nueva cliente en el campo **Name**. El nombre puede ser el de la base de datos u otro nombre de cliente exclusivo. Por ejemplo: MYCLIENTE.
5. Escriba la contraseña de cifrado si el campo **Encryption password** está habilitado. No necesitará escribir la contraseña otra vez porque la herramienta de administración la guarda.
6. Pulse el pulsador **Add**.
La nueva cliente se crea y se añade al final de la lista de clientes. Adicionalmente, el nuevo nombre queda resaltado y se visualizan las propiedades por omisión para la cliente en el recuadro de grupo **Properties**. Puede cambiar estos valores para que se ajusten a la configuración.
7. Cierre la herramienta de administración o pulse otra pestaña para realizar tareas de configuración adicionales.

Para modificar una cliente:

1. Inicie la herramienta de administración.
2. En la página **Clients**, seleccione el nombre de la cliente que desea cambiar en la lista **Client name**. Se visualizan las propiedades de la cliente en el recuadro de grupo **Properties**.
3. Modifique las propiedades en el recuadro de grupo **Properties**, como sea necesario.
 - a. El campo **Type** visualiza el tipo de cliente que se está definiendo y corresponde a un nombre de tipo de entorno de lenguaje. Net.Data

llena de datos este campo cuando se añade una cliette nueva y se han definido las elecciones en la lista **Cliette type** de la ventana Add a Cliette.

- b. El campo **Name** visualiza el nombre de la cliette, que es generalmente el nombre de la base de datos. Net.Data llena de datos este campo cuando se añade una cliette nueva.
 - c. Escriba el número del proceso de cliette que se puede iniciar cuando se inicia el Connection Manager en el campo **Min process**. Necesita una dirección de puerto exclusiva para cada proceso. Consulte el apartado “Configuración de Live Connection” en la página 36 para obtener más información sobre los valores de MIN Process.
 - d. En el campo **Max process**, escriba el número de procesos de cliette que se pueden ejecutar al mismo tiempo, además de los procesos iniciados al iniciar Connection Manager. Necesita una dirección de puerto exclusiva para cada proceso. Consulte el apartado “Configuración de Live Connection” en la página 36 para obtener más información sobre los valores de MAX Process.
 - e. Escriba un número de puerto exclusivo en el campo **Private port** para especificar el número de puerto inicial para utilizarlo con los procesos de cliette que se inician con el Connection Manager. Se utiliza un número de puerto adicional para cada uno de los procesos especificados por el valor **Min Process**. Por ejemplo, si especifica el número de puerto 7012 para **Private port** y el valor 5 para **Min process**, se utilizan los números de puerto 7012 a 7016 y éstos no deben estar en conflicto con otras asignaciones de puerto del sistema.
 - f. Escriba un número de puerto exclusivo en el campo **Public port** para especificar el número de puerto inicial utilizado con los procesos de cliette que se inician cuando se inician procesos adicionales, hasta el número especificado en el campo **Max process**. Se utiliza un número de puerto adicional para cada uno de los procesos. Por ejemplo, si especifica el número de puerto 7020 para **Public port** y el valor 5 para **Max process**, se utilizan los números de puerto 7020 a 7024 y éstos no deben estar en conflicto con otras asignaciones de puerto del sistema.
 - g. El campo **Exec name** visualiza el nombre del archivo ejecutable de cliette.
4. Si la cliette se está utilizando con una base de datos, modifique los valores para el recuadro de grupo **Database**, como sea necesario:
- a. Especifique el nombre de la base de datos con la que está asociada la cliette en el campo **Database name**, por ejemplo, MYDBASE.
 - b. El campo **Bind file** contiene el nombre y la vía de acceso del archivo de vinculación para el tipo de cliette que está utilizando.
 - c. El campo **Login** especifica el ID de usuario de inicio de sesión utilizado para conectarse a la base de datos.

- d. El pulsador **Change password** abre la ventana Change Database Password. Escriba la contraseña de cifrado y la contraseña nueva, dos veces. Puede cifrar la contraseña de base de datos utilizando las funciones de cifrado especificadas en el menú desplegable **Security**.
5. Seleccione **File** y, a continuación, **Save** para guardar los cambios.
6. Cierre la herramienta de administración o pulse otra pestaña para realizar tareas de configuración adicionales.

Para probar el inicio de sesión y la conexión de base de datos DB2:

1. En la página **Cliettes** de la herramienta de administración, pulse el pulsador **DB2 test logon**. Cuando la prueba se ha completado se abre una ventana de confirmación, que visualiza el estado de la prueba de conexión.
2. Cierre la ventana para continuar configurando o cierre la herramienta de administración.

Para suprimir una cliette:

1. Inicie la herramienta de administración.
2. En la página **Cliettes**, seleccione en la lista **Cliette name** el nombre de la cliette que desea suprimir.
3. Pulse el pulsador **Delete**.
4. Cierre la herramienta de administración o pulse otra pestaña para realizar tareas de configuración adicionales.

Para activar el cifrado de los ID de usuario y de las contraseñas de cliette:

El cifrado proporciona seguridad para las conexiones de base de datos con cliettes. Cuando se activa el cifrado, todas las contraseñas de base de datos del archivo de configuración de Live Connection se cifran y necesitan una contraseña de cifrado para el acceso y el descifrado.

Requisito: Deberá utilizar un archivo de configuración de Live Connection de Net.Data Versión 2 para utilizar el cifrado.

1. **Importante:** Haga una copia de seguridad del archivo de configuración de Live Connection, <víaacceso>dtwcm.cnf. Necesitará este archivo si pierde la contraseña de cifrado o desea descifrar contraseñas de base de datos y necesita restaurar las contraseñas.
2. En la página **Cliettes** de la herramienta de administración, seleccione la opción de menú desplegable **Security** -> **Turn encryption on**. Se abrirá la ventana de confirmación Turn Encryption On.
3. Pulse **Yes** para continuar. Se abrirá la ventana Encryption Password.
4. Escriba la contraseña dos veces para tener autorización para trabajar con cliettes que tienen contraseñas cifradas.

5. Pulse **OK** para definir la nueva contraseña y cifrar todas las contraseñas de base de datos para las cliettes.

Para desactivar el cifrado de los ID de usuario y de las contraseñas de cliette:

1. En la página **Cliettes** de la herramienta de administración, seleccione la opción de menú desplegable **Security** -> **Turn encryption off**. Se abrirá la ventana de confirmación Turn Encryption Off.
2. Pulse **Yes** para continuar. Todas las contraseñas se establecen en *USE_DEFAULT por razones de seguridad. Puede restaurar las contraseñas de la copia de seguridad del archivo de Live Connection, <víaaacceso>dtwcm.cnf.

Para cambiar la contraseña para el cifrado:

1. En la página **Cliettes** de la herramienta de administración, seleccione la opción de menú desplegable **Security** -> **Change Encryption Password**. Se abrirá la ventana de confirmación Change Encryption Password.
2. Pulse **Yes** para continuar. Se abrirá la ventana Change Encryption Password.
3. Escriba la contraseña de cifrado anterior una vez y la contraseña nueva dos veces.
4. Pulse **OK** para cambiar la contraseña de cifrado.

Para cambiar la contraseña de base de datos:

1. En la página **Cliettes** de la herramienta de administración, pulse el pulsador **Change Password**. Se abrirá la ventana Change Database Password.
2. Escriba la contraseña de cifrado una vez y la nueva contraseña de base de datos dos veces.
3. Pulse **OK** para cambiar la contraseña y cerrar la ventana. La contraseña de base de datos modificada estará cifrada si ha activado el cifrado.

Configuración de entornos de lenguaje

Utilice la página **Language Environment** para añadir, modificar o suprimir entornos de lenguaje de Net.Data. Los entornos de lenguaje se describen en el apartado “Sentencias de configuración de entorno” en la página 28. La Figura 11 en la página 60 muestra la página **Language Environment**.

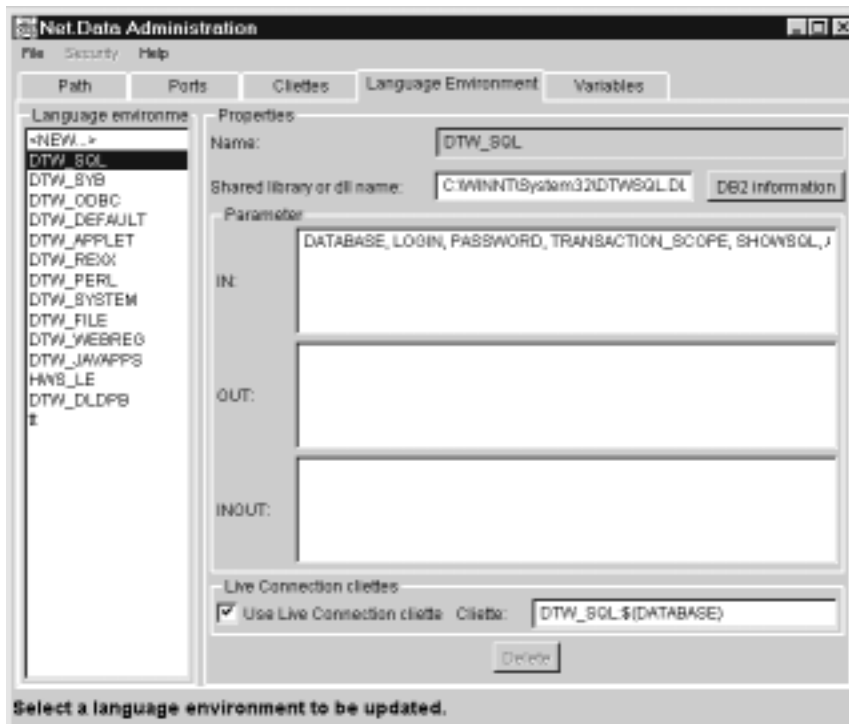


Figura 11. Página Language Environment de la herramienta de administración de Net.Data. Utilice esta página para especificar entornos de lenguaje.

Para añadir un entorno de lenguaje:

1. Inicie la herramienta de administración.
2. En la página **Language Environment**, seleccione <new...> en la lista **Language environment**. Se abrirá la ventana **Add a new language environment**.
3. Escriba el nombre de un entorno de lenguaje en el campo y pulse el pulsador **Add**. Se abrirá la ventana **Add a Language Environment**.



Figura 12. Ventana Add a Language Environment de la herramienta de administración de Net.Data. Utilice esta página para especificar un entorno de lenguaje nuevo.

Se crea el nuevo entorno de lenguaje y se añade su nombre al final de la lista de entornos de lenguaje. Adicionalmente, el nuevo nombre queda resaltado y se visualizan las propiedades por omisión para el entorno de lenguaje en el recuadro de grupo **Properties**. Puede cambiar estos valores para que se ajusten a la configuración.

4. Cierre la herramienta de administración o pulse otra pestaña para realizar tareas de configuración adicionales.

Para modificar un entorno de lenguaje:

1. Inicie la herramienta de administración.
2. En la página **Language Environment**, seleccione en la lista **Language environment** el nombre del entorno de lenguaje que desea cambiar. Se visualizan las propiedades de la cliente en el recuadro de grupo **Properties**.
3. Modifique como sea necesario las propiedades en el recuadro de grupo **Properties**, que se muestra en la Figura 12 en la página 60:
 - a. Especifique el nombre del entorno de lenguaje en el campo **Name**; este nombre corresponde al tipo de entorno de lenguaje utilizado para definir una cliente. Para cambiar este valor, efectúe una doble pulsación en un nombre diferente en la lista **Language environment**. Consulte el apartado “Sentencias de configuración de entorno” en la página 28 para obtener más información sobre los tipos de entorno de lenguaje.
 - b. Especifique el nombre de programa DLL o de biblioteca compartida y la vía de acceso para el entorno de lenguaje en el campo **Shared library or dll name**.
 - c. Seleccione el pulsador **DB2 information** para visualizar la ventana DB2 Information que se muestra en la Figura 13. Especifique los valores para las variables de entorno de DB2:

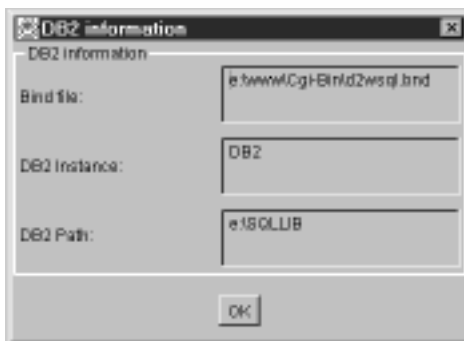


Figura 13. Ventana DB2 Information de la herramienta de administración de Net.Data. Utilice esta página para proporcionar información para bases de datos DB2 específicamente.

- 1) Escriba la vía de acceso y el nombre del archivo de vinculación en el campo **Bind file**.

- 2) Especifique el valor de DB2INSTANCE para la base de datos asociada al utilizar el entorno de lenguaje SQL en el campo **DB2 Instance**.
 - 3) Especifique el nombre de directorio de vía de acceso para los archivos ejecutables de DB2, generalmente \SQLLIB, en el campo **DB2 Path**.
 - 4) Pulse **OK** para guardar los cambios y cerrar la ventana.
- d. Especifique en el recuadro de grupo **Parameters** los parámetros de entrada y salida que se pasan a y desde un entorno de lenguaje cada vez que se llama al entorno de lenguaje.
- Consejo:** No actualice estos campos a no ser que esté definiendo un entorno de lenguaje propio.
- e. Especifique si se deben utilizar cliettes y qué cliette debe asociarse con el entorno de lenguaje en el recuadro de grupo **Live Connection cliettes**.
- 1) Especifique si la cliette para el entorno de lenguaje está activa, seleccionando el recuadro de selección **Use Live Connection cliette**. Seleccione este recuadro de selección si desea utilizar la cliette especificada en el campo **Cliette** al llamar al entorno de lenguaje.
 - 2) Especifique en el campo **Cliette** el nombre de la cliette que debe ejecutarse con el entorno de lenguaje que se está definiendo. La sintaxis del nombre depende de si se está configurando una base de datos o el entorno de lenguaje Java Application. El valor por omisión es DTW_SQL:\$(DATABASE).

Sintaxis para las bases de datos:

tipo:nombre

Donde:

tipo Tipo de entorno de lenguaje para la cliette. Puede ser uno de los valores siguientes:

Para Windows NT:

DTW_ODBC, DTW_ORA, DTW_SQL,
DTW_JAVAPPS

Para OS/2:

DTW_SQL, DTW_JAVAPPS

Para AIX:

DTW_ODBC, DTW_ORA, DTW_SQL,
DTW_JAVAPPS

nombre Nombre de la cliette que se ha definido en la página **Cliette**. El valor por omisión es \$(DATABASE).

Sintaxis para las aplicaciones Java:

DTW_JAVAPPS

4. Seleccione **File** y, a continuación, **Save** para guardar los cambios
5. Cierre la herramienta de administración o pulse otra pestaña para realizar tareas de configuración adicionales.

Para suprimir un entorno de lenguaje:

Restricción: Sólo puede suprimir los entornos de lenguaje creados por los usuarios, no los entornos de lenguaje que vienen con Net.Data.

1. Inicie la herramienta de administración.
2. En la página **Language Environment**, seleccione en la lista **Language environment** el nombre del entorno de lenguaje que desea suprimir.
3. Pulse el pulsador **Delete**.
4. Cierre la herramienta de administración o pulse otra pestaña para realizar tareas de configuración adicionales.

Definición de variables de configuración

Utilice la página **Variables** para especificar el directorio inicial de Net.Data y para seleccionar el nivel de anotación cronológica de mensajes de error. La Figura 14 en la página 64 muestra la página **Variables**.



Figura 14. Página Variables de la herramienta de administración de Net.Data. Utilice esta página para especificar variables de inicialización.

Para especificar el directorio inicial para Net.Data:

Esta variable también se conoce como la variable del directorio de instalación.

1. Inicie la herramienta de administración.
2. En la página **Variables**, escriba en el campo **Installation directory** la vía de acceso para el directorio donde deberá almacenarse el archivo de anotaciones cronológicas. El valor por omisión es `\dir_inst\logs\`. Por ejemplo: `e:\db2www`.
3. Cierre la herramienta de administración o pulse otra pestaña para realizar tareas de configuración adicionales.

Para especificar el nivel de anotación cronológica de mensajes de error para Net.Data:

1. Inicie la herramienta de administración.
2. En la página **Variables**, seleccione un nivel de anotación cronológica de errores en el recuadro de grupo **Error logging**:
 - **off**
 - **errors only**

- **both warnings and errors**
3. Cierre la herramienta de administración o pulse otra pestaña para realizar tareas de configuración adicionales.

Cómo otorgar derechos de acceso a los archivos a los que accede Net.Data

Antes de utilizar Net.Data, necesita asegurarse de que los ID de usuario bajo los que se ejecuta Net.Data tienen los derechos de acceso apropiados a los archivos a los que se hace referencia en una macro de Net.Data y a la macro a la que hace referencia un URL. Esto significa que estos archivos deben estar en directorios o bibliotecas a los cuales se puede conectar el servidor Web para los cuales estos ID de usuario tienen derechos de acceso explícitos.

Más específicamente, asegúrese de que los ID de usuario bajo los que se ejecuta Net.Data están autorizados a realizar las tareas siguientes:

- Leer el archivo de inicialización de Net.Data, `db2www.ini`
- Ejecutar los archivos ejecutables y las DLL de Net.Data y buscar en los directorios de las vías de acceso a los archivos ejecutables y a las DLL
- Leer las macros de Net.Data apropiadas y buscar en los directorios apropiados identificados por la sentencia de configuración de vía de acceso `MACRO_PATH`
- Ejecutar los archivos apropiados y buscar en los directorios apropiados identificados por la sentencia de configuración de vía de acceso `EXEC_PATH`
- Leer los archivos apropiados y buscar en los directorios apropiados identificados por la sentencia de configuración de vía de acceso `INCLUDE_PATH`
- Leer y grabar en los archivos apropiados y buscar en los directorios apropiados identificados por la sentencia de configuración de vía de acceso `FFI_PATH`
- Leer el archivo de configuración de Live Connection, `dtwcm.cnf`
- Leer el archivo de configuración de Cache Manager, `CACHEMGR.CNF`
- Leer archivos ejecutables Perl y REXX externos a los que hacen referencia los entornos de lenguaje

Los métodos para otorgar acceso a estos archivos dependen del sistema operativo en el que se ejecuta Net.Data.

Capítulo 3. Cómo mantener seguros sus activos

La seguridad de Internet se proporciona mediante una combinación de la tecnología de cortafuegos, las características de sistemas operativos, las características de servidor Web, los mecanismos de Net.Data y los mecanismos de control de acceso que forman parte de las fuentes de datos.

Deberá decidir qué nivel de seguridad es apropiado para sus activos. Este capítulo describe métodos que puede utilizar para mantener seguros sus activos y también proporciona referencias a recursos adicionales que puede utilizar para planificar la seguridad del sitio Web.

Las secciones siguientes contienen directrices para proteger sus activos. Los mecanismos de seguridad descritos incluyen:

- “Utilización de cortafuegos”
- “Cifrado de los datos en la red” en la página 70
- “Utilización de la autenticación” en la página 70
- “Utilización de la autorización” en la página 71
- “Utilización de los mecanismos de Net.Data” en la página 71

Utilización de cortafuegos

Los *cortafuegos* son conjuntos de hardware, software y políticas que están diseñados para limitar el acceso a los recursos en un entorno de red.

Los cortafuegos:

- Protegen la red interna de la infiltración o intrusión
- Protegen la red interna de los datos y programas introducidos por los usuarios internos
- Limitan el acceso de usuario interno a los datos externos
- Limitan el daño que se puede hacer si se rompe el cortafuegos

Net.Data puede utilizarse con productos cortafuegos que se ejecuten en el entorno.

Las configuraciones posibles siguientes proporcionan recomendaciones para gestionar la seguridad de la aplicación Net.Data. Estas configuraciones proporcionan información de alto nivel y suponen que se ha configurado un

cortafuegos que aísla la intranet protegida de la Internet pública. Examine cuidadosamente estas configuraciones junto con las políticas de seguridad de su organización:

- **Configuración de alta seguridad**

Esta configuración crea una subred que aísla Net.Data y el servidor Web de la intranet protegida y la Internet pública. El software cortafuegos se utiliza para crear un cortafuegos entre el servidor Web y la Internet pública y otro cortafuegos entre el servidor Web y la intranet protegida, que contiene el Servidor DB2. Esta configuración se muestra en la Figura 15.

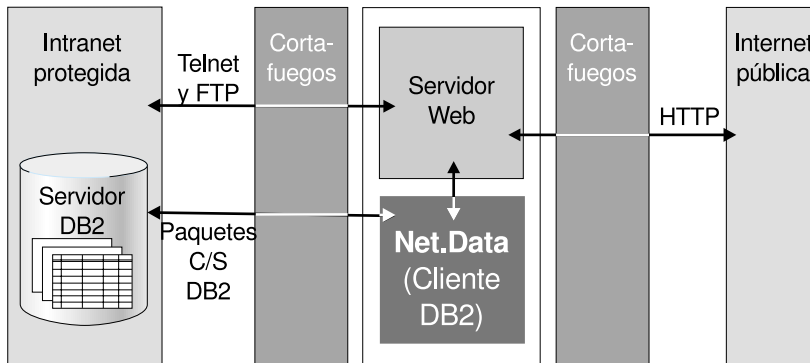


Figura 15. Configuración de alta seguridad

Para definir una configuración:

- Instale Net.Data en la máquina servidor Web y asegúrese de que Net.Data puede acceder al Servidor DB2 dentro de la intranet:
 - Instalando el Client Application Enabler (CAE) en la máquina servidor Web.
 - Configurando el cortafuegos para permitir el tráfico DB2 a través del mismo. Un método consiste en añadir una norma de filtrado de paquetes para permitir las peticiones de cliente DB2 de Net.Data y reconocer los paquetes enviados del Servidor DB2 a Net.Data.
- Permitir el acceso FTP y Telnet entre el servidor Web y la intranet protegida. Un método consiste en instalar un servidor socks en la máquina servidor Web.
- En el archivo de configuración de filtrado de paquetes del software cortafuegos, especifique que los paquetes TCP de entrada del puerto HTTP estándar pueden acceder al servidor Web. Asimismo, especifique que los paquetes de reconocimiento TCP de salida pueden ir a cualquier sistema principal de la Internet pública desde el servidor Web.

- **Configuración de seguridad intermedia**

En esta configuración, el software cortafuegos aísla la intranet protegida con el servidor DB2 de la Internet pública. Net.Data y el servidor Web están fuera del cortafuegos en una plataforma de estación de trabajo. Esta configuración es más simple que la primera, pero ofrece sin embargo protección de base de datos. La Figura 16 muestra esta configuración.

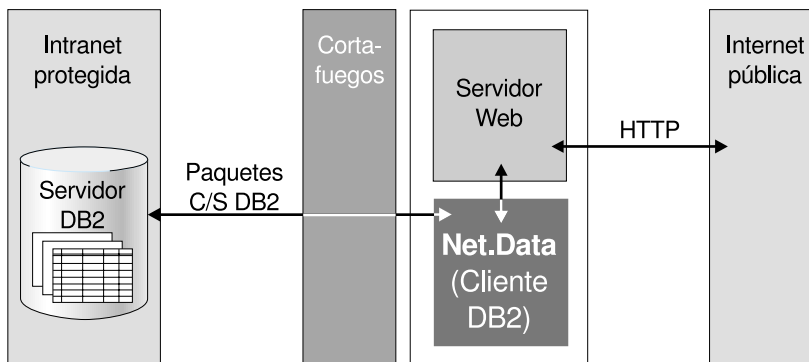


Figura 16. Configuración de seguridad intermedia:

Deberá instalar el CAE en el servidor Web para permitir que Net.Data se comuniquen con el servidor DB2. El cortafuegos debe configurarse para permitir que las peticiones de cliente DB2 fluyan de Net.Data a DB2 y para permitir que los paquetes de reconocimiento fluyan de DB2 a Net.Data.

- **Configuración de baja seguridad**

En esta configuración, el servidor DB2 y Net.Data se instalan fuera del cortafuegos y de la intranet protegida. No están protegidos de los ataques externos. El cortafuegos no necesita normas de filtrado de paquetes para esta configuración. La Figura 17 en la página 70 muestra esta configuración.

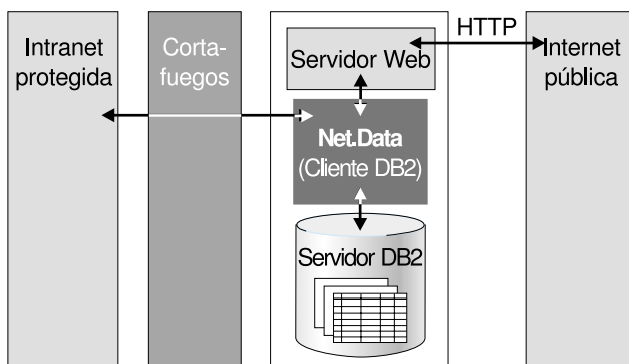


Figura 17. Configuración de baja seguridad:

Cifrado de los datos en la red

Puede cifrar todos los datos que se envían entre un sistema cliente y el servidor Web cuando utilice un servidor Web que soporte SSL (Secured Sockets Layer - Capa de sockets protegida). Esta medida de seguridad soporta el cifrado de los ID de inicio de sesión, de las contraseñas, de todos los datos que se transmiten mediante formularios HTML del sistema cliente al servidor Web y de todos los datos que se envían del servidor Web al sistema cliente. La mayoría de los servidores Web soportan SSL.

Utilización de la autenticación

La *autenticación* se utiliza para asegurar que un ID de usuario que realiza una petición de Net.Data esté autorizado a acceder a los datos y a actualizarlos dentro de la aplicación. La autenticación es el proceso mediante el cual se empareja el ID de usuario con una contraseña para validar que la petición viene de un ID de usuario válido. El servidor Web asocia un ID de usuario con cada petición de Net.Data que procesa. Entonces el proceso o la hebra que está manejando la petición puede acceder a cualquier recurso para el que dicho ID de usuario esté autorizado.

Puede utilizar dos tipos de autenticación: una que protege determinados directorios en el servidor y otra que protege la base de datos.

- La mayoría de los servidores Web permiten especificar los directorios del servidor que hay que proteger. También se puede hacer que el sistema exija un ID de usuario y una contraseña a las personas que accedan a los archivos de los directorios que especifique. Consulte la *Guía del administrador* del servidor Web para determinar las posibilidades del sistema.
- DB2 tiene un sistema de autenticación para acceso de base de datos que puede restringir el acceso a tablas y columnas a determinados usuarios.

Puede utilizar las variables especiales de Net.Data, por ejemplo LOGIN y PASSWORD, para enlazar con la rutina de autenticación de DB2.

Consejo: Para proteger las macros de Net.Data realice lo siguiente:

1. Añada directivas de protección en el archivo de configuración del servidor Web para el objeto de programa Net.Data.
2. Asegúrese de que el ID de usuario bajo el que se ejecutará Net.Data tiene derechos de acceso a las macros. Para obtener información sobre cómo otorgar derechos de acceso, consulte el apartado “Cómo otorgar derechos de acceso a los archivos a los que accede Net.Data” en la página 65.

Utilización de la autorización

La *autorización* proporciona a un usuario el acceso completo o restringido a un objeto, un recurso o una función. Las fuentes de datos tales como DB2 proporcionan sus propios mecanismos de autorización para proteger la información que gestionan. Estos mecanismos de autorización suponen que el ID de usuario asociado con el proceso que está ejecutando la petición de Net.Data se ha autenticado correctamente, como se explica en el apartado “Utilización de la autenticación” en la página 70. Entonces los mecanismos de control de acceso existentes para estas fuentes de datos permiten o niegan el acceso basándose en las autorizaciones que posee el ID de usuario autenticado.

Utilización de los mecanismos de Net.Data

Además de los métodos descritos anteriormente, puede utilizar variables de configuración de Net.Data o técnicas de desarrollo de macros para limitar las actividades de los usuarios finales, ocultar los activos corporativos, tales como el diseño de la base de datos, y validar valores de entrada proporcionados por el usuario en entornos de producción.

Variables de configuración de Net.Data

Net.Data proporciona varias variables de configuración que se pueden utilizar para limitar las actividades de los usuarios finales o para ocultar el diseño de la base de datos.

Controlar el acceso a archivos con sentencias de vía de acceso

Net.Data evalúa los valores de las sentencias de configuración de vía de acceso para determinar la ubicación de los archivos y los programas ejecutables utilizados por las macros de Net.Data. Estas sentencias de vía de acceso identifican uno o más directorios en los que Net.Data busca al intentar localizar macros, archivos ejecutables, archivos de inclusión u otros archivos planos. Mediante la inclusión selectiva de directorios en estas sentencias de vía de acceso, puede

controlarse explícitamente los archivos a los que pueden acceder los usuarios en los navegadores. Consulte el “Capítulo 2. Configuración de Net.Data” en la página 5 para obtener detalles adicionales sobre las sentencias de vía de acceso.

También deberá utilizar la comprobación de autorización como se describe en el apartado “Utilización de la autorización” en la página 71 y verificar que los nombres de archivo no se puedan cambiar en las sentencias INCLUDE como se describe en el apartado “Técnicas de desarrollo de macros” en la página 73.

Inhabilitar SHOWSQL para sistemas de producción

La variable SHOWSQL permite al usuario especificar que Net.Data visualice en un navegador Web las sentencias de SQL especificadas en las funciones de Net.Data. Esta variable se utiliza principalmente para desarrollar y probar el SQL dentro de una aplicación y no está destinada a utilizarse en sistemas de producción.

Puede inhabilitar la visualización de sentencias de SQL en entornos de producción utilizando uno de los métodos siguientes:

- Cuando utilice Net.Data Versión 2.0.7 o posterior, utilice la variable de configuración DTW_SHOWSQL en el archivo de inicialización de Net.Data para alterar temporalmente el efecto del establecimiento de SHOWSQL en las macros de Net.Data. Consulte el apartado “DTW_SHOWSQL: Habilitar o inhabilitar la variable de configuración SHOWSQL” en la página 19 para conocer la sintaxis y obtener información adicional.
- Los usuarios de Net.Data Versión 2.0.5 y anteriores pueden utilizar la función DTW_ASSIGN() como se describe en el apartado “Técnicas de desarrollo de macros” en la página 73.

Consulte SHOWSQL en el capítulo de variables del manual *Net.Data Guía de consulta* para conocer la sintaxis y ver ejemplos de la variable SHOWSQL de Net.Data.

Considerar si es apropiado habilitar la petición directa para entornos de producción

El método de petición directa para invocar Net.Data permite a un usuario especificar la ejecución de una sentencia de SQL o un programa Perl, REXX o C directamente desde un URL. El método de petición de macro permite a los usuarios ejecutar sólo las sentencias y funciones de SQL definidas o llamadas en una macro.

Deberá considerar detenidamente si debe permitir el uso de la petición directa porque puede que ésta proporcione a los usuarios la posibilidad de ejecutar un amplio conjunto de funciones. Cuando

habilite este método de invocación, asegúrese de que el ID de usuario bajo el cual se procesa la petición de Net.Data tiene el nivel de autorización apropiado.

Puede utilizar la variable de configuración DTW_DIRECT_REQUEST para inhabilitar la petición directa. Consulte el apartado “DTW_DIRECT_REQUEST: Habilitar variable de petición directa” en la página 17 para conocer la sintaxis y obtener información adicional.

Técnicas de desarrollo de macros

Net.Data proporciona varios mecanismos que permiten a los usuarios asignar valores a las variables de entrada. Para asegurarse de que las macros se ejecutan del modo esperado, la macro debe validar estas variables de entrada. La base de datos y la aplicación también deben estar diseñadas para limitar el acceso de un usuario a los datos que el usuario está autorizado a ver.

Utilice las técnicas de desarrollo siguientes al escribir las macros de Net.Data. Estas técnicas le ayudarán a asegurarse de que las aplicaciones se ejecutan tal como se había pensado y que el acceso a los datos está limitado a los usuarios debidamente autorizados.

Asegurarse de que las variables de Net.Data no se pueden alterar temporalmente en un URL

El establecimiento de variables de Net.Data por parte de un usuario en un URL altera temporalmente el efecto de las sentencias DEFINE utilizadas para inicializar variables en una macro. Puede que esto modifique el modo en que se ejecuta la macro. Para evitar esta posibilidad, inicialice las variables de Net.Data utilizando la función DTW_ASSIGN().

Ejemplo: En lugar de utilizar %DEFINE SHOWSQL="NO" para establecer la variable SHOWSQL de Net.Data, utilice @DTW_ASSIGN(SHOWSQL, "NO"). Entonces, una asignación de serie de consulta como, por ejemplo, SHOWSQL=YES no altera temporalmente el valor de la macro.

Puede inhabilitar la visualización de sentencias de SQL en entornos de producción utilizando uno de los métodos siguientes:

- Cuando utilice versiones de Net.Data que soportan la variable de configuración DTW_SHOWSQL, utilice esta variable en el archivo de inicialización de Net.Data para alterar temporalmente el efecto del establecimiento de SHOWSQL en las macros de Net.Data. Consulte el apartado “DTW_SHOWSQL: Habilitar o inhabilitar la variable de configuración SHOWSQL” en la página 19 para conocer la sintaxis y obtener información adicional.
- Utilice la función DTW_ASSIGN() como se describe en el ejemplo anterior, para asignar el valor de SHOWSQL a fin de evitar que éste quede alterado temporalmente.

Consulte SHOWSQL en el capítulo de variables del manual *Net.Data Guía de consulta* para conocer la sintaxis y ver ejemplos de la variable SHOWSQL de Net.Data.

También puede utilizar DTW_ASSIGN para asegurarse de que otras variables de Net.Data, por ejemplo RPT_MAX_ROWS o START_ROW_NUM, no se alteran temporalmente. Consulte el capítulo de variables del manual *Net.Data Guía de consulta* para obtener más información acerca de estas variables.

Validar que las sentencias de SQL no se puedan modificar utilizando modos que alteren el comportamiento esperado de la aplicación

La adición de una variable de Net.Data a una sentencia de SQL dentro de una macro permite a los usuarios modificar de forma dinámica la sentencia de SQL antes de ejecutarla. Es responsabilidad de la persona que escribe la macro validar los valores de entrada proporcionados por el usuario, así como asegurar que una sentencia de SQL que contenga una referencia de variable no se esté modificando de un modo inesperado. La aplicación Net.Data deberá validar los valores de entrada proporcionados por el usuario desde el URL para que la aplicación Net.Data pueda rechazar la entrada no válida. El proceso de diseño de validación debe incluir los pasos siguientes:

1. Identificar la sintaxis de la entrada válida; por ejemplo, un ID de cliente debe empezar por una letra y puede contener solamente caracteres alfanuméricos.
2. Determinar qué daño potencial puede producirse si se permite entrada incorrecta, entrada intencionalmente dañina o entrada realizada para obtener acceso a los activos internos de la aplicación de Net.Data.
3. Incluir en la macro sentencias de verificación de entrada que satisfagan las necesidades de la aplicación. Dicha verificación depende de la sintaxis de la entrada y del modo en que ésta se utilice. En los casos más simples, puede ser suficiente con comprobar si hay contenido no válido en la entrada o invocar Net.Data para verificar el tipo de entrada. Si la sintaxis de la entrada es más compleja, puede que el desarrollador de macros tenga que analizar la entrada parcial o totalmente para verificar si es válida.

Ejemplo 1: Utilización de la función de serie DTW_POS() para verificar las sentencias de SQL

```
%FUNCTION(DTW_SQL) query1() {  
    select * from shopper where shlogid = '${shlogid}'  
%}
```

El valor de la variable shlogid está destinado a ser un ID de comprador. Su finalidad es limitar las filas devueltas por la sentencia SELECT a las filas que contienen información acerca del comprador identificado por el ID de comprador. Sin embargo, si se pasa la serie "smith' or shlogid<>'smith" como el valor de la variable shlogid, la consulta se convierte en:

```
select * from shopper where shlogid = 'smith' or shlogid<>'smith'
```

Esta versión modificada por el usuario de la sentencia SELECT original de SQL devuelve la tabla de compradores entera.

Las funciones de serie de Net.Data pueden utilizarse para verificar que el usuario no modifica la sentencia de SQL de un modo inapropiado. Por ejemplo, se puede utilizar la lógica siguiente para asegurar que el valor de entrada asociado con la variable shlogid consta de un solo ID de comprador:

```
@DTW_POS(" ", $(shlogid), result)
%IF (result == "0")
    @query1()
%ELSE
    %{ perform some sort of error processing %}
%ENDIF
```

Ejemplo 2: Utilización de DTW_TRANSLATE()

Suponga que la aplicación necesita validar que el valor proporcionado en la variable de entrada num_orders es un entero. Un modo de llevar a cabo esto consiste en crear una tabla de conversión trans_table que contenga todos los caracteres del teclado excepto los caracteres numéricos 0 a 9 y utilizar las funciones de serie DTW_TRANSLATE y DTW_POS para validar la entrada:

```
@DTW_TRANSLATE(num_orders, "x", trans_table, "x", string_out)

@DTW_POS("x", string_out, result)

    %IF (result = "0")

        %{ continue with normal processing %}

    %ELSE

        %{ perform some sort of error processing %}

    %ENDIF
```

Tenga en cuenta que los usuarios no pueden modificar en los navegadores Web las sentencias de SQL dentro de procedimientos almacenados y que los valores de parámetros de entrada

proporcionados por el usuario están restringidos por los tipos de datos SQL asociados con los parámetros de entrada. En situaciones en las que no resulta práctico validar valores de entrada de usuario utilizando las funciones de serie de Net.Data, puede utilizar procedimientos almacenados.

Asegurarse de que un nombre de archivo en una sentencia INCLUDE no se modifique utilizando modos que alteran el comportamiento esperado de la aplicación

Si especifica el valor para el nombre de archivo con una sentencia INCLUDE utilizando una variable de Net.Data, el archivo que se debe incluir no se determina hasta que se ejecuta el archivo INCLUDE. Si su intención es establecer el valor de esta variable dentro de la macro, pero no permitir que un usuario en el navegador altere temporalmente el valor proporcionado por la macro, deberá establecer el valor de la variable utilizando DTW_ASSIGN en lugar de DEFINE. Si sí tiene la intención de permitir al usuario en un navegador que proporcione un valor para el nombre de archivo, la macro deberá validar el valor proporcionado.

Ejemplo: Una asignación de serie de consulta como filename=".././x" puede producir la inclusión de un archivo de un directorio normalmente no especificado en la sentencia de configuración INCLUDE_PATH. Suponga que el archivo de inicialización de Net.Data contiene la sentencia de configuración de vía de acceso siguiente:

```
INCLUDE_PATH /usr/lpp/netdata/include
```

y que la macro de Net.Data contiene la sentencia INCLUDE siguiente:

```
%INCLUDE "$(filename)"
```

Una asignación de serie de consulta de filename=".././x" incluirá el archivo /usr/lpp/x, lo cual no era la intención de la especificación de la sentencia de configuración INCLUDE_PATH.

Las funciones de serie de Net.Data pueden utilizarse para verificar que el nombre de archivo proporcionado es apropiado para la aplicación. Por ejemplo, se puede utilizar la lógica siguiente para asegurar que el valor de entrada asociado con la variable de nombre de archivo no contiene la serie "../":

```
@DTW_POS("../", $(filename), result)
%IF (result > "0")
  %{ perform some sort of error processing %}
%ELSE
  %{ continue with normal processing %}
%ENDIF
```


Diseñar la base de datos y las consultas para que las peticiones de usuario no tengan acceso a datos sensibles acerca de otros usuarios

Algunos diseños de base de datos reúnen los datos de usuario sensibles en una sola tabla. A menos que las peticiones SELECT de SQL estén calificadas de algún modo, este planteamiento puede hacer que todos los datos sensibles puedan ser utilizados por cualquier usuario en un navegador Web.

Ejemplo: La sentencia siguiente de SQL devuelve información de un pedido identificado por la variable `order_rn`:

```
select setsstatcode, setsfailtype, mestname
      from merchant, setstatus
     where merfnbr   = setsmenbr
        and setsornbr = $(order_rn)
```

Este método permite a los usuarios en un navegador Web especificar números de pedido aleatorios y posiblemente obtener información sensible acerca de los pedidos de otros clientes. Un modo de evitar este tipo de riesgo consiste en realizar los cambios siguientes:

- Añada una columna a la tabla de información de pedido que identifique al cliente asociado con la información de pedido dentro de una fila específica.
- Modifique la sentencia SELECT de SQL para asegurarse de que SELECT está calificado por un ID de cliente autenticado proporcionado por el usuario en el navegador.

Por ejemplo, si `shlogid` es la columna que contiene el ID de cliente asociado con el pedido y `SESSION_ID` es una variable de `Net.Data` que contiene el ID autenticado del usuario en el navegador, puede sustituir la sentencia SELECT anterior por la sentencia siguiente:

```
select setsstatcode, setsfailtype, mestname
      from merchant, setstatus
     where merfnbr   = setsmenbr
        and setsornbr = $(order_rn)
        and shlogid  = $(SESSION_ID)
```

Utilizar variables ocultas de Net.Data

Puede utilizar variables ocultas de `Net.Data` para esconder diversas características de la macro de `Net.Data` de los usuarios que ven la fuente HTML con su navegador Web. Por ejemplo, puede ocultar la estructura interna de la base de datos. Consulte el apartado “Variables ocultas” en la página 127 para obtener más información sobre las variables ocultas.

Solicitar información de validación de un usuario

Puede crear su propio esquema de protección basándose en la entrada proporcionada por el usuario. Por ejemplo, puede solicitar información de validación de un usuario mediante un formulario

HTML y validarla utilizando datos que la macro de Net.Data recupera de una base de datos o llamando a un programa externo desde una función definida en la macro de Net.Data.

Para obtener más información sobre cómo proteger sus activos, consulte la lista de seguridad de Internet de preguntas frecuentes (FAQ) en este sitio Web:

<http://www.w3.org/Security/Faq>

Capítulo 4. Invocación de Net.Data

Este capítulo describe cómo invocar Net.Data utilizando las diversas interfaces de servidor Web. Para poder utilizar uno de los métodos de invocación, se deberá configurar primero Net.Data para la interfaz específica. Puede configurar Net.Data para utilizar las interfaces de servidor Web siguientes:

- CGI (Common Gateway Interface)
- FastCGI
- GWAPI (Lotus Domino Go Webserver)
- NSAPI (Netscape Server)
- ISAPI (Internet Server de Microsoft)
- Servlets Java

Consulte el “Capítulo 2. Configuración de Net.Data” en la página 5 para obtener más información sobre la configuración de Net.Data para estas interfaces. Por omisión, el servidor Web invoca Net.Data como un programa CGI, con cada petición de Net.Data ejecutándose en un proceso nuevo e independiente. Cuando configure el servidor Web, determine el modo en que se invocará Net.Data.

Las secciones siguientes describen los tipos de peticiones que Net.Data acepta y los métodos que se pueden utilizar para invocar Net.Data utilizando las diversas API y servlets.

- “Tipos de peticiones de invocación”
- “Invocación de Net.Data mediante las API de servidor Web” en la página 92
- “Invocación de Net.Data con servlets Java y JavaBeans” en la página 95

Tipos de peticiones de invocación

Independientemente del método con el que invoque Net.Data, existen dos tipos de peticiones que se pueden especificar.

Petición de macro

Especifica que Net.Data ejecute la macro especificada.

Petición directa

Especifica que Net.Data ejecute una sentencia de SQL, un procedimiento almacenado o una función.

Los desarrolladores de Web que deseen escribir una sola consulta de SQL o llamar a una sola función, por ejemplo un procedimiento almacenado de DB2, un programa REXX o una función Perl, pueden emitir una petición directa a

la base de datos. Una petición directa no tiene ninguna lógica de aplicación de Net.Data compleja que requiera una macro de Net.Data y, por consiguiente, ignora el procesador de macros de Net.Data. Los parámetros de la petición directa se pasan al entorno de lenguaje apropiado para procesarlos a fin de obtener una mejora de rendimiento.

La Figura 18 ilustra las diferencias entre una petición de macro y una petición directa. Una petición de macro especifica siempre una macro dentro del URL para la petición y también puede utilizar datos de formulario. Una petición directa no especifica nunca una macro dentro del URL, pero sin embargo puede utilizar datos de formulario.

La sintaxis para invocar Net.Data depende del modo en que se haya

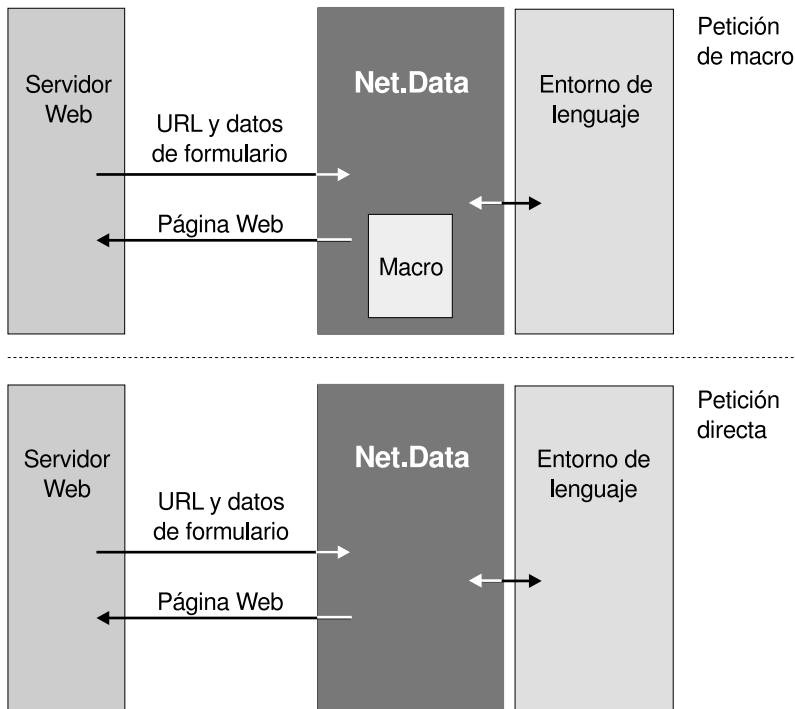


Figura 18. Petición de macro frente a petición directa

configurado Net.Data y del tipo de petición que se realice. Net.Data se invoca utilizando un URL, tanto para las peticiones de macro como para las peticiones directas. El URL puede entrarlo directamente el usuario o puede codificarse en la página HTML como un enlace HTML o un formulario HTML. El servidor Web invoca Net.Data utilizando CGI, FastCGI o una de las API de servidor Web. Adicionalmente, puede invocar Net.Data utilizando servlets Net.Data.

Para las peticiones de macro, especifique dentro del URL el nombre de la macro de Net.Data y el nombre del bloque HTML que se debe ejecutar en la macro de Net.Data. Para las peticiones directas, especifique dentro del URL el nombre del entorno de lenguaje Net.Data, la sentencia de SQL o el nombre de la función y los valores de parámetros adicionales necesarios. Especifique estos valores utilizando una sintaxis definida por Net.Data.

Las secciones siguientes describen estas peticiones de invocación de forma más detallada:

- “Invocación de Net.Data con una macro (Petición de macro)”
- “Invocación de Net.Data sin ninguna macro (petición directa)” en la página 86

Aunque los ejemplos especifican la sintaxis a utilizar al invocar Net.Data utilizando CGI, los conceptos se aplican a todas las interfaces que se utilizan para invocar Net.Data. Para conocer la sintaxis exacta necesaria para cada tipo de interfaz, consulte la sección específica de cada una.

- “Invocación de Net.Data mediante las API de servidor Web” en la página 92
- “Invocación de Net.Data con servlets Java y JavaBeans” en la página 95

Invocación de Net.Data con una macro (Petición de macro)

Un navegador de cliente invoca Net.Data enviando una petición en forma de URL. Esta sección muestra cómo invocar Net.Data especificando una macro en la petición de URL.

La petición enviada a Net.Data tiene el formato siguiente:

`http://servidor/víaaacceso_invocación_Net.Data/nombreachivo/bloque[?nombre=val&...]`

Parámetros:

servidor

Especifica el nombre y la vía de acceso del servidor Web. Si el servidor es el servidor local, puede omitir el nombre de servidor y utilizar un URL relativo.

víaaacceso_invocación_Net.Data

Vía de acceso y nombre de archivo del ejecutable, la clase de servlet, la DLL o la biblioteca compartida de Net.Data. Por ejemplo, `/cgi-bin/db2www/`.

nombreachivo

Especifica el nombre del archivo de macros de Net.Data. Net.Data busca este nombre de archivo y lo intenta comparar con las sentencias de vía de acceso definidas en la variable de vía de acceso de inicialización `MACRO_PATH`. Consulte el apartado “`MACRO_PATH`” en la página 27 para obtener más información.

bloque Especifica el nombre del bloque HTML en la macro de Net.Data a la que se hace referencia.

?nombre=val&...

Especifica uno o más parámetros opcionales pasados a Net.Data.

Especifique este URL directamente en el navegador. O bien, puede utilizarlo en un enlace HTML o crearlo utilizando un formulario como se muestra a continuación:

- Enlace HTML:

```
<a href="URL">cualquier texto</a>
```

- Formulario HTML:

```
<form method=método ACTION="URL">cualquier texto</form>
```

Parámetros:

método Especifica el método HTML utilizado con el formulario.

URL Especifica el URL utilizado para ejecutar la macro de Net.Data, cuyos parámetros están descritos más arriba.

Ejemplos

Los ejemplos siguientes muestran los diferentes métodos para invocar Net.Data.

Ejemplo 1: Invocación de Net.Data utilizando un enlace HTML:

```
<a href="http://server/cgi-bin/db2www/myMacro.d2w/report">
.
.
.
</a>
```

Ejemplo 2: Invocación de Net.Data utilizando un formulario

```
<form method=post
  action="http://server/cgi-bin/db2www/myMacro.d2w/report">
.
.
.
</form>
```

Las secciones siguientes describen los enlaces y formularios HTML y proporcionan más información sobre cómo invocar Net.Data con ellos:

- “Enlaces HTML” en la página 83
- “Formularios HTML” en la página 83

Enlaces HTML

Si es el autor de una página Web, puede crear un enlace HTML que haga que se ejecute un bloque HTML. Cuando un usuario en el navegador pulse un texto o una imagen que se haya definido como enlace HTML, Net.Data ejecutará el bloque HTML en la macro.

Para crear un enlace HTML, utilice el código de HTML <a>. Decida qué texto o gráfico desea utilizar como hipervínculo a la macro de Net.Data y, a continuación, inclúyalo entre los códigos <a> y . En el atributo HREF del código <a>, especifique la macro y el bloque HTML.

El ejemplo siguiente muestra un enlace que hace que se ejecute una consulta de SQL cuando un usuario selecciona el texto "List all monitors" en un página Web.

```
<a href="http://server/cgi-bin/db2www/listA.d2w/report">
List all monitors</a>
```

Al pulsar el enlace, se llama a una macro llamada listA.d2w, que tiene un bloque HTML llamado "report", como en el ejemplo siguiente:

```
%DEFINE DATABASE="MNS97"
```

```
%FUNCTION(DTW_SQL) myQuery(){
SELECT MODNO, COST, DESCRIP FROM EQPTABLE
WHERE TYPE='MONITOR'
%}
```

```
%HTML(report){
@myQuery()
%}
```

La consulta devuelve una tabla que contiene información sobre el número de modelo, el coste y la descripción para cada monitor que se describe en la tabla EQPTABLE. Este ejemplo visualiza los resultados de la consulta generando un informe por omisión. Consulte el apartado "Bloques de informe" en la página 147 para obtener información sobre cómo puede personalizar los informes utilizando un bloque REPORT.

Formularios HTML

Puede personalizar de forma dinámica la ejecución de las macros de Net.Data utilizando formularios HTML. Los formularios permiten a los usuarios proporcionar valores de entrada que pueden afectar a la ejecución de la macro y al contenido de la página Web que crea Net.Data.

El ejemplo siguiente se basa en el ejemplo de la lista de monitores del apartado "Enlaces HTML" permitiendo a los usuarios en un navegador utilizar un simple formulario HTML para seleccionar el tipo de producto para el que se visualizará información.

```

<h1>Hardware Query Form</h1>
<hr>
<form method=post action="/cgi-bin/db2www/equip1st.d2w/report">
<p>What type of hardware do you want to see?</p>
<menu>
<li><input type="radio" name="hardware" value="mon" checked /> Monitors</li>
<li><input type="radio" name="hardware" value="pnt" /> Pointing devices</li>
<li><input type="radio" name="hardware" value="prt" /> Printers</li>
<li><input type="radio" name="hardware" value="scn" /> Scanners</li>
</menu>

<input type="submit" value="submit" />
</form>

```

Cuando el usuario en el navegador ha realizado una selección y ha pulsado el botón Submit, el servidor Web procesa el parámetro ACTION del código FORM, que invoca Net.Data. Entonces Net.Data ejecuta el bloque de informe HTML de la macro equip1st.d2w:

```

%DEFINE DATABASE="MNS97"

%FUNCTION(DTW_SQL) myQuery(){
SELECT MODNO, COST, DESCRIP FROM EQPTABLE
WHERE TYPE='${hardware}'
%REPORT{
<h3>Here is the list you requested</h3>
%ROW{
<hr>
$(N1): $(V1), $(N2): $(V2)
<p>$(N3): $(V3)
%}
%}
%}

%HTML(report){
@myQuery()
%}

```

En el ejemplo anterior, el valor de TYPE=\${hardware} de la sentencia de SQL se toma de la entrada del formulario HTML. Consulte el manual *Net.Data Guía de consulta* para obtener una descripción detallada de las variables que se utilizan en el bloque ROW.

Un tipo de entrada al que Net.Data da un trato especial es el tipo de entrada FILE. Con este tipo de entrada, los usuarios pueden subir al servidor un archivo, que Net.Data o cualquier otra aplicación del servidor puede procesar adicionalmente.

Net.Data no realiza ninguna conversión en los archivos subidos, que se tratan como datos binarios. Los archivos subidos se almacenan en el directorio

especificado en DTW_UPLOAD_DIR y se les da un nombre exclusivo, determinado utilizando las normas siguientes:

Sintaxis:

NombreArchivoMacro + '.' + *NombreVarForm* + '.' + *IdentifExclusivo* + '.' + *NombreArchivoForm*

NombreArchivoMacro

Nombre de la macro que maneja la petición (a la que se llama en el formulario). Sólo se utiliza el nombre de archivo, no la vía de acceso completa.

NombreVarForm

Nombre de la variable utilizada para identificar el archivo en el formulario.

IdentifExclusivo

Serie utilizada para asegurar la exclusividad. Esta serie se construye de formas diferentes en función de la plataforma de Net.Data. En OS/400, por ejemplo, se utiliza el método siguiente:

AAAAMDDHHMSSCCC + '-' + PID + '-' + TID

donde

AAAA = año
MM = mes
DD = día
HH = hora (de 00 a 23)
SS = segundos
CCC = milisegundos
PID = ID de proceso
TID = ID de hebra

Ejemplo:

Primero, establezca DTW_UPLOAD_DIR en el archivo de inicialización de Net.Data:

DTW_UPLOAD_DIR /home/http/pub/upload

A continuación, cree un formulario que invoque una macro y pase el nombre de archivo y el archivo como un parámetro:

```
<form method="post" enctype="multipart/form-data"
  action="/netdatadev/form.dtw/report">
  <input type="text" name="name" value="john doe" />Enter name
  <input type="text" name="zipno" value="55901" />Enter number
  <input type="file" name="resume" value="myresume.txt" />RÚsumÚ
  <input type="submit" />
</form>
```

Si un usuario tuviera que someter el formulario, aceptando los valores por omisión, el archivo resultante aparecería en el servidor como algo parecido a:
`/home/http/pub/upload/form.dtw.resume.20000313112341275-6245-021.myresume.txt`

Invocación de Net.Data sin ninguna macro (petición directa)

Esta sección muestra cómo invocar Net.Data utilizando la *petición directa*. Cuando utilice la petición directa, no especifique el nombre de una macro en el URL. En lugar de ello, especifique el entorno de lenguaje de Net.Data, la sentencia de SQL o un programa que se deberá ejecutar y los valores de los parámetros adicionales necesarios en el URL, utilizando una sintaxis definida por Net.Data. Consulte el apartado “DTW_DIRECT_REQUEST: Habilitar variable de petición directa” en la página 17 para obtener información sobre cómo habilitar e inhabilitar la petición directa.

La sentencia de SQL o el programa y los demás parámetros especificados se pasan directamente al entorno de lenguaje designado para su proceso. La petición directa mejora el rendimiento porque Net.Data no necesita leer y procesar una macro. Los entornos de lenguaje proporcionados por Net.Data SQL, ODBC, Oracle, Java, System, Perl y REXX soportan la petición directa y se puede llamar a Net.Data utilizando un URL, un formulario HTML o un enlace.

Una petición directa invoca Net.Data pasando los parámetros de la serie de consulta del URL o de los datos del formulario. El ejemplo siguiente ilustra el contexto en el que se especifica una petición directa.

```
<a href="http://servidor/cgi-bin/db2www/?petición_directa">cualquier texto</a>
```

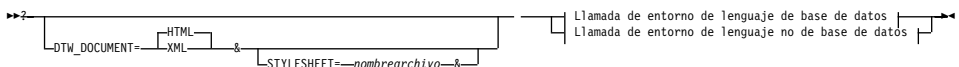
Donde *petición_directa* representa la sintaxis de la petición directa. Por ejemplo, el enlace HTML siguiente contiene la petición directa:

```
<a href="http://servidor/cgi-bin/db2www/?LANGENV=DTW_PERL&FUNC=my_perl(hi)">cualquier texto</a>
```

Sintaxis de la petición directa

La sintaxis para invocar Net.Data con petición directa puede contener una llamada a un entorno de lenguaje de base de datos o a un entorno de lenguaje no de base de datos.

Sintaxis



Llamada de entorno de lenguaje de base de datos:



Parámetros

DTW_DOCUMENT

Especifica el tipo de documento que Net.Data deberá devolver como salida. Los valores permitidos son XML o HTML. Este parámetro es opcional y, si no se especifica, se supone HTML.

DTW_STYLESHEET

Especifica la hoja de estilo que Net.Data debe utilizar al visualizar XML. Este parámetro es opcional y sólo es pertinente cuando DTW_DOCUMENT=XML.

hojaestilo

Especifica el nombre de archivo en el servidor para la hoja de estilo.

Llamada de entorno de lenguaje de base de datos

Especifica una petición directa a Net.Data que invoca un entorno de lenguaje de base de datos.

Entrada de datos de formulario

Parámetros que permiten especificar los valores de las variables de SQL o solicitar formato HTML simple. Consulte el capítulo de variables del manual *Net.Data Guía de consulta* para obtener más información sobre estas variables.

DATABASE

Especifica la base de datos a la que Net.Data debe pasar la petición de SQL. Este parámetro es necesario.

DB_CASE

Especifica las letras (mayúsculas o minúsculas) para las sentencias de SQL.

DTW_HTML_TABLE

Especifica si Net.Data debe devolver una tabla HTML o una tabla de texto preformateado.

DTW_DOCUMENT

Especifica si Net.Data debe visualizar los resultados como XML o HTML. Los valores permitidos son XML o HTML. HTML es el valor por omisión cuando no se especifica la palabra clave.

LOGIN

Especifica el ID de usuario de base de datos.

PASSWORD

Especifica la contraseña de base de datos.

RPT_MAX_ROWS

Especifica el número máximo de filas que una función debe devolver en el informe.

SHOWSQL

Especifica si Net.Data debe ocultar o visualizar la sentencia de SQL que se está ejecutando.

START_ROW_NUM

Especifica el número de la fila donde una función debe iniciar su informe.

variable_definida_usuario

Variables que se pasan a Net.Data y proporcionan información necesaria o afectan al comportamiento de Net.Data. Las variables definidas por el usuario son variables que el usuario define para la aplicación.

VALUE

Especifica el valor de la variable de Net.Data.

LANGENV

Especifica el entorno de lenguaje de destino para la llamada de procedimiento almacenado o la sentencia de SQL. Si el entorno de lenguaje es uno de los entornos de lenguaje de base de datos, también se deberá especificar el nombre de la base de datos.

entlengbd

Nombre del entorno de lenguaje de base de datos:

- DTW_SQL
- DTW_ODBC
- DTW_ORA

SQL

Indica que la petición directa especifica la ejecución de una sentencia de SQL incorporada.

sent_sql

Especifica una serie que contiene cualquier sentencia de SQL válida que pueda ejecutarse utilizando el SQL dinámico.

FUNC

Indica que la petición directa especifica la ejecución de un procedimiento almacenado.

nombre_proc_almacenado

Especifica cualquier nombre válido de procedimiento almacenado de DB2.

tipo_parám

Especifica cualquier tipo de parámetro válido para un procedimiento almacenado de DB2.

nombre_parám

Especifica cualquier nombre de parámetro válido.

valor_parám

Especifica cualquier valor válido de parámetro para un procedimiento almacenado de DB2.

IN Especifica que Net.Data debe utilizar el parámetro para pasar los datos de entrada al procedimiento almacenado.

INOUT

Especifica que Net.Data debe utilizar el parámetro para pasar los datos de entrada al procedimiento almacenado y devolver los datos de salida del entorno de lenguaje.

OUT

Especifica que el entorno de lenguaje debe utilizar el parámetro para devolver los datos de salida del procedimiento almacenado.

Llamada de entorno de lenguaje no de base de datos

Especifica una petición directa a Net.Data que invoca un entorno de lenguaje no de base de datos.

LANGENV

Especifica el entorno de lenguaje de destino para la ejecución de la función.

ent_leng

Especifica el nombre del entorno de lenguaje no de base de datos:

- DTW_PERL
- DTW_REXX
- DTW_SYSTEM

FUNC

Indica que la petición directa especifica la ejecución de un programa.

nombre_programa

Especifica el programa que contiene la función a ejecutar.

valor_parám

Especifica cualquier valor válido de parámetro para la función.

Ejemplos de petición directa

Los ejemplos siguientes muestran los diferentes modos en que puede invocar Net.Data mientras utiliza el método de petición directa.

Enlaces HTML: Los ejemplos siguientes utilizan la petición directa para invocar Net.Data mediante enlaces.

Ejemplo 1: Enlace que invoca el entorno de lenguaje Perl y llama a un script Perl que está en la sentencia de vía de acceso EXEC del archivo de inicialización de Net.Data

```
<a href="http://servidor/cgi-bin/db2www/?LANGENV=DTW_PERL&FUNC=my_perl(hi)">
cualquier texto</a>
```

Ejemplo 2: Enlace que invoca el entorno de lenguaje Perl, como en el ejemplo anterior, pero pasa una serie con valores codificados por el URL para los caracteres de comillas y de espacio

```
<a href="http://servidor/cgi-bin/db2www/?LANGENV=DTW_PERL&FUNC=my_perl
(%22Hello+World%22)">cualquier texto</a>
```

Consejo: Deberá codificar determinados caracteres, por ejemplo los espacios y las comillas, dentro de los URL. En este ejemplo, los caracteres de comillas y los espacios dentro del valor de parámetro deben codificarse como %22 o el carácter +, respectivamente. Puede utilizar la función incorporada DTW_URLESCSEQ para codificar cualquier texto que deba codificarse en un URL. Para obtener más información sobre la función DTW_URLESCSEQ, consulte la descripción de la misma en el manual *Net.Data Guía de consulta*.

Formularios HTML: Los ejemplos siguientes utilizan la petición directa para invocar Net.Data mediante formularios.

Ejemplo 1: Un formulario HTML, que produce la ejecución de una consulta de SQL utilizando el entorno de lenguaje SQL, se conecta a la base de datos CELDIAL y consulta una tabla

```
<form method="post"
  action="http://servidor/cgi-bin/db2www/">
<input type="hidden" name="langenv" value="dtw_sql" />
<input type="hidden" name="database" value="ce1dial" />
  <input type="hidden" name="sql"
    value="select * from table1 where col1=$(inputname)" />
Enter Customer name:
<input type="text" name="inputname" value="john" />
<input type="submit" />
</form>
```

Este ejemplo contiene una sustitución de variable en la sentencia de SQL para hacer que la cláusula WHERE sea dinámica.

URL: Los ejemplos siguientes utilizan la petición directa para invocar Net.Data mediante los URL.

Ejemplo 1: URL que produce la ejecución de una consulta de SQL utilizando el entorno de lenguaje SQL

```
http://servidor/cgi-bin/db2www/?LANGENV=DTW_SQL&DATABASE=CELDIAL  
&SQL=select+++from+customer
```

Ejemplo 2: URL que invoca el entorno de lenguaje Perl y llama a un archivo ejecutable que no está en la sentencia de vía de acceso EXEC del archivo de inicialización de Net.Data

```
http://servidor/cgi-bin/db2www/?LANGENV=DTW_PERL&FUNC=/u/MYDIR/macros/myexec.pl
```

Ejemplo 3: URL que invoca el entorno de lenguaje System y llama a un script Perl externo

```
http://servidor/cgi-bin/db2www/?LANGENV=DTW_SYSTEM&  
FUNC=perl+/u/MYDIR/macros/myexec.pl
```

Ejemplo 4: URL que invoca el entorno de lenguaje REXX, llama a un programa REXX y pasa parámetros al programa

```
http://servidor/cgi-bin/db2www/?LANGENV=DTW_REXX&FUNC=myexec.cmd(parm1,parm2)
```

Ejemplo 5: URL que llama a un procedimiento almacenado y pasa parámetros al entorno de lenguaje SQL

```
http://servidor/cgi-bin/db2www/?LANGENV=DTW_SQL&FUNC=MY_STORED_PROC  
(IN+CHAR(30)+Salaries)&DATABASE=CELDIAL
```

Invocación de Net.Data mediante las API de servidor Web

Net.Data soporta las API de Web de la lista siguiente, en función del sistema operativo:

Plug-in GWAPI

Plug-in de API de Lotus Domino Go Webserver

Plug-in ISAPI

Plug-in de API de Internet Server de Microsoft

Plug-in NSAPI

Plug-in de API de Netscape Server

Consulte el apéndice de consulta de sistemas operativos del manual *Net.Data Guía de consulta* para determinar qué API de servidor Web se soportan para el sistema operativo correspondiente. Consulte el apartado “Configuración de Net.Data para utilizarlo con las API del servidor Web” en la página 46 para aprender a configurar Net.Data y el servidor Web para utilizarlos con las API.

Requisitos:

- Si ejecuta Net.Data en modalidad GWAPI, ISAPI o NSAPI, reinicie el servidor Web para que éste pueda volver a cargar Net.Data y ejecutarlo como un proceso.
- Si realiza cambios en el archivo de inicialización después de que el servidor Web haya invocado Net.Data en modalidad API, deberá reiniciar el servidor Web. Los cambios efectuados en el archivo de inicialización de Net.Data (db2www.ini) no entran en vigor. En modalidad API, Net.Data sólo lee el archivo de inicialización una vez para reducir la degradación del rendimiento.
- Cuando se ejecuta en modalidad API, los entornos de lenguaje Oracle y ODBC necesitan Live Connection.

Para invocar las API de servidor Web:**Para GWAPI:****Sintaxis:**

`http://servidor/CGI-BIN/db2www/nombre_macro/bloque[?nombre=val&...]`

Parámetros:*servidor*

Nombre del servidor.

nombre_macro

Nombre de vía de acceso relativa de la macro bajo el directorio especificado por MACRO_PATH.

bloque

Nombre del bloque HTML o XML en la macro que se debe procesar.

?nombre=val&...

Especifica uno o más parámetros opcionales pasados a Net.Data.

Ejemplo:

`http://myserver/CGI-BIN/db2www/mymacro.d2w/report`

Para ISAPI:**Sintaxis:**

`http://servidor/directorio_raíz_HTML_servidor/nombre_dll/nombre_macro/bloque[?nombre=val&...]`

Parámetros:*servidor*

Nombre del servidor.

directorio_raíz_HTML_servidor

Nombre del directorio raíz HTML del servidor Web.

nombre_dll

Nombre de archivo .dll de ISAPI de Net.Data, dtwisapi.dll.

nombre_macro

Nombre de vía de acceso relativa de la macro bajo el directorio especificado por MACRO_PATH.

bloque

Nombre del bloque HTML o XML en la macro que se debe procesar.

?nombre=val&...

Especifica uno o más parámetros opcionales pasados a Net.Data.

Ejemplo:

http://myserver/scripts/dtwisapi.dll/mymacro.d2w/report

Para NSAPI:

Sintaxis:

http://servidor/nombre_macro/bloque[?nombre=val&...]

Parámetros:

servidor

Nombre del servidor.

nombre_macro

Nombre de vía de acceso relativa de la macro bajo el directorio especificado por MACRO_PATH. La extensión de la macro, por ejemplo .d2w, debe definirse en el archivo de configuración del servidor Web. Consulte el apartado “Configuración de Net.Data para utilizarlo con las API del servidor Web” en la página 46 para obtener más información.

bloque

Nombre del bloque HTML o XML en la macro que se debe procesar.

?nombre=val&...

Especifica uno o más parámetros opcionales pasados a Net.Data.

Ejemplo:

http://myserver/mymacro.d2w/report

Invocación de Net.Data con servlets Java y JavaBeans

Las servlets son clases Java que realizan una función similar a la de los plug-in de API de servidor Web o de los programas CGI. Un servidor Web puede utilizar servlets para crear dinámicamente páginas HTML. Las servlets no tienen su propia interfaz gráfica de usuario, pero sus clases pueden cargarse dinámicamente de forma local o a través de la red. Pueden llamarse utilizando una dirección URL (de forma remota) o puede llamarlas un nombre de clase (de forma local).

Net.Data proporciona servlets que se pueden utilizar para invocar macros de Net.Data, ejecutar funciones de Net.Data o ejecutar sentencias de SQL mediante Net.Data en cualquier sistema operativo habilitado para Java soportado por Net.Data.

Este capítulo describe los conceptos y las tareas para:

“Servlets de Net.Data”

“JavaBeans de Net.Data” en la página 102

Servlets de Net.Data

Net.Data proporciona servlets y plug-in NetObjects Fusion con las funciones de Net.Data que se pueden utilizar en un entorno Java. Con las servlets y los plug-in puede:

- Ejecutar macros de Net.Data desde un URL y como una SSI (Server-Side-Include - Inclusión del extremo del servidor)
- Ejecutar funciones de Net.Data desde un URL y como una SSI
- Utilizar NetObjects Fusion (NOF) para gestionar las macros, proporcionando una mejor integración con la gestión de sitios Web y una interfaz gráfica de usuario fácil de utilizar para desarrollar macros simples. Consulte el “Apéndice E. Utilización de los plug-in de NetObjects Fusion (NOF) con servlets de Net.Data” en la página 267 para obtener información sobre cómo utilizar los plug-in NOF.

Esta sección describe los temas siguientes sobre las servlets:

- “Acerca de las servlets de Net.Data”
- “Ejecución de servlets de Net.Data” en la página 96

Acerca de las servlets de Net.Data

Net.Data proporciona servlets para ayudarle a desarrollar y gestionar macros utilizando el entorno Java. Las servlets son clases Java que realizan una función similar a la de los plug-in de API de servidor Web o de los programas CGI. Las servlets las utiliza un servidor Web habilitado para servlets Java para crear páginas HTML. Las servlets no tienen su propia interfaz gráfica de usuario, pero sus clases pueden cargarse dinámicamente de forma local, o a través de la red, y pueden llamarse utilizando una dirección URL (de forma

remota) o puede llamarlas un nombre de clase (de forma local). Las servlets están disponibles para los sistemas operativos Windows NT y AIX.

Las servlets de Net.Data son un reiniciador basado en Java que ejecutan una petición directa o macro de Net.Data Versión 2 utilizando un archivo DLL nativo. Estas servlets le permiten ejecutar una macro existente (MacroServlet) o una función individual (FunctionServlet). Se necesita Net.Data Versión 2 para utilizar estas servlets. Las servlets de Net.Data pueden ejecutar entornos de lenguaje de base de datos o no de base de datos y pueden ejecutarse con Live Connection.

Las servlets vienen con una API para utilizarse con las aplicaciones. Las API de servlet se documentan con Net.Data. Consulte el archivo `<dir_inst>/servlets/NetDataServlets.jar` para obtener documentación de las API.

Net.Data proporciona dos servlets:

Servlet de macro (com.ibm.netdata.servlets.MacroServlet)

Ejecuta una macro de Net.Data existente.

La utilización de la servlet de macro es similar a la ejecución de una macro de Net.Data mediante la interfaz CGI-BIN, excepto en que la macro se ejecuta mediante una servlet Java. La servlet de macro necesita que esté instalado Net.Data Versión 2 o posterior.

Las ventajas de utilizar la servlet de macro incluyen:

- Las consultas de SQL se ejecutan mediante ODBC utilizando Live Connection Manager para aumentar el rendimiento.
- Puede ejecutar macros mediante las SSI (Server-Side-Include) para incorporar múltiples macros en el archivo HTML.

La servlet de macro también permite el acceso nativo a bases de datos heterogéneas, por ejemplo DB2 y Oracle, así como diversos entornos de lenguaje, por ejemplo Perl, REXX y Java.

Servlet de función (com.ibm.netdata.servlets.FunctionServlet)

Ejecuta la función de Net.Data o la sentencia de SQL mediante una interfaz de servlet, por ejemplo `%FUNCTION DTW_SQL()`. Consulte el apartado “Invocación de Net.Data sin ninguna macro (petición directa)” en la página 86 para obtener más información.

La servlet de función necesita que esté instalado Net.Data Versión 2.

Ejecución de servlets de Net.Data

Las servlets de Net.Data pueden ejecutarse desde un URL o como una SSI en un archivo HTML. Puede utilizar los plug-in NetObjects Fusion para incorporar las servlets de Net.Data en el sitio NOF. Las secciones siguientes describen cómo modificar y ejecutar las servlets escribiendo la sintaxis para la

servlet. Consulte el “Apéndice E. Utilización de los plug-in de NetObjects Fusion (NOF) con servlets de Net.Data” en la página 267 para aprender a modificar y ejecutar las servlets con NetObjects Fusion.

- “Ejecución de la servlet de macro”
- “Ejecución de la servlet de función” en la página 99

Ejecución de la servlet de macro: Desde dentro de un archivo HTML, entre los parámetros de servlet utilizando una de las opciones de sintaxis siguientes:

1. URL:

```
http://myserver/servlet/com.ibm.netdata.servlets.MacroServlet
?MACRO=valor_macro&BLOCK=valor_bloque&parmnn=valornn
```

Por ejemplo:

```
http://myserver/servlet/com.ibm.netdata.servlets.MacroServlet
?MACRO=my_macro&BLOCK=my_block&field1=custno
```

2. SSI:

```
<servlet code="com.ibm.netdata.servlets.MacroServlet">
  <param name="MACRO" value="valor_macro">
  <param name="BLOCK" value="valor_bloque">
  <param name="parmnn" value="valornn">
</servlet>
```

Por ejemplo:

```
<servlet code="com.ibm.netdata.servlets.MacroServlet">
  <param name="MACRO" value="my_macro.d2w">
  <param name="BLOCK" value="report">
  <param name="field1" value="custno">
</servlet>
```

Parámetros:

valor_macro

Vía de acceso totalmente calificada a una macro de Net.Data existente

valor_bloque

Nombre del bloque HTML en la macro de Net.Data especificada que se debe ejecutar; el valor por omisión es report (opcional).

parmnn

Parámetros adicionales que necesita la macro, por ejemplo

```
<param name="field1" ...
```

valornn

Valores adicionales que necesita la macro, por ejemplo

```
... value="custnum"
```

Parámetro HTMLPATH: Si obtiene un mensaje de error que indica que falta un parámetro HTMLPATH, añada el parámetro HTMLPATH al mandato de invocación de servlet:

- URL:

```
http://myserver/servlet/com.ibm.netdata.servlets.MacroServlet
?MACRO=nombre_macro&BLOCK=valor_bloque&htmlpath=víaacceso_html&parmn=valornn
```

Por ejemplo:

```
http://myserver/servlet/com.ibm.netdata.servlets.MacroServlet?MACRO=my_macro
&BLOCK=my_blockhtmlpath=e:\html&field1=custno
```

- SSI:

```
<servlet code="com.ibm.netdata.servlets.MacroServlet">
  <param name="MACRO" value="valor_macro">
  <param name="BLOCK" value="valor_bloque">
  <param name="htmlpath" value="víaacceso_html">
  <param name="parmn" value="valornn">
</servlet>
```

Por ejemplo:

```
<servlet code="com.ibm.netdata.servlets.MacroServlet">
<param name="MACRO" value="my_macro">
<param name="BLOCK" value="my_block">
<param name="htmlpath" value="e:\html">
<param name="field1" value="custno">
</servlet>
```

Parámetro OUTBUFLN: Si los resultados de la macro tienen más de 32 KB, especifique el parámetro OUTBUFLN. Si no se especifican estos parámetros cuando son necesarios, los resultados pueden ser imprevisibles.

- URL:

```
http://myserver/servlet/com.ibm.netdata.servlets.MacroServlet
?MACRO=nombre_macro&BLOCK=valor_bloque
&OUTBUFLN=tamaño_almacint_salida&parmn=valornn
```

Por ejemplo:

```
http://myserver/servlet/com.ibm.netdata.servlets.MacroServlet?MACRO=my_macro
&BLOCK=my_block&OUTBUFLN=48&field1=custno
```

- SSI:

```
<servlet code="com.ibm.netdata.servlets.MacroServlet">
  <param name="MACRO" value="valor_macro">
  <param name="BLOCK" value="valor_bloque">
  <param name="OUTBUFLN" value="tamaño_almacint_salida">
  <param name="parmn" value="valornn">
</servlet>
```

Por ejemplo:

```

<servlet code="com.ibm.netdata.servlets.MacroServlet">
<param name="MACRO" value="my_macro">
<param name="BLOCK" value="my_block">
<param name="OUTBUFLen" value="48">
<param name="field1" value="custno">
</servlet>

```

Ejecución de la servlet de función: La servlet de función puede invocar Net.Data utilizando la petición directa para ejecutar una función (por ejemplo, una función REXX) o una sentencia de SQL. Los parámetros que se especifiquen para la servlet dependerán de si se está ejecutando una función o una sentencia de SQL. Desde dentro de un archivo HTML, entre los parámetros de servlet utilizando una de las opciones de sintaxis siguientes:

1. Para un URL:

- **Para invocar una función:**

```

http://myserver/servlet/com.ibm.netdata.servlets.FunctionServlet
?LANGENV=nombre_ent_leng&FUNC=nombre_programa&parmn=valornn

```

Por ejemplo:

```

http://myserver/servlet/com.ibm.netdata.servlets.FunctionServlet
?LANGENV=DTW_REXX&FUNC=my_rexx&field1=custno

```

- **Para invocar una sentencia de SQL:**

```

http://myserver/servlet/com.ibm.netdata.servlets.FunctionServlet
?LANGENV=nombre_ent_leng_basedatos&SQL=sentencia_SQL
&DATABASE=nombre_basedatos&parmn=valornn

```

Por ejemplo:

```

http://myserver/servlet/com.ibm.netdata.servlets.FunctionServlet
?LANGENV=DTW_SQL&SQL=select+++from+myTable&DATABASE=CELDIAL

```

2. SSI:

- **Para invocar una función:**

```

<servlet code="com.ibm.netdata.servlets.FunctionServlet">
  <param name="LANGENV" value="nombre_ent_leng">
  <param name="FUNC" value="nombre_programa">
  <param name="parmn" value="valornn">
</servlet>

```

Por ejemplo:

```

<servlet code="com.ibm.netdata.servlets.FunctionServlet">
<param name="LANGENV" value="DTW_REXX">
<param name="FUNC" value="myREXX">
<param name="field1" value="custno">
</servlet>

```

- **Para invocar una sentencia de SQL:**

```
<servlet code="com.ibm.netdata.servlets.FunctionServlet">
  <param name="LANGENV" value="nombre_ent_leng">
  <param name="SQL" value="nombre_sent_SQL">
  <param name="DATABASE" value="nombre_basedatos">
  <param name="parmnn" value="valornn">
</servlet>
```

Por ejemplo:

```
<servlet code="com.ibm.netdata.servlets.FunctionServlet">
<param name="LANGENV" value="DTW_SQL">
<param name="SQL" value="select * from employee">
<param name="DATABASE" value="CELDIAL">
</servlet>
```

Parámetros:

nombre_ent_leng

Entorno de lenguaje de Net.Data (por ejemplo DTW_SQL, DTW_REXX) que se está llamando para procesar la función, la sentencia de SQL o el procedimiento almacenado. Este parámetro requiere que se utilice FUNC o SQL.

nombre_programa

Nombre del programa que contiene la función que se debe ejecutar. Por ejemplo, my_rexx, donde my_rexx es el nombre de un ejecutable REXX. Especifique parámetros de entrada para la función con los parámetros *parmnn* y *valornn*.

nombre_basedatos

Nombre de la base de datos asociada con el parámetro DATABASE.

nombre_sent_SQL

Sentencia de SQL o nombre de procedimiento almacenado que accede a una base de datos. Por ejemplo, "select * from employee". Especifique parámetros de entrada para la sentencia de SQL o el procedimiento almacenado con los parámetros *parmnn* y *valornn*.

parmnn

Parámetros adicionales que necesita la macro, por ejemplo <param name="field1"

valornn

Valores adicionales que necesita la macro, por ejemplo ... value="custnum".

Parámetro HTMLPATH: Si obtiene un mensaje de error que indica que falta un parámetro HTMLPATH, añada el parámetro HTMLPATH al mandato de invocación de servlet:

- URL:


```
http://myserver/servlet/com.ibm.netdata.servlets.FunctionServlet
?LANGENV=nombre_ent_leng&FUNC=nombre_programa&htmlpath=víaacceso_html
&parmn=valornn
```

Por ejemplo:

```
http://myserver/servlet/com.ibm.netdata.servlets.FunctionServlet
?LANGENV=DTW_REXX&FUNC=my_rexx&htmlpath=e:\html&field1=custno
```

- SSI:

```
<servlet code="com.ibm.netdata.servlets.FunctionServlet">
  <param name="LANGENV" value="nombre_ent_leng">
  <param name="SQL" value="nombre_sent_SQL">
  <param name="htmlpath" value="víaacceso_html">
  <param name="parmn" value="valornn">
</servlet>
```

Por ejemplo:

```
<servlet code="com.ibm.netdata.servlets.FunctionServlet">
<param name="LANGENV" value="DTW_SQL">
<param name="SQL" value="select * from employee">
<param name="htmlpath" value="e:\html">
<param name="field1" value="custno">
<param name="DATABASE" value="SAMPLE">
</servlet>
```

Donde *víaacceso_html* especifica la vía de acceso al directorio raíz HTML del servidor Web; por ejemplo: `htmlpath=e:\html`.

Parámetros OUTBUFLEN: Si los resultados de la macro tienen más de 32 KB, deberá especificar el parámetro OUTBUFLEN. Si no se especifican estos parámetros cuando son necesarios, los resultados pueden ser imprevisibles.

- URL:

```
http://myserver/servlet/com.ibm.netdata.servlets.FunctionServlet
?LANGENV=nombre_ent_leng&FUNC=nombre_programa&OUTBUFLEN=tamaño_almacint_salida
&parmn=valornn
```

Por ejemplo:

```
http://myserver/servlet/com.ibm.netdata.servlets.FunctionServlet
?LANGENV=DTW_REXX&FUNC=my_rexx&OUTBUFLEN=48K&field1=custno
```

- SSI:

```
<servlet code="com.ibm.netdata.servlets.FunctionServlet">
  <param name="LANGENV" value="nombre_ent_leng">
  <param name="FUNC" value="nombre_programa">
  <param name="OUTBUFLEN" value="tamaño_almacint_salida">
  <param name="parmn" value="valornn">
</servlet>
```

Por ejemplo:

```

<servlet code="com.ibm.netdata.servlets.FunctionServlet">
<param name="LANGENV" value="DTW_REXX">
<param name="FUNC" value="my_rexx">
<param name="OUTBUFLN" value="48K">
<param name="field1" value="custno">
</servlet>

```

JavaBeans de Net.Data

Net.Data proporciona JavaBeans que se pueden utilizar en un entorno Java sin tener un servidor Web en ejecución. Un JavaBean es una interfaz de programación orientada a objetos que le permite crear aplicaciones o bloques de construcción de programas reutilizables. Estos objetos se pueden utilizar en una red en un sistema operativo habilitado para Java.

Mediante el uso de una DLL nativa de Net.Data, el JavaBean invoca Net.Data, llenando de datos el código de retorno y una serie que contiene la salida (resultados) de Net.Data. Puesto que los JavaBeans utilizan una DLL nativa, no tiene que tener un servidor Web en ejecución para utilizar las funciones de Net.Data.

Consejo respecto al diseño: Los resultados devueltos por los JavaBeans de Net.Data son lo que devuelve la macro o la función; en general, se trata de HTML. Considere la posibilidad de pasar los resultados a un JavaBean parecido a HTML que interprete HTML y visualice los resultados.

Con los JavaBeans puede:

- Ejecutar macros de Net.Data
- Ejecutar sentencias de SQL mediante Net.Data

Esta sección describe los temas siguientes sobre los JavaBean:

- “Acerca de los JavaBeans de Net.Data”
- “Configuración y ejecución de los JavaBeans de Net.Data” en la página 103

Acerca de los JavaBeans de Net.Data

Net.Data proporciona JavaBeans para ayudarle a desarrollar y gestionar macros utilizando el entorno Java. Los JavaBeans son objetos Java que proporcionan la interfaz siguiente:

- Cuando se utilizan en un entorno de desarrollo de JavaBean (por ejemplo Lotus BeanMachine), utilice el personalizador proporcionado para conectar los componentes deseados a fin de procesar y visualizar los resultados de una macro o una sentencia de SQL y producir una applet Java.
- Cuando utilice la API, puede usar los JavaBeans para proporcionar la funcionalidad de Net.Data a una aplicación o applet Java propia. La documentación de la API está en *<dir_inst>/beans/NetDataBeans.jar*.

Net.Data proporciona dos tipos de JavaBeans:

JavaBean de macro de Net.Data

Proporciona una interfaz basada en Java para ejecutar una macro existente de Net.Data mediante Net.Data.

JavaBean de SQL de Net.Data

Proporciona una interfaz basada en Java para ejecutar una sentencia de SQL mediante Net.Data.

Los JavaBeans de Net.Data son reiniciadores basados en Java que se ejecutan mediante Net.Data utilizando un archivo DLL nativo. Ambos requieren que se instale Net.Data Versión 2 o posterior y JDK Versión 1.1 o posterior.

Configuración y ejecución de los JavaBeans de Net.Data

Esta sección describe cómo configurar y ejecutar los JavaBeans de Net.Data utilizando una herramienta de desarrollo de JavaBean, por ejemplo Bean Machine. Los pasos para utilizar herramientas de desarrollo son genéricos, de modo que puede utilizar la herramienta de su elección.

- “Bean de macro”
- “Bean de SQL” en la página 104

Bean de macro: El bean de Macro de Net.Data, `com.ibm.netdata.beans.NetDataMacro`, le permite utilizar Java para ejecutar una macro existente. Para utilizar este bean, necesita especificar propiedades de Net.Data para el bean de modo que éste pueda funcionar con la macro.

Para configurar el JavaBean de macro de Net.Data con una herramienta de desarrollo de JavaBean:

1. Añada o importe el archivo `<dir_inst>/beans/NetDataBeans.jar` a la herramienta de desarrollo de JavaBean.
2. Utilizando la interfaz de personalización de la herramienta de desarrollo, establezca las propiedades de entrada siguientes:

Macro Especifica el nombre de la macro existente a ejecutar. Por ejemplo: `MyMacro.mac`

Block Especifica el nombre de la sección de bloque HTML a ejecutar; el valor por omisión es `report`.

HTML path

Especifica la vía de acceso al archivo `db2www.ini` de Net.Data.

Parameters

Especifica el nombre de parámetro y los valores a utilizar al ejecutar la macro.

Sintaxis:

`nombre1=valor1&nombreN=valorN`

Para ejecutar el JavaBean de macro de Net.Data con una herramienta de desarrollo de JavaBean:

1. Seleccione la acción de ejecución (run o execute) proporcionada por la herramienta de desarrollo de JavaBean para ejecutar la macro.
2. Después de que se haya ejecutado la macro, puede hacer referencia a las propiedades de salida siguientes:

RC Especifica el código de retorno devuelto de Net.Data.

Results

Especifica los datos devueltos de la ejecución de la macro de Net.Data.

Bean de SQL: El bean de SQL de Net.Data, com.ibm.netdata.beans.NetDataSQL, le permite utilizar Java para ejecutar una sentencia de SQL mediante Net.Data. Para utilizar este bean, necesita especificar propiedades de Net.Data para el bean de modo que éste pueda funcionar con la macro.

Para configurar el JavaBean de SQL de Net.Data con una herramienta de desarrollo de JavaBean:

1. Añada o importe el archivo NetDataBeans.jar a la herramienta de desarrollo de JavaBean.
2. Utilizando la interfaz de personalización de la herramienta de desarrollo, establezca las propiedades de entrada siguientes:

Language environment

Especifica el entorno de lenguaje a utilizar; el valor por omisión es DTW_SQL.

SQL Especifica la sentencia de SQL a ejecutar; el valor por omisión es `select * from employee.`

DATABASE

Especifica la base de datos a utilizar; el valor por omisión es SAMPLE.

HTML path

Especifica la vía de acceso al archivo db2www.ini de Net.Data.

Parámetros

Especifica el nombre de parámetro y los valores a utilizar al ejecutar la sentencia de SQL.

Sintaxis:

`nombre1=valor1&nombreN=valorN`

Para ejecutar el JavaBean de SQL de Net.Data con una herramienta de desarrollo de JavaBean:

1. Seleccione la acción de ejecución (run o execute) proporcionada por la herramienta de desarrollo de JavaBean para ejecutar la macro.
2. Después de que se haya ejecutado la sentencia de SQL, puede hacer referencia a las propiedades de salida siguientes:

RC Especifica el código de retorno devuelto de Net.Data.

Results

 Especifica los datos devueltos de la sentencia de SQL.

Capítulo 5. Desarrollo de macros de Net.Data

Una macro de Net.Data es un archivo de texto que consta de una serie de construcciones de lenguaje de macro de Net.Data que:

- Especifican el diseño de las páginas Web
- Definen variables y funciones
- Llaman a funciones que están incorporadas en Net.Data o definidas en la macro
- Formatean la salida del proceso y la devuelven al navegador Web para que la visualice

La macro de Net.Data contiene dos partes de organización: la parte de declaración y la parte de presentación, como se muestra en la Figura 19.

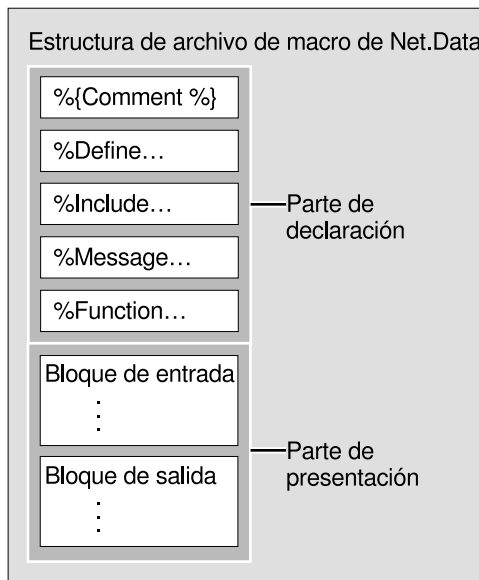


Figura 19. Estructura de macro

- La *parte de declaración* contiene las definiciones de las variables y las funciones de la macro.
- La *parte de presentación* contiene bloques HTML que especifican el diseño de la página Web. Los bloques HTML están compuestos por sentencias de presentación de texto a las que el navegador Web da soporte, por ejemplo HTML y JavaScript.

Puede utilizar estas partes varias veces y en cualquier orden. Consulte el manual *Net.Data Guía de consulta* para conocer la sintaxis de las partes y las construcciones de macro.

Consejo respecto a la autorización: Asegúrese de que el ID de usuario bajo el que se ejecuta Net.Data tenga autorización para leer este archivo. Consulte el apartado “Cómo otorgar derechos de acceso a los archivos a los que accede Net.Data” en la página 65 para obtener más información.

Este capítulo examina los diferentes bloques que componen una macro de Net.Data y los métodos que se pueden utilizar para escribir la macro.

- “Anatomía de una macro de Net.Data”
- “Variables de macro de Net.Data” en la página 118
- “Funciones de Net.Data” en la página 133
- “Generación de marcación de documentos” en la página 145
- “Lógica condicional y repetición en bucle en una macro” en la página 153

Anatomía de una macro de Net.Data

La macro consta de dos partes:

- La parte de declaración, que contiene definiciones utilizadas en la parte de presentación. La parte de declaración utiliza dos bloques principales opcionales:
 - Bloque DEFINE
 - Bloque FUNCTION

La parte de declaración también puede contener otras construcciones de lenguaje y sentencias, por ejemplo sentencias EXEC, bloques IF, sentencias INCLUDE y bloques MESSAGE. Para obtener más información acerca de las construcciones de lenguaje, consulte el capítulo sobre construcciones de lenguaje en el manual *Net.Data Guía de consulta*.

Consejo respecto a la autorización: Asegúrese de que el ID de usuario bajo el que se ejecuta Net.Data tiene autorización para leer y ejecutar los archivos a los que hacen referencia las sentencias EXEC y para leer los archivos a los que hacen referencia las sentencias INCLUDE. Consulte el apartado “Cómo otorgar derechos de acceso a los archivos a los que accede Net.Data” en la página 65 para obtener más información.

- La parte de presentación define el diseño de la página Web, hace referencia a variables y llama a las funciones utilizando bloques HTML que se utilizan como puntos de entrada y salida de la macro. Al invocar Net.Data, especifique un nombre de bloque HTML como punto de entrada para procesar la macro. Los bloques HTML se describen en el apartado “Bloques HTML” en la página 112.

En esta sección, una macro de Net.Data simple ilustra los elementos del lenguaje de macro. Esta macro de ejemplo presenta un formulario que solicita información para pasarla a un programa REXX. La macro pasa esta información a un programa REXX externo llamado ompsamp.cmd, que hace eco de los datos que entra el usuario. Entonces se visualizan los resultados en una segunda página HTML.

En primer lugar, examine la macro entera y, a continuación, cada bloque detalladamente:

```
%{ *****                               DEFINE block                               *****%}
%DEFINE {
    page_title="Net.Data Macro Template"
}%

%{ ***** FUNCTION Definition block *****%}
%FUNCTION(DTW_REXX) rexx1 (IN input) returns(result)
{
    %EXEC{ompsamp.cmd %}
}%

%FUNCTION(DTW_REXX) today () RETURNS(result)
{
    result = date()
}%

%{ *****                               HTML Block: Input                               *****%}
%HTML (INPUT) {
<html>
<head>
<title>$(page_title)</title>
</head><body>
<h1>Input Form</h1>
Today is @today()

<form method="post" action="output">
Type some data to pass to a REXX program:
<input name="input_data" type="text" size="30" />
<p>
<input type="submit" value="enter" />
</p>
</form>

<hr>
<p>[<a href="/">Home page</a>]
</body></html>
}%

%{ *****                               HTML Block: Output                               *****%}
%HTML (OUTPUT) {
<html>
<head>
<title>$(page_title)</title>
</head><body>
```

```

<h1>Output Page</h1>
<p>@rex1(input_data)
<p><hr>
<p>[<a href="/">Home page</a> |
<a href="input">Previous page</a>]
</body></html>
%}

```

La macro de ejemplo consta de cuatro bloques principales: el bloque DEFINE, el bloque FUNCTION y los dos bloques HTML. Pueden haber varios bloques DEFINE, FUNCTION y HTML en una sola macro de Net.Data.

Los dos bloques HTML contienen sentencias de presentación de texto tales como HTML, que facilitan la escritura de macros de Web. Si está familiarizado con HTML, la creación de una macro implica simplemente la adición de sentencias de macro que se deben procesar dinámicamente en el servidor y de sentencias de SQL que se deben enviar a la base de datos.

Aunque la macro tenga un aspecto similar a un documento HTML, el servidor Web accede a ella mediante Net.Data utilizando CGI, una API de servidor Web o una servlet Java. Para invocar una macro, Net.Data necesita dos parámetros: el nombre de la macro a procesar y el bloque HTML de dicha macro a visualizar.

Cuando se invoca la macro, Net.Data la procesa desde el principio. Las secciones siguientes examinan qué sucede mientras Net.Data procesa el archivo.

Bloque DEFINE

El bloque DEFINE contiene la construcción de lenguaje DEFINE y las definiciones de variable utilizadas posteriormente en los bloques HTML. El ejemplo siguiente muestra un bloque DEFINE con una definición de variable:

```

%{ ***** DEFINE Block *****%}
%DEFINE {
    page_title="Net.Data Macro Template"
%}

```

La primera línea es un comentario. Un comentario es texto escrito entre %{ y %}. Los comentarios pueden estar en cualquier parte de la macro. La sentencia siguiente inicia un bloque DEFINE. Se pueden definir múltiples variables en un solo bloque de definición. En este ejemplo, sólo se define una variable, page_title. Una vez definida, se puede hacer referencia a esta variable en cualquier lugar de la macro utilizando la sintaxis, \$(page_title). La utilización de variables facilita posteriormente los cambios globales en la macro. La última línea de este bloque, %}, identifica el final del bloque DEFINE.

Bloque FUNCTION

El bloque FUNCTION contiene declaraciones para las funciones invocadas por los bloques HTML. Las funciones las procesan los entornos de lenguaje y pueden ejecutar programas, consultas de SQL o procedimientos almacenados.

El ejemplo siguiente muestra dos bloques FUNCTION. Uno define una llamada a un programa REXX externo y el otro contiene sentencias REXX incorporadas.

```
%{ ***** FUNCTION Block *****%}
%FUNCTION(DTW_REXX) rexx1 (IN input) returns(result) { <-- Esta función acepta
                                                    un parámetro y devuelve la
                                                    variable 'result', que
                                                    el programa REXX externo
                                                    asigna
%EXEC{ompsamp.cmd %} <-- La función ejecuta un programa REXX externo
                                                    llamado "ompsamp.cmd"
%}

%FUNCTION(DTW_REXX) today () RETURNS(result) {
    result = date() <-- La sentencia fuente individual para este función
                        está incorporada.
%}
```

El primer bloque de función, rexx1, es una declaración de función REXX que, a su vez, ejecuta un programa REXX externo llamado ompsamp.cmd. Esta función acepta una variable de entrada, input, y ésta se pasa automáticamente al mandato REXX externo. El mandato REXX también devuelve una variable llamada result. El contenido de la variable result en el mandato REXX sustituye la llamada de función de invocación @rexx1() contenida en el bloque OUTPUT. El programa REXX puede acceder directamente a las variables input y result, como se muestra en el código fuente para ompsamp.cmd:

```
/* REXX */
result = 'The REXX program received "input" from the macro.'
```

El código de esta función hace eco de los datos que se le pasan. Se puede formatear el texto resultante del modo que se desee escribiendo la llamada de función solicitante @rexx1() entre códigos de estilo de marcación normales (como o). En lugar de utilizar la variable result, el programa REXX podría haber escrito códigos HTML en la salida estándar utilizando sentencias REXX SAY.

El segundo bloque de función, today, también hace referencia a un programa REXX. Sin embargo, en este caso el programa REXX entero está contenido en la propia declaración de función. No se necesita un programa externo. Se permiten programas incorporados para las funciones REXX y Perl porque son lenguajes interpretados que pueden analizarse y ejecutarse dinámicamente. Los programas incorporados tienen la ventaja de ser simples al no necesitar

un archivo de programa independiente para gestionarse. La primera función REXX también se podría haber manejado incorporada.

Bloques HTML

Los bloques HTML definen el diseño de la página Web, las variables de referencia y las funciones de llamada. Los bloques HTML se utilizan como puntos de entrada y salida de la macro. En la petición de macro de Net.Data se especifica siempre un bloque HTML y cada macro debe tener como mínimo un bloque HTML.

El primer bloque HTML de la macro de ejemplo se denomina INPUT. HTML(INPUT) contiene el HTML para un formulario simple con un campo de entrada.

```
%{ ***** HTML Block: Input *****%}
%HTML (INPUT) { <--- Identifica el nombre de este bloque HTML.
<html>
<head>
<title>$(page_title)</title> <--- Observe la sustitución de variable.
</head><body>
<h1>Input Form</h1>
Today is @today() <--- Esta línea contiene una llamada a una función.

<form method="post" action="output"> <--- Cuando se somete este formulario,
                                     se llama al bloque HTML "OUTPUT".<p>
Type some data to pass to a REXX program:
<input name="input_data" <--- "input_data" se define al someter el formulario
TYPE="text" SIZE="30" /> y puede hacerse referencia a ello en cualquier
                           lugar de esta macro. Se inicializa en el valor
                           que escriba el usuario en el campo de entrada.

</p>
<input type="submit" value="enter" />

<hr>
<p>
[
<a href="/">Home page</a></p>
</body><html>
%} <--- Cierra el bloque HTML.
```

El bloque entero está rodeado por el identificador de bloque HTML, %HTML (INPUT) {...%}. INPUT identifica el nombre de este bloque. El nombre puede contener cualquier carácter alfanumérico, subrayados o puntos. El código de HTML <title> contiene un ejemplo de sustitución de variable. El valor de la variable page_title se sustituye por el título del formulario.

Este bloque también tiene una llamada de función. La expresión @today() es una llamada a la función today. Esta función se define en el bloque FUNCTION que se ha descrito anteriormente. Net.Data inserta el resultado de la función today, la fecha actual, en el texto HTML en la misma ubicación donde se encuentra la expresión @today().

El parámetro ACTION de la sentencia FORM proporciona un ejemplo de navegación entre bloques HTML o entre macros. Si se hace referencia al nombre de otro bloque en un parámetro ACTION, se accede a dicho bloque cuando se somete el formulario. Los datos de entrada de un formulario HTML se pasan al bloque como variables implícitas. Esto es cierto para el único campo de entrada definido en este formulario. Cuando se somete el formulario, los datos entrados en este formulario se pasan al bloque HTML (OUTPUT) en la variable *input_data*.

Puede acceder a los bloques HTML de otras macros con una referencia relativa si las macros están en el mismo servidor Web. Por ejemplo, el parámetro ACTION ACTION="../otramacro.d2w/main" accede al bloque HTML llamado main de la macro otramacro.d2w. De nuevo, los datos entrados en el formulario se pasan a esta macro en la variable *input_data*.

Cuando se invoca Net.Data, se pasa la variable como parte del URL. Por ejemplo:

```
<a href="/cgi-bin/db2www/otramacro.d2w/main?input_data=value">Next macro</a>
```

Puede acceder o manipular datos de formulario de la macro haciendo referencia al nombre de variable especificado en el formulario.

El bloque HTML siguiente del ejemplo es el bloque HTML (OUTPUT). Contiene los códigos HTML y las sentencias de macro de Net.Data que definen la salida procesada de la petición HTML (INPUT).

```
%{ ***** HTML Block: Output *****}%
%HTML (OUTPUT) {
<html>
<head>
<title>$(page_title)</title> <--- Más sustitución.

</head><body>
<h1>Output Page</h1>
<p>@rex1(input_data) <--- Esta línea contiene una llamada a la función rex1
                             pasando el argumento "input_data".

<p>
<hr>
<p>
[
<a href="/">Home page</a> |
<a href="input">Previous page</a>]
%}
```

Como el bloque HTML (INPUT), este bloque es HTML estándar con sentencias de macro de Net.Data para sustituir variables y una llamada de función. De nuevo, la variable *page_title* se sustituye en la sentencia de título. Y, como antes, este bloque contiene una llamada de función. En este caso, llama a la función *rex1* y le pasa el contenido de la variable *input_data*, que ha

recibido del formulario definido en el bloque de entrada (Input). Puede pasar cualquier número de variables a y desde una función. La definición de función especifica el número y el uso de las variables que se pasan.

Bloques XML

Si desea entregar XML a otra aplicación de proceso o a un navegador cliente, por ejemplo Microsoft Internet Explorer Versión 5, puede utilizar la estructura de bloque XML para entregar el contenido XML.

El bloque XML funciona del mismo modo que el bloque HTML; es un punto de entrada o de salida para la macro. Dentro del bloque puede entrar códigos XML directamente, hacer referencia a variables y realizar llamadas de función. Cuando se realiza una llamada de función en un bloque XML, Net.Data sabe que debe presentar los resultados utilizando códigos XML en lugar de HTML.

Para que pueda personalizar según sus necesidades el documento XML generado, el bloque XML no genera los códigos de prólogo. Entre la información de prólogo particular de la empresa e incluya una hoja de estilo de su elección. Con Net.Data se incluyen tres hojas de estilo XSL que puede utilizar. Estas hojas de estilo contienen transformaciones para todos los elementos XML y HTML generados por Net.Data (los elementos HTML generados cumplen con el estándar XHTML recién salido). No obstante, las hojas de estilo son ejemplos y se le aconseja que las amplíe o que cree unas propias.

```
%DEFINE DATABASE = "SAMPLE"

%FUNCTION(DTW_SQL) NewManager(){
select * from staff where job = 'Mgr' and years <= 5
%}

%XML(report) {
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" href="ndTable.xsl" ?>

<XMLBlock>
  <h1>List of New Managers</h1>
  @NewManager()
</XMLBlock>
%}
```

Figura 20. Macro que contiene un bloque de informe XML

Net.Data genera la información del bloque XML utilizando un pequeño conjunto de elementos, como se muestra en la descripción de documento siguiente:

```
<!------->
<!-- The root element used in all XML output generated -->
<!-- by Net.Data is the XMLBlock element. -->
```

```

<!------->
<ELEMENT XMLBlock (RowSet|ShowSQL|Message)*>
<!ATTLIST XMLBlock name CDATA #IMPLIED>

<!------->
<!-- The default presentation format for tables uses -->
<!-- the RowSet, Row, and Column elements. -->
<!------->
<ELEMENT RowSet (Row)*>
<!ATTLIST RowSet name CDATA #IMPLIED>
<ELEMENT Row (Column)*>
<!ATTLIST Row name CDATA #IMPLIED
number CDATA #IMPLIED>
<ELEMENT Column (#PCDATA)>

<!------->
<!-- SQL statements resulting from setting the SHOWSQL -->
<!-- variable are presented with the ShowSQL element. -->
<!------->
<ELEMENT ShowSQL (#PCDATA)>

<!------->
<!-- SQL statements resulting from setting the SHOWSQL -->
<!-- variable are presented with the ShowSQL element. -->
<!------->
<ELEMENT Message (#PCDATA)>

```

Los elementos se definen del modo siguiente:

XMLBlock

Contiene la parte de visualización de datos del bloque XML. Este código debe entrarse manualmente.

RowSet

Generado por Net.Data. Contiene las filas de un conjunto de resultados. El atributo de nombre de RowSet se determina del modo siguiente:

- Para una referencia de variable de tabla, se utiliza el nombre de la variable de tabla.
- Para un conjunto de resultados obtenido de una llamada a una función que ejecuta una consulta de SQL, se utiliza el nombre de la función.
- Para un conjunto de resultados obtenido de una llamada a una función que ejecuta un procedimiento almacenado, se utiliza el nombre del parámetro de conjunto de resultados. Si sólo se devuelve un conjunto de resultados y no existe ningún parámetro de conjunto de resultados, se utiliza el nombre de función.

Ejemplo:

Si se llamara a la función siguiente en un bloque XML, el RowSets asociado se visualizaría del modo mostrado.

```
%FUNCTION(DTW_SQL) call_sp() {
    CALL stored_proc
%}

%XML(report){
...
<RowSet name="call_sp" number="1">...</RowSet>
<RowSet name="call_sp" number="2">...</RowSet>
...
%}
```

Row Generado por Net.Data. Contiene las columnas de una fila y está numerado para su identificación.

Column

Generado por Net.Data. Contiene el valor de datos para la fila en particular y la columna de la que recibe el nombre.

Utilizando los elementos anteriores, Net.Data generará la salida siguiente de la macro listada en la Figura 20 en la página 114.

Content-type: text/xml

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" href="ndTable.xsl" ?>
<XMLBlock>
  <h1>List of New Managers</h1>
  <RowSet name="NewManager">
    <Row number="1">
      <Column name="ID">30</Column>
      <Column name="NAME">Marenghi</Column>
      <Column name="DEPT">38</Column>
      <Column name="JOB">Mgr</Column>
      <Column name="YEARS">5</Column>
      <Column name="SALARY">17506.75</Column>
      <Column name="COMM"></Column>
    </Row>
    <Row number="2">
      <Column name="ID">240</Column>
      <Column name="NAME">Daniels</Column>
      <Column name="DEPT">10</Column>
      <Column name="JOB">Mgr</Column>
      <Column name="YEARS">5</Column>
      <Column name="SALARY">19260.25</Column>
      <Column name="COMM"></Column>
    </Row>
  </RowSet>
</XMLBlock>
```


La Figura 21 y la Figura 22 en la página 118 muestran cómo aparecerían los datos anteriores en un navegador utilizando dos de las hojas de estilo proporcionadas con Net.Data: ndTable.xsl y ndRecord.xsl.

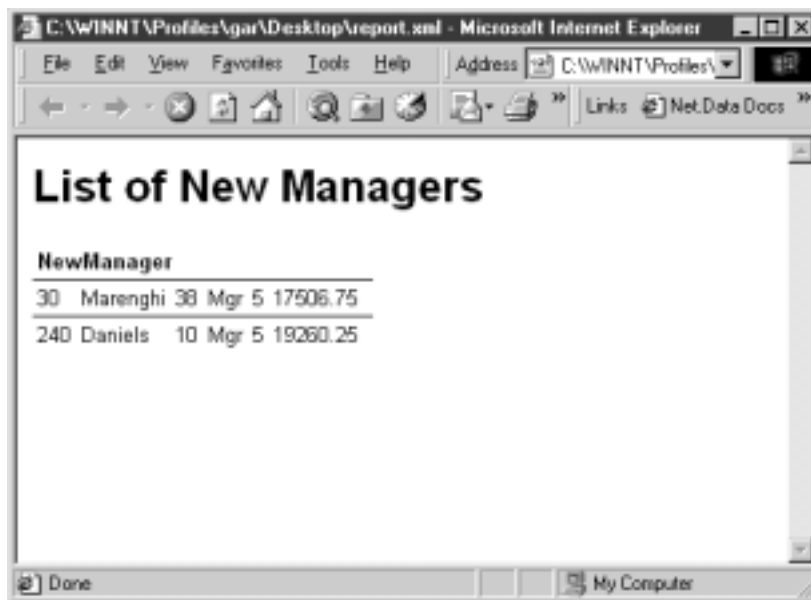


Figura 21. XML visualizado utilizando la hoja de estilo ndTable.xsl

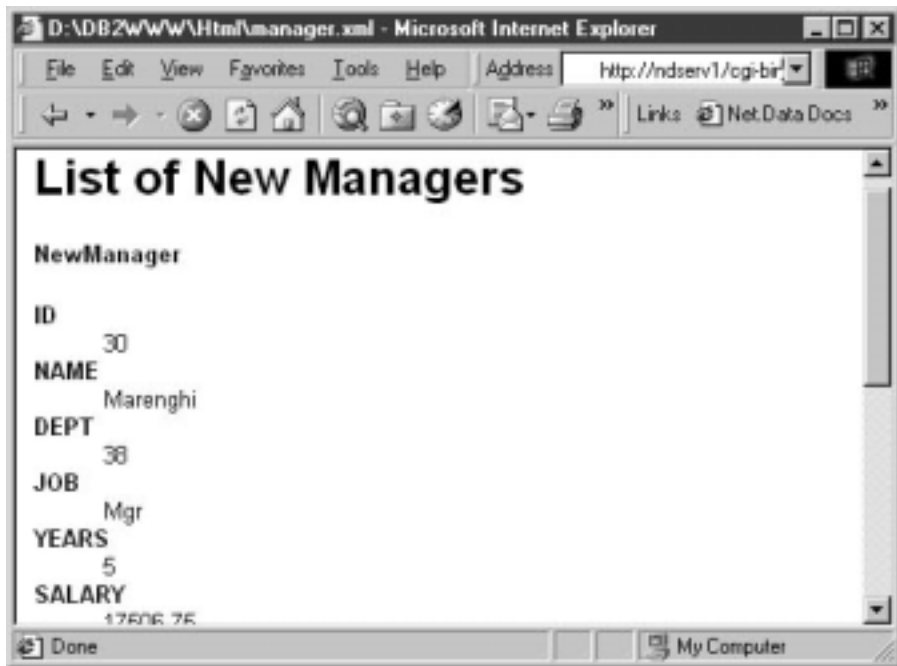


Figura 22. XML visualizado utilizando la hoja de estilo ndRecord.xsl

Variables de macro de Net.Data

Net.Data le permite definir y hacer referencia a variables de una macro de Net.Data. Además, puede pasar estas variables de la macro a los entornos de lenguaje y de éstos de nuevo a la macro. Los nombres de variable, los valores y las series literales que se pasan se denominan símbolos. Net.Data no pone ningún límite en el tamaño de los símbolos y pasará cualquier símbolo que pueda manejar la memoria del sistema. Sin embargo, puede que algunos entornos de lenguaje individuales proporcionen restricciones en el tamaño de símbolo.

Se pueden definir variables de Net.Data en función del tipo de variable y de si ésta tiene un valor predefinido. Estas variables se pueden clasificar por categorías en los tipos siguientes, basándose en el modo en que se definen:

- Variables definidas explícitamente que utilizan la sentencia DEFINE en el bloque DEFINE
- Variables predefinidas, que son variables que Net.Data deja disponibles y se establecen en un valor. Generalmente este valor no se puede cambiar.
- Variables definidas implícitamente, que son de cuatro tipos:

- Variables que no se definen explícitamente pero que se instancian cuando se les asigna un valor por primera vez.
- Variables de parámetro que forman parte de una definición de bloque FUNCTION y a las que sólo se puede hacer referencia dentro de un bloque FUNCTION.
- Variables que Net.Data instancia y que corresponden a datos de formulario o datos de serie de consulta.
- Variables que están asociadas con una tabla Net.Data y a las que sólo se puede hacer referencia dentro de un bloque ROW o un bloque REPORT.

Las secciones siguientes describen:

- “Ámbito del identificador”
- “Definición de variables” en la página 120
- “Cómo hacer referencia a variables” en la página 122
- “Tipos de variables” en la página 124

Ámbito del identificador

Un identificador, que es una variable o una llamada de función, se convierte en *visible*, lo que significa que se puede hacer referencia a él cuando se declara o se instancia. La región donde un identificador está visible se denomina su *ámbito*. Los cinco tipos de ámbito son:

- Global

Un identificador tiene ámbito global si se puede hacer referencia a él en cualquier lugar dentro de una macro. Los identificadores que tienen ámbito global son:

- Las funciones incorporadas de Net.Data
- Los datos de formulario
- Los datos de serie de consulta
- Las variables instanciadas desde dentro de un bloque HTML

- Macro

Un identificador tiene este ámbito si su declaración aparece fuera de cualquier bloque. Un bloque se inicia con un corchete de apertura ({) y finaliza con un signo de porcentaje y corchete (%}). (Tenga en cuenta que los bloques DEFINE se excluyen de esta definición y deberán crearse como sentencias DEFINE independientes). A diferencia de un identificador con ámbito global, a un identificador con ámbito de macro sólo pueden hacer los elementos de la macro que siguen a la declaración del identificador.

- Bloque FUNCTION o bloque MACRO_FUNCTION

Un identificador tiene ámbito de bloque de función si:

- El identificador se declara en la lista de parámetros de la definición de función.

Si ya existe un identificador con el mismo nombre fuera de la definición de función, Net.Data utiliza el identificador de la lista de parámetros de función dentro del bloque de función.

- El identificador se instancia en el bloque de función y no se declara o se instancia antes de la llamada de función.

Un identificador no tiene ámbito de bloque de función si se ha declarado o inicializado fuera de la función y no se ha declarado en la lista de parámetros de función. El valor del identificador dentro del bloque de función permanece sin modificaciones a no ser que lo actualice la función.

- **Bloque REPORT**

Un identificador tiene ámbito de bloque de informe si sólo se puede hacer referencia a él desde dentro de un bloque REPORT (por ejemplo, los nombres de columna de tabla N1, N2, ..., Nn). Sólo las variables que Net.Data define implícitamente como parte de su proceso de tabla pueden tener un ámbito de bloque de informe. Cualquier otra variable que se instancie tiene ámbito de bloque de función.

- **Bloque ROW**

Un identificador tiene ámbito de bloque de fila si sólo se puede hacer referencia a él desde dentro de un bloque ROW (por ejemplo, los nombres de valor de tabla V1, V2, ..., Vn). Sólo las variables que Net.Data define implícitamente como parte de su proceso de tabla pueden tener un ámbito de bloque de fila. Cualquier otra variable que se instancie tiene ámbito de bloque de función.

Definición de variables

Existen tres modos de definir variables en una macro de Net.Data:

- Sentencia o bloque de definición
- Códigos de formulario HTML
- Datos de serie de consulta

Un valor de variable recibido de los datos de serie de formulario o consulta altera temporalmente un valor de variable establecido por una sentencia DEFINE en una macro de Net.Data.

- **Sentencia o bloque DEFINE**

El modo más simple de definir una variable para usarla en una macro de Net.Data consiste en utilizar la sentencia DEFINE. La sintaxis es la siguiente:

```
%DEFINE nombre_variable="variable value"
```

```
%DEFINE nombre_variable={ valor de variable en múltiples  
                             líneas de texto  %}
```

```
%DEFINE {
    nombre_variable1="valor variable 1"
    nombre_variable2="valor variable 2"
%}
```

El *nombre_variable* es el nombre que se le da a la variable. Los nombres de variables deben empezar con una letra o un subrayado y pueden contener cualquier carácter alfanumérico, un subrayado, un punto, o un símbolo de almohadilla (#). Todos los nombres de variables son sensibles a las mayúsculas y minúsculas, excepto *N_nombreColumna* y *V_nombreColumna*, que son variables de tabla.

Por ejemplo:

```
%DEFINE reply="hello"
```

La variable *reply* tiene el valor *hello*.

Dos comillas consecutivas solas equivalen a una serie vacía. Por ejemplo:

```
%DEFINE empty=""
```

La variable *empty* tiene una serie vacía.

Si la variable contiene caracteres especiales, por ejemplo un final de línea, escriba el valor entre llaves de bloque:

```
%DEFINE introduction={
Hello,
My name is John.
%}
```

Para incluir comillas en una serie, puede utilizar dos comillas consecutivas.

```
%DEFINE HI="say ""hello"""
```

También puede utilizar llaves de bloque para evitar las comillas:

```
%DEFINE HI={ say "hello" %}
```

Para definir varias variables con una sentencia *DEFINE*, utilice un bloque *DEFINE*:

```
%DEFINE {
    variable1="valor1"
    variable2="valor2"
    variable3="valor3"
    variable4="valor4"
%}
```

- **Códigos de formulario HTML: SELECT, INPUT y TEXTAREA**

Puede utilizar códigos HTML FORM para asignar valores a variables, es decir los códigos SELECT, INPUT y TEXTAREA. El ejemplo siguiente utiliza códigos de formulario HTML estándares para definir variables de Net.Data:

```
<input name="nombre_variable" TYPE=... />
```

o

```
<select name="nombre_variable">
  <option>value one
  <option>value two
</select>
```

Para asignar una variable que se fragmenta en varias líneas o contiene caracteres especiales, por ejemplo comillas, se puede utilizar el código TEXTAREA:

```
<textarea name="nombre_variable" ROWS="4">
Please type the multi-line value
of your variable here.
</textarea>
```

El *nombre_variable* es el nombre que se le da a la variable y el valor de la variable se determina a partir de la entrada recibida en el formulario. Consulte el apartado “Formularios HTML” en la página 83 para ver un ejemplo de cómo se utiliza este tipo de definición de variable en una macro de Net.Data.

- **Datos de serie de consulta**

Puede pasar variables a Net.Data mediante la serie de consulta. Por ejemplo:

```
http://www.ibm.com/cgi-bin/db2www/stdqry1.d2w/input?field=custno
```

En el ejemplo anterior, el nombre de la variable, *field* y el valor de la variable, *custno*, especifican datos adicionales que Net.Data recibe de la serie de consulta. Net.Data recibe y procesa los datos como lo haría con datos de un formulario.

Cómo hacer referencia a variables

Puede hacer referencia a una variable definida anteriormente para devolver su valor. Para hacer referencia a una variable en las macros de Net.Data, especifique el nombre de variable entre \$(y). Por ejemplo:

```
$(nombreVariable)
$(homeURL)
```

Cuando Net.Data encuentra una referencia de variable, sustituye dicha referencia de variable por el valor de la variable. Las referencias de variables pueden contener series, referencias de variables y llamadas de función.

Puede generar nombres de variable de forma dinámica. Con esta técnica, puede utilizar bucles para procesar tablas de tamaños variables o datos de entrada para listas que se crean durante la ejecución, cuando el número de la lista no se puede determinar por adelantado. Por ejemplo, puede generar listas de elementos de formulario HTML que se generan basándose en registros devueltos de una consulta de SQL.

Para utilizar variables como parte de las sentencias de presentación de texto, haga referencia a ellas en los bloques HTML de la macro.

Referencia de variables no válidas: Las referencias de variables no válidas se resuelven en la serie vacía. Por ejemplo, si una referencia de variable contiene caracteres no válidos como, por ejemplo, un punto de exclamación (!), la referencia se resuelve en la serie vacía.

Los nombres de variables válidos deben empezar por un carácter alfanumérico o un subrayado y pueden constar de caracteres alfanuméricos, incluyendo un punto, un subrayado y un signo de almohadilla.

Ejemplo 1: Referencia de variable en un enlace

Si ha definido la variable *homeURL*:

```
%DEFINE homeURL="http://www.ibm.com/"
```

Puede hacer referencia a la página de presentación como $\$(homeURL)$ y crear un enlace:

```
<a href="\$(homeURL)">Home page</a>
```

Puede hacer referencia a variables en muchas partes de la macro de Net.Data; consulte las construcciones de lenguaje de este capítulo para determinar en qué partes de la macro están permitidas las referencias de variables. Si, cuando se hace referencia a la variable, ésta aún no se ha definido, Net.Data devuelve una serie vacía. Una referencia de variable sola no define la variable.

Ejemplo 2: Referencias de variable generadas dinámicamente

Suponga que ejecuta una sentencia SELECT de SQL con cualquier número de elementos. Puede crear un formulario HTML con campos de entrada utilizando los bloques ROW siguientes:

```
...  
%ROW {  
<input type=text name=@dtw_rconcat("I", ROW_NUM) size=10 maxlength=10 />  
%}  
...
```

Dado que ha creado campos INPUT, probablemente deseará acceder a los valores que ha entrado el usuario cuando se ha sometido el formulario en la macro para su proceso. Puede codificar un bucle para recuperar los valores en una lista de longitud variable:

```
<pre>
...
@dtw_assign(rowIndex, "1")
%while (rowIndex <= rowCount) {
The value entered for row $(rowIndex) is: $(I$(rowIndex))
@dtw_add(rowIndex, "1", rowIndex) %}
...
</pre>
```

Primero Net.Data genera el nombre de variable utilizando la referencia I\$(rowIndex). Por ejemplo, el primer nombre de variable sería I1. Entonces Net.Data utiliza dicho valor y lo resuelve en el valor de la variable.

Ejemplo 3: Referencia de variable con referencias de variable anidadas y una llamada de función

```
%define my = "my"
%define u = "lower"
%define myLOWERvar = "hey"

$( $(my)@dtw_ruppercase(u)var)
```

La referencia de variable devuelve el valor de hey.

Tipos de variables

Puede utilizar los siguientes tipos de variables en las macros.

- “Variables condicionales” en la página 125
- “Variables de entorno” en la página 125
- “Variables ejecutables” en la página 126
- “Variables ocultas” en la página 127
- “Variables de lista” en la página 128
- “Variables de tabla” en la página 129
- “Variables diversas” en la página 130
- “Variables de proceso de tabla” en la página 131
- “Variables de informe” en la página 132
- “Variables de entorno de lenguaje” en la página 133

Si asigna series a variables que Net.Data ha definido de un modo determinado, por ejemplo ENVVAR, LIST, variables de lista de condiciones, la variable ya no se comporta del modo definido. En otras palabras, la variable se convierte en una variable simple, que contiene una serie.

Consulte el manual *Net.Data Guía de consulta* para ver la sintaxis y ejemplos de cada variable.

Variables condicionales

Las variables condicionales le permiten definir un valor condicional para una variable utilizando un método similar a una construcción IF, THEN. Al definir la variable condicional, puede especificar dos valores de variable posibles. Si existe la primera variable a la que hace referencia, la variable condicional obtiene el primer valor; de lo contrario la variable condicional obtiene el segundo valor. La sintaxis para una variable condicional es:

```
varA = varB ? "valor_1" : "valor_2"
```

Si se ha definido varB, varA="valor_1", de lo contrario varA="valor_2". Esto es equivalente a utilizar un bloque IF, como en el ejemplo siguiente:

```
%IF ($(varB))  
    varA = "valor_1"  
%ELSE  
    varA = "valor_2"  
%ENDIF
```

Consulte el apartado “Variables de lista” en la página 128 para ver un ejemplo de uso de variables condicionales con variables de lista.

Variables de entorno

Puede hacer referencia a variables de entorno que el servidor Web deja disponibles para el proceso o la hebra que está procesando la petición de Net.Data. Cuando se hace referencia a la variable ENVVAR, Net.Data devuelve el valor actual de la variable de entorno con el mismo nombre.

La sintaxis para definir variables de entorno es:

```
%DEFINE var=%ENVVAR
```

Donde *var* es el nombre de la variable de entorno que se está definiendo.

Por ejemplo, la variable SERVER_NAME puede definirse como variable de entorno:

```
%DEFINE SERVER_NAME=%ENVVAR
```

Y luego se puede hacer referencia a ella:

```
The server is $(SERVER_NAME)
```

La salida tiene este aspecto:

```
The server is www.ibm.com
```

Consulte el manual *Net.Data Guía de consulta* para obtener más información sobre la sentencia ENVVAR.

Variables ejecutables

Puede invocar otros programas desde una referencia de variable utilizando variables ejecutables.

Defina variables ejecutables en una macro de Net.Data utilizando la construcción de lenguaje EXEC en el bloque DEFINE. Para obtener más información acerca del elemento de lenguaje EXEC, consulte el capítulo de construcciones de lenguaje del manual *Net.Data Guía de consulta*. En el ejemplo siguiente, se define la variable runit para ejecutar el programa ejecutable testProg:

```
%DEFINE runit=%EXEC "testProg"
```

runit se convierte en una variable ejecutable.

Net.Data ejecuta el programa ejecutable cuando encuentra una referencia de variable válida en una macro de Net.Data. Por ejemplo, se ejecuta el programa testProg cuando se hace una referencia válida a la variable runit en una macro de Net.Data.

Un método simple consiste en hacer referencia a una variable ejecutable desde otra definición de variable. El ejemplo siguiente muestra este método. La variable date se define como una variable ejecutable y dateRpt contiene una referencia a la variable ejecutable.

```
%DEFINE date=%EXEC "date"  
%DEFINE dateRpt="Today is ${date}"
```

En cualquier lugar en el que aparezca \${dateRpt} en la macro de Net.Data, Net.Data busca el programa ejecutable date y, cuando lo localiza, devuelve:

```
Today is Tue 11-07-1999
```

Cuando Net.Data encuentra una variable ejecutable en una macro, busca el programa ejecutable al que se hace referencia utilizando el método siguiente:

1. Busca en los directorios especificados por EXEC_PATH en el archivo de inicialización de Net.Data. Consulte el apartado "EXEC_PATH" en la página 23 para obtener detalles.
2. Si Net.Data no localiza el programa, el sistema busca los directorios definidos por la variable de entorno de sistema PATH o la lista de bibliotecas. Si localiza el programa ejecutable, Net.Data ejecuta el programa.

Restricción: No establezca una variable ejecutable en el valor de la salida del programa ejecutable al que llama. En el ejemplo anterior, el valor de la

variable `date` es `NULL`. Si utiliza esta variable en una llamada de función `DTW_ASSIGN` para asignar su valor a otra variable, el valor de la nueva variable después de la asignación es también `NULL`. La única finalidad de una variable ejecutable es invocar el programa que define.

También puede pasar parámetros al programa que se debe ejecutar especificándolos con el nombre de programa en la definición de variable. En este ejemplo, los valores de distancia (`distance`) y tiempo (`time`) se pasan al programa *calcMPH*.

```
%DEFINE mph=%EXEC "calcMPH $(distance) $(time)"
```

El ejemplo siguiente devuelve la fecha del sistema como parte del informe:

```
%DEFINE database="celdial"
%DEFINE tstamp=%EXEC "date"

%FUNCTION(DTW_SQL) myQuery() {
SELECT CUSTNO, CUSTNAME from dist1.customer
%REPORT{
%ROW{
<a href="/cgi-bin/db2www/exmp.d2w/report?value1=$(V1)&value2=$(V2)">
$(V1) $(V2) </a> <br />
%}
%}
%}

%HTML(report){
<h1>Report made: $(tstamp) </h1>
@myQuery()
%}
```

Cada informe visualiza la fecha para facilitar el seguimiento. Este ejemplo también pone el número de cliente y el nombre en un enlace para otra macro de `Net.Data`. Si se pulsa en cualquier cliente del informe, se llama a la macro `exmp.d2w` de `Net.Data`, pasando el número y el nombre de cliente a la macro de `Net.Data`.

Variables ocultas

Puede utilizar variables ocultas para esconder el nombre real de una variable a los usuarios de la aplicación que ven la fuente de la página Web con su navegador Web. Para definir una variable oculta:

1. Defina una variable para cada serie que desee ocultar, después de la última referencia de la variable en el bloque HTML. Las variables se definen siempre con la construcción de lenguaje `DEFINE` después de utilizarlas en el bloque HTML, como en el ejemplo siguiente. Se hace referencia a las variables `$(variable)` y, a continuación, éstas se definen.
2. En el bloque HTML donde se hace referencia a las variables, utilice signos de doble dólar en lugar de un signo de un solo dólar para hacer referencia a las variables. Por ejemplo, `$(X)` en lugar de `$(X)`.

```

%HTML(INPUT) {
<form ...>
<p>Select fields to view:
shanghai<select name="field">
<option value="$(name)"> Name
<option value="$(addr)"> Address
...
</form>
%}

%DEFINE {
name="customer.name"
addr="customer.address"
%}

%FUNCTION(DTW_SQL) mySelect() {
    SELECT $(Field) FROM customer
%}

...

```

Cuando un navegador Web visualiza el formulario HTML, `$(name)` y `$(addr)` se sustituyen por `$(name)` y `$(addr)` respectivamente, de modo que los nombres de tabla y columna reales no aparecen nunca en el formulario HTML. Los usuarios de la aplicación no pueden darse cuenta de que los verdaderos nombres de variable están ocultos. Cuando el usuario somete el formulario, se llama al bloque HTML(REPORT). Cuando `@mySelect()` llama al bloque FUNCTION, `$(Field)` se sustituye en la sentencia de SQL por `customer.name` o `customer.addr` en la consulta de SQL.

Variables de lista

Utilice variables de lista para crear una serie delimitada de valores. Son especialmente útiles para ayudarle a crear una consulta de SQL con múltiples elementos como los que se encuentran en algunas cláusulas WHERE o HAVING. La sintaxis para una variable de lista es:

```
%LIST " separador_valor " nombre_variable
```

Recomendación: Los espacios en blanco son significativos. Inserte un espacio antes y después del separador de valor en la mayoría de los casos. La mayoría de las consultas utilizan operadores booleanos o matemáticos (por ejemplo AND, OR o >) para el separador de valor. El ejemplo siguiente ilustra el uso de variables de lista, ocultas y condicionales:

```

%HTML(INPUT) {
<form method="post" action="/cgi-bin/db2www/example2.d2w/report">
<h2>Select one or more cities:</h2>
<input type="checkbox" name="conditions" value="$(cond1)" />Sao Paolo<br />
<input type="checkbox" name="conditions" value="$(cond2)" />Seattle<br />
<input type="checkbox" name="conditions" value="$(cond3)" />Shanghai<br />
<input type="submit" value="submit query" />

```

```

</form>
%}

%DEFINE{
DATABASE="custcity"
%LIST " OR " conditions
cond1="cond1='Sao Paolo'"
cond2="cond2='Seattle'"
cond3="cond3='Shanghai'"
whereClause= ? "WHERE $(conditions)"
%}

%FUNCTION(DTW_SQL) mySelect(){
SELECT name, city FROM citylist
$(whereClause)
%}

%HTML(REPORT){
@mySelect()
%}

```

En el formulario HTML, si no se selecciona ningún recuadro, conditions es NULL, de modo que whereClause es también NULL en la consulta. De lo contrario, whereClause tiene los valores seleccionados separados por OR. Por ejemplo, si se seleccionan las tres ciudades, la consulta de SQL es:

```

SELECT name, city FROM citylist
WHERE cond1='Sao Paolo' OR cond2='Seattle' OR cond3='Shanghai'

```

Este ejemplo muestra que se selecciona Seattle, lo que produce esta consulta de SQL:

```

SELECT name, city FROM citylist
WHERE cond1='Seattle'

```

Variables de tabla

La variable de tabla define un conjunto de datos relacionados. Contiene un conjunto de filas y columnas que incluyen una fila de cabeceras de columna. Una tabla se define en la macro de Net.Data como en la sentencia siguiente:

```
%DEFINE myTable=%TABLE(30)
```

El número a continuación de %TABLE es el límite del número de filas que puede contener esta variable de tabla. Para especificar una tabla sin límite en el número de filas, utilice el valor por omisión o especifique ALL, como se muestra en estos ejemplos:

```

%DEFINE myTable2=%TABLE
%DEFINE myTable3=%TABLE(ALL)

```

Cuando se define una tabla, ésta tiene cero filas y cero columnas. El único modo de poder llenar una tabla con valores es pasándola como un parámetro OUT o INOUT a una función o utilizando las funciones de tabla incorporadas

proporcionadas por Net.Data. El entorno de lenguaje DTW_SQL pone automáticamente los resultados de una sentencia SELECT en una tabla.

Para entornos de lenguaje que no son de base de datos, por ejemplo DTW_REXX o DTW_PERL, el entorno de lenguaje también es responsable de establecer los valores de tabla. Sin embargo, el programa o script de entorno de lenguaje define los valores de tabla de casilla en casilla. Consulte el “Capítulo 6. Utilización de entornos de lenguaje” en la página 159 para obtener más información sobre cómo utilizan los entornos de lenguaje las variables de tabla.

Puede pasar una tabla entre funciones haciendo referencia al nombre de variable de tabla. Se puede hacer referencia a los elementos individuales de la tabla en un bloque REPORT de una función o utilizando las funciones de tabla de Net.Data. Consulte el apartado “Variables de proceso de tabla” en la página 131 para acceder a elementos individuales de una tabla dentro de un bloque REPORT y consulte el apartado “Funciones de tabla” en la página 144 para acceder a elementos individuales de una tabla utilizando una función de tabla. Normalmente las variables de tabla se llenan con valores en una función de SQL y entonces se utilizan como entrada a un informe, en la función de SQL o en otra función después de haberse pasado a dicha función como parámetro. Puede pasar variables de tabla como parámetros IN, OUT o INOUT a cualquier función no SQL. Las tablas pueden pasarse a las funciones de SQL sólo como parámetros OUT.

Si hace referencia a una variable de tabla, el contenido de la tabla se visualiza y se formatea basándose en el valor de la variable DTW_HTML_TABLE. En el ejemplo siguiente, se visualizará el contenido de myTable:

```
%HTML (output) {  
    $(myTable)  
}
```

Los nombres de columna y los valores de campo de una tabla se tratan como elementos de matriz con un origen de 1.

Variables diversas

Estas variables son variables definidas por Net.Data que se pueden utilizar para:

- Influir en el proceso de Net.Data
- Averiguar el estado de una llamada de función
- Obtener información acerca del conjunto de resultados de una consulta de base de datos
- Determinar información acerca de las ubicaciones y las fechas de los archivos

Las variables diversas pueden tener un valor predefinido que Net.Data determina o pueden tener valores establecidos por el usuario. Por ejemplo, Net.Data determina el valor de la variable DTW_CURRENT_FILENAME basándose en el archivo actual que está procesando, mientras que el usuario puede especificar si Net.Data elimina el espacio en blanco adicional producido por los tabuladores y los caracteres de línea nueva.

Las variables predefinidas se utilizan como referencias de variable dentro de la macro y proporcionan información acerca del estado actual de los archivos, de las fechas o del estado de una llamada de función. Por ejemplo, para recuperar el nombre del archivo actual, puede utilizar:

```
%REPORT {  
  <p>This file is <i>$(DTW_CURRENT_FILENAME)</i>.</p>  
}
```

Los valores de variable modificables se establecen generalmente utilizando una sentencia DEFINE o con la función @DTW_ASSIGN() y permiten que afecte al modo en que Net.Data procesa la macro. Por ejemplo, para especificar si se elimina espacio en blanco, puede utilizar la sentencia DEFINE siguiente:

```
%DEFINE DTW_REMOVE_WS="YES"
```

Variables de proceso de tabla

Net.Data define variables de proceso de tabla para utilizarlas en los bloques REPORT y ROW. Utilice estas variables para hacer referencia a valores de consultas de SQL y de llamadas de función.

Las variables de proceso de tabla tienen un valor predefinido que Net.Data determina. Estas variables le permiten hacer referencia a los valores de los conjuntos de resultados de las llamadas de función o consultas de SQL por la columna, la fila o el campo que se está procesando. También puede acceder a la información acerca del número de filas que se están procesando o acerca de una lista de todos los nombres de columna.

Por ejemplo, mientras Net.Data procesa un conjunto de resultados de una consulta de SQL, asigna el valor de la variable Nn para cada nombre de columna actual, de modo que N1 se asigna a la primera columna, N2 se asigna a la segunda columna y así sucesivamente. Se puede hacer referencia al nombre de columna actual para la salida de página Web.

Utilice variables de proceso de tabla como referencias de variable dentro de la macro. Por ejemplo, para recuperar el nombre de la columna actual que se está procesando, puede utilizar:

```
%REPORT {  
  <p>Column 1 is <i>$(N1)</i>.</p>  
}
```

Las variables de proceso de tabla también proporcionan información acerca de los resultados de una consulta. Puede hacer referencia a la variable `TOTAL_ROWS` de la macro para mostrar cómo se devuelven muchas filas de una consulta de SQL, como en el ejemplo siguiente:

```
Names found: $(TOTAL_ROWS)
```

algunas de las variables de proceso de tabla se ven afectadas por otras variables o funciones incorporadas. Por ejemplo, `TOTAL_ROWS` necesita que la variable de entorno de lenguaje de SQL `DTW_SET_TOTAL_ROWS` esté activada para que Net.Data asigne el valor de `TOTAL_ROWS` al procesar los resultados de una llamada de función o una consulta de SQL como en el ejemplo siguiente:

```
%DEFINE DTW_SET_TOTAL_ROWS="YES"
...
```

```
Names found: $(TOTAL_ROWS)
```

Variables de informe

Net.Data visualiza la salida de página Web generada desde la macro en un formato de informe por omisión. En un bloque HTML, el formato de informe por omisión visualiza una tabla utilizando los códigos `<pre>` `</pre>`, en un bloque XML, se utilizan los códigos `<RowSet>`, `<Row>` y `<Column>`. Puede alterar temporalmente el informe por omisión definiendo un bloque `REPORT` con instrucciones para visualizar la salida o utilizando una de las variables de informe para impedir que se genere el informe por omisión.

Las variables de informe le ayudan a personalizar cómo se visualiza la salida de página Web y cómo se utiliza dicha salida con informes por omisión y tablas Net.Data. Deberá definir estas variables antes de utilizarlas con una sentencia `DEFINE` o con la función `@DTW_ASSIGN()`.

Las variables de informe especifican el espaciado, alteran temporalmente los formatos de informe por omisión, especifican si la salida de tabla debe visualizarse en HTML o en caracteres de longitud fija y especifican otras características de visualización. Por ejemplo, puede utilizar la variable `ALIGN` para controlar los espacios iniciales y de cola para las variables de proceso de tabla. El ejemplo siguiente utiliza la variable `ALIGN` para separar mediante un espacio cada nombre de columna de una lista devuelta por una consulta.

```
%DEFINE ALIGN="YES"
...
<p>Your query was on these columns: $(NLIST)
```

La variable de informe `START_ROW_NUM` le permite determinar en qué fila se deben empezar a visualizar los resultados de una consulta. Por ejemplo, el valor de variable siguiente especifica que Net.Data empezará a visualizar los resultados de una consulta en la tercera fila.


```
%DEFINE START_ROW_NUM = "3"
```

También puede determinar si Net.Data utiliza códigos HTML para el formato por omisión. Con DTW_HTML_TABLE establecido en YES, se genera una tabla HTML en lugar de una tabla formateada con texto.

```
%DEFINE DTW_HTML_TABLE="YES"
```

```
%FUNCTION(DTW_SQL){  
  SELECT NAME, ADDRESS FROM $(qTable)  
%}
```

Variables de entorno de lenguaje

Estas variables se utilizan con entornos de lenguaje y afectan al modo en que el entorno de lenguaje procesa una petición.

Con estas variables, puede efectuar tareas tales como establecer conexiones a bases de datos, proporcionar texto alternativo para applets Java, habilitar soporte NLS y determinar si la ejecución de una sentencia de SQL es satisfactoria.

Por ejemplo, puede utilizar la variable SQL_STATE para acceder al valor de estado de SQL devuelto de la base de datos o visualizar dicho valor.

```
%FUNCTION (DTW_SQL) val1() {  
  select * from customer  
%REPORT {  
  ...  
%ROW {  
  ...  
%}  
  SQLSTATE=$(SQL_STATE)  
%}
```

El ejemplo siguiente muestra cómo definir a qué base de datos se debe acceder.

```
%DEFINE DATABASE="CELDIAL"
```

Funciones de Net.Data

Net.Data proporciona funciones incorporadas para utilizarlas en las aplicaciones, por ejemplo funciones de manipulación de palabras o series o funciones que recuperan y establecen funciones de variables de tabla. También puede definir funciones para utilizarlas con la aplicación, por ejemplo para llamar a un programa externo o a un procedimiento almacenado.

Funciones definidas por el usuario

Funciones que define para utilizarlas con la aplicación, por ejemplo llamar a un programa externo o a un procedimiento almacenado.

Funciones incorporadas de Net.Data

Funciones que Net.Data proporciona para utilizarlas en las aplicaciones, por ejemplo funciones para manipular palabras y series y funciones que obtienen y establecen variables de tabla.

Estas secciones describen los temas siguientes:

- “Definición de funciones”
- “Llamada de funciones” en la página 140
- “Llamada de funciones incorporadas de Net.Data” en la página 140

Definición de funciones

Para definir sus propias funciones en la macro, utilice un bloque FUNCTION o un bloque MACRO_FUNCTION:

Bloque FUNCTION

Define una subrutina que se invoca desde una macro de Net.Data y que procesa un entorno de lenguaje. Los bloques FUNCTION deben contener sentencias de lenguaje o llamadas a un programa externo.

Bloque MACRO_FUNCTION

Define una subrutina que se invoca desde una macro de Net.Data y que procesa Net.Data en lugar de un entorno de lenguaje. Los bloques MACRO_FUNCTION pueden contener cualquier sentencia que esté permitida en un bloque HTML.

Sintaxis: Utilice la sintaxis siguiente para definir funciones:

Bloque FUNCTION:

```
%FUNCTION(tipo) nombre-función([uso] [tipodatos] parámetro, ...)
    [RETURNS(var-retorno)] {
    sentencias-ejecutables
    [bloque-informe]
    ...
    [bloque-informe]
    [bloque-mensaje]
%}
```

Bloque MACRO_FUNCTION:

```
%MACRO_FUNCTION nombre-función([uso] parámetro, ...) {
    sentencias-ejecutables
    [bloque-informe]
    ...
    [bloque-informe]
%}
```

Donde:

tipo Identifica un entorno de lenguaje que se configura en el archivo de inicialización. El entorno de lenguaje invoca un procesador de

lenguaje específico (que procesa las sentencias ejecutables) y proporciona una interfaz estándar entre Net.Data y el procesador de lenguaje.

nombre-función

Especifica el nombre del bloque FUNCTION o MACRO_FUNCTION. Una llamada de función especifica el *nombre-función*, precedido por un signo de arroba (@). Consulte el apartado “Llamada de funciones” en la página 140 para obtener detalles.

Puede definir múltiples bloques FUNCTION o MACRO_FUNCTION con el mismo nombre para que se procesen al mismo tiempo. Todos y cada uno de los bloques deben tener listas de parámetros idénticas. Cuando Net.Data llama a una función, todos los bloques FUNCTION con el mismo nombre o los bloques MACRO_FUNCTION con el mismo nombre se ejecutan en el orden en el que se han definido en la macro de Net.Data.

uso

Especifica si un parámetro es un parámetro de entrada (IN), un parámetro de salida (OUT) o de ambos tipos (INOUT). Esta designación indica si el parámetro se pasa al bloque FUNCTION, al bloque MACRO_FUNCTION o a ambos o si se recibe de uno u otro bloque o de ambos. El tipo de uso se aplica a todos los parámetros subsiguientes de la lista de parámetros hasta que lo modifique otro tipo de uso. El tipo por omisión es IN.

tipodatos

Tipo de datos del parámetro. Algunos entornos de lenguaje esperan tipos de datos para los parámetros que se pasan. Por ejemplo, el entorno de lenguaje SQL los espera al llamar a los procedimientos almacenados. Consulte el “Capítulo 6. Utilización de entornos de lenguaje” en la página 159 para obtener más información acerca de los tipos de datos soportados para el entorno de lenguaje que esté utilizando.

parámetro

El nombre de una variable con ámbito local que se sustituye por el valor de un argumento correspondiente especificado en una llamada de función. Las referencias de parámetros, por ejemplo \$(parm1), en las sentencias ejecutables o el bloque REPORT se sustituyen por el valor real del parámetro. Además, los parámetros se pasan al entorno de lenguaje y son accesibles para las sentencias ejecutables utilizando la sintaxis natural de dicho lenguaje o como variables de entorno. Las referencias de variables de parámetro no son válidas fuera de los bloques FUNCTION o MACRO_FUNCTION.

var-retorno

Especifique este parámetro después de la palabra clave RETURNS para identificar un parámetro OUT especial. El valor de la variable de

retorno se asigna en el bloque de función y su valor se devuelve al lugar en la macro desde el que se ha llamado a la función. Por ejemplo, en la sentencia siguiente, `<p>Mi nombre es @my_name()`., se sustituye `@my_name()` por el valor de la variable de retorno. Si no especifica la cláusula RETURNS, el valor de la llamada de función es:

- NULL si el código de retorno de la llamada al entorno de lenguaje es cero
- El valor del código de retorno, cuando el código de retorno es distinto de cero.

sentencias-ejecutables

Conjunto de sentencias de lenguaje que se pasa al entorno de lenguaje especificado para procesarlo después de que se hayan sustituido las variables y se hayan procesado las funciones. *sentencias-ejecutables* puede contener referencias de variables de Net.Data y llamadas de función de Net.Data. *sentencias-ejecutables* incluye las sentencias ejecutables a las que se permite el acceso a un bloque HTML.

Para los bloques FUNCTION, Net.Data sustituye todas las referencias de variables por los valores de variable, ejecuta todas las llamadas de función y sustituye las llamadas de función por sus valores resultantes antes de que se pasen las sentencias ejecutables al entorno de lenguaje. Cada entorno de lenguaje procesa las sentencias de un modo diferente. Para obtener más información sobre cómo especificar sentencias ejecutables o llamar a programas ejecutables, consulte el apartado “Variables ejecutables” en la página 126.

Para los bloques MACRO_FUNCTION, las sentencias ejecutables son una combinación de texto y construcciones de lenguaje de macro de Net.Data. En este caso, no hay ningún entorno de lenguaje implicado porque Net.Data actúa como el procesador de lenguajes y procesa las sentencias ejecutables.

bloque-informe

Define uno o más bloques REPORT para manejar la salida del bloque FUNCTION o MACRO_FUNCTION. Consulte el apartado “Bloques de informe” en la página 147.

bloque-mensaje

Define el bloque MESSAGE, que maneja los mensajes devueltos por el bloque FUNCTION. Consulte el apartado “Bloques de mensajes” en la página 138.

Defina funciones fuera de cualquier otro bloque y antes de que se llamen en la macro de Net.Data.

Utilización de caracteres especiales en las funciones

Cuando, en la sección de sentencias de lenguaje de un bloque de función, se utilizan caracteres que coinciden con la sintaxis de las construcciones de

lenguaje de Net.Data como parte del código de programa incorporado sintácticamente válido (por ejemplo REXX o Perl), dichos caracteres pueden interpretarse incorrectamente como construcciones de lenguaje de Net.Data, produciendo errores o resultados imprevisibles en una macro.

Por ejemplo, una función Perl puede utilizar los caracteres delimitadores de bloque COMMENT, %{. Cuando se ejecuta la macro, los caracteres %{ se interpretan como el principio de un bloque COMMENT. Entonces Net.Data busca el final del bloque COMMENT, que cree encontrar cuando lee el final del bloque de función. A continuación Net.Data procede a buscar el final del bloque de función y, cuando no lo puede encontrar, emite un error.

Use uno de los métodos siguientes para utilizar caracteres delimitadores de bloque COMMENT o cualquier otro carácter especial de Net.Data como parte del código de programa incorporado, sin que Net.Data los interprete como caracteres especiales:

- Utilice la sentencia EXEC para llamar al código de programa, en lugar de poner el código incorporado.
- Utilice una referencia de variable para especificar los caracteres especiales.

Por ejemplo, la función Perl siguiente contiene caracteres que representan un delimitador de bloque COMMENT, %{, como parte de las sentencias de lenguaje Perl:

```
%FUNCTION(DTW_PERL) func() {
    ...
    for $num_words (sort bynumber keys %{ $Rtitles{$num} }) {
        &make_links($Rtitles{$num}{$num_words});
    }
    ...
%}
```

Para asegurar que Net.Data interpreta los caracteres %{ como código fuente Perl en lugar de como un delimitador de bloque COMMENT de Net.Data, vuelva a escribir la función de uno de los modos siguientes:

- Utilice la sentencia %EXEC:


```
%FUNCTION(DTW_PERL) func() {
    %EXEC{ func.pr1 %{
%}
```
- Utilice una referencia de variable para especificar los caracteres %{:


```
%define percent_openbrace = "%{"

%FUNCTION(DTW_PERL) func() {
    ...
    for $num_words (sort by number keys $(percent_openbrace) $Rtitles{$num} ) {
```

```

        &make_links($Rtitles{$num}{$num_words});
    }
    ...
%}

```

Bloques de mensajes

El bloque MESSAGE le permite determinar cómo continuar después de una llamada de función, basándose en el éxito o el fracaso de la llamada de función, y le permite visualizar información al llamante de la función. Cuando se procesa un mensaje, Net.Data establece la variable de entorno de lenguaje RETURN_CODE para cada llamada de función en un bloque FUNCTION. En una llamada de función RETURN_CODE no se establece en un bloque MACRO_FUNCTION.

Un bloque MESSAGE consta de una serie de sentencias de mensaje, cada una de las cuales especifica un valor de código de retorno, texto de mensaje y una acción a realizar. La sintaxis de un bloque MESSAGE se muestra en el capítulo de construcciones de lenguaje del manual *Net.Data Guía de consulta*.

Un bloque MESSAGE puede tener un ámbito global o local. Si se especifica en la capa de macro más externa, el bloque MESSAGE tiene ámbito global y está activo para todas las llamadas de función ejecutadas en la macro de Net.Data. Si define más de un bloque MESSAGE global, estará activo el último que haya definido. Sin embargo, si el bloque MESSAGE se define en un bloque FUNCTION, su ámbito es local a dicho bloque FUNCTION (excepto para las funciones incorporadas de Net.Data, cuyos errores los manejan los bloques de mensajes globales).

Net.Data utiliza estas normas para procesar el valor de las variables RETURN_CODE o SQL_STATE de una llamada de función:

1. Buscar en el bloque MESSAGE local una coincidencia exacta del valor de RETURN_CODE o SQL_STATE; salir o continuar tal como se especifique.
2. Si el valor no es 0, buscar +default o -default en el bloque MESSAGE local; en función del signo del valor, salir o continuar tal como se especifique.
3. Si el valor no es 0, buscar default en el bloque MESSAGE local; salir o continuar tal como se especifique.
4. Buscar en el bloque MESSAGE global una coincidencia exacta de RETURN_CODE o SQL_STATE; salir o continuar tal como se especifique.
5. Si el valor no es 0, buscar +default o -default en el bloque MESSAGE global; en función del signo del valor, salir o continuar tal como se especifique.
6. Si el valor no es 0, buscar default en el bloque MESSAGE global; salir o continuar tal como se especifique.

7. Si el valor no es 0, emitir el mensaje interno por omisión de Net.Data y salir.

El ejemplo siguiente muestra parte de una macro de Net.Data con un bloque MESSAGE global y un bloque MESSAGE para una función.

```
%{ global message block %}  
%MESSAGE {  
    -100      : "Return code -100 message"    : exit  
    100       : "Return code 100 message"     : continue  
    +default : {  
        This is a long message that spans more  
        than one line. You can use HTML tags, including  
        links and forms, in this message. %}    : continue  
    %}  
  
%{ local message block inside a FUNCTION block %}  
%FUNCTION(DTW_REXX) my_function() {  
    %EXEC { my_command.cmd %}  
    %MESSAGE {  
        -100      : "Return code -100 message"    : exit  
        100       : "Return code 100 message"     : continue  
        -default : {  
            This is a long message that spans more  
            than one line. You can use HTML tags, including  
            links and forms, in this message. %}    : exit  
        %}  
    }
```

Si se devuelve *my_function()* con un valor RETURN_CODE de 50, Net.Data procesa el error en este orden:

1. Busca una coincidencia exacta en el bloque MESSAGE local.
2. Busca +default en el bloque MESSAGE local.
3. Busca default en el bloque MESSAGE local.
4. Busca una coincidencia exacta en el bloque MESSAGE global.
5. Busca +default en el bloque MESSAGE global.

Cuando Net.Data encuentra una coincidencia, envía el texto de mensaje al navegador Web y comprueba la acción solicitada.

Cuando se especifica continue, Net.Data continúa procesando la macro de Net.Data después de imprimir el texto del mensaje. Por ejemplo, si una macro llama a *my_functions()* cinco veces y se encuentra el error 100 durante el proceso con el bloque MESSAGE del ejemplo, la salida de un programa puede tener este aspecto:

```
.  
. .  
11 May 1997          $245.45  
13 May 1997          $623.23  
19 May 1997          $ 83.02
```

return code 100 message	
22 May 1997	\$ 42.67
Total:	\$994.37

Llamada de funciones

Utilice una sentencia de llamada de función de Net.Data para llamar a las funciones definidas por el usuario y las funciones incorporadas. Utilice el carácter de arroba (@) seguido de un nombre de función o un nombre de función de macro:

@nombre_función([*argumento*,...])

nombre_función

Es el nombre de la función o de la función de macro a invocar. La función debe estar ya definida en la macro de Net.Data, a no ser que se trate de una función incorporada.

argumento

Es el nombre de una variable, una serie entrecomillada, una referencia de variable o una llamada de función. Los argumentos de una llamada de función se comparan con los parámetros de una lista de parámetros de función o de función de macro. Y se asigna a cada parámetro el valor de su argumento correspondiente mientras se está procesando la función o la función de macro. Los argumentos deben tener el mismo número y tipo que los parámetros correspondientes.

Las series entrecomilladas utilizadas como argumentos pueden contener referencias de variables y llamadas de funciones.

Ejemplo 1: Llamada de función con un argumento de texto

```
@myFunction("abc")
```

Ejemplo 2: Llamada de función con un argumento de variable y un argumento de llamada de función

```
@myFunction(myvar, @DTW_rADD("2","3"))
```

Ejemplo 3: Llamada de función con un argumento de texto que contiene una referencia de variable y una llamada de función

```
@myFunction("abc$(myvar)def@DTW_rADD("2","3")ghi")
```

Llamada de funciones incorporadas de Net.Data

Net.Data proporciona un gran conjunto de funciones incorporadas para simplificar el desarrollo de páginas Web. Estas funciones ya están definidas por Net.Data, por consiguiente no necesita definir las. Puede llamar a estas funciones igual que llamaría a otras funciones.

La Figura 23 muestra cómo interactúan las funciones incorporadas de Net.Data y la macro.

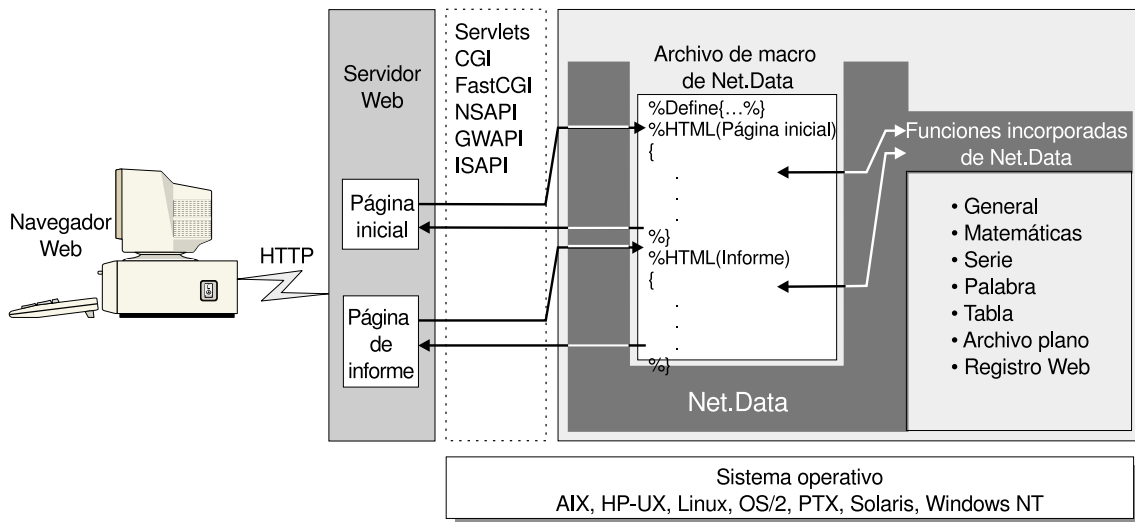


Figura 23. Funciones incorporadas de Net.Data

Las funciones incorporadas pueden devolver los resultados de tres modos, en función del prefijo:

- **DTW_, DTWF_ y DTWR_:** Los resultados de la llamada se devuelven en un parámetro de salida o no se devuelve ningún resultado. (**DTWF_** es el prefijo para las funciones de archivo plano. **DTWR_** es el prefijo para las funciones de registro Web).
- **DTW_r y DTWR_r:** Los resultados de la llamada de función sustituyen a la llamada de función en la macro, del mismo modo que el valor de la palabra clave **RETURNS** sustituye a la llamada de función para una función definida por el usuario que ha especificado una palabra clave **RETURNS**.
- **DTW_m:** Se devuelven varios resultados en cada uno de los parámetros pasados a la función.

Algunas funciones incorporadas no tienen cada uno de los tipos. Para determinar qué tipo tiene una función incorporada determinada, consulte el capítulo de funciones incorporadas de Net.Data del manual *Net.Data Guía de consulta*.

Las secciones siguientes proporcionan una visión general de alto nivel de las funciones incorporadas de Net.Data. Utilice estas funciones para realizar funciones de manipulación de tabla, de palabra, de serie, matemáticas o de uso general. Consulte en el manual *Net.Data Guía de consulta* las descripciones de cada función con sintaxis y ejemplos. Algunas de estas funciones necesitan

que se establezcan variables antes de utilizarse o deben utilizarse en un contexto específico. No todos los sistemas operativos soportan cada una de las funciones incorporadas. Consulte el manual *Net.Data Guía de consulta* para determinar qué funciones se soportan para el sistema operativo correspondiente.

- “Funciones de uso general”
- “Funciones matemáticas” en la página 143
- “Funciones de serie” en la página 143
- “Funciones de palabra” en la página 143
- “Funciones de tabla” en la página 144
- “Funciones de archivo plano” en la página 144
- “Funciones de registro Web” en la página 144

Funciones de uso general

Este conjunto de funciones le ayudan a desarrollar páginas Web modificando datos o accediendo a servicios del sistema. Puede utilizarlas para enviar correo, procesar cookies HTTP, generar códigos de escape HTML y obtener información adicional útil del sistema.

Por ejemplo, para especificar que Net.Data debe salir de una macro si se produce una condición específica, sin procesar el resto de la macro, utilice la función DTW_EXIT:

```
%HTML(cache_example) {

<html>
  <head>
    <title>This is the page title</title>
  </head>
  <body>
    <center>
      <h3>This is the Main Heading</h3>
      <!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
      <! Joe Smith sees a very short page                      !>
      <!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
      %IF (customer == "Joe Smith")
    </body>
  </html>

@DTW_EXIT()

%ENDIF

...

</body>
</html>
%}
```

Otra función útil es la función DTW_URLESCSEQ, que sustituye caracteres que no están permitidos en un URL por sus valores de escape. Por ejemplo, si la variable de entrada `string1` es igual a "Guys & Dolls", DTW_URLESCSEQ asigna la variable de salida al valor "Guys%20%26%20Dolls".

Funciones matemáticas

Estas funciones realizan operaciones matemáticas, permitiéndole calcular o modificar datos numéricos. Además de las operaciones matemáticas estándares, también puede realizar división de módulo, especificar una precisión de resultado y utilizar notación científica.

Por ejemplo, la función DTW_POWER eleva el valor de su primer parámetro a la potencia de su segundo parámetro y devuelve el resultado, tal como se muestra en el ejemplo siguiente:

```
@DTW_POWER("2", "-3", result)
```

DTW_POWER devuelve ".125" en la variable `result`

Funciones de serie

Estas funciones le permiten manipular caracteres dentro de series. Puede cambiar las mayúsculas y minúsculas de una serie, insertar o suprimir caracteres, asignar un valor de serie a otra variable, más otras funciones útiles.

Por ejemplo, puede utilizar DTW_ASSIGN para asignar el valor de una variable de entrada a una variable de salida. También puede utilizar esta función para cambiar una variable en una macro. En el ejemplo siguiente, la variable `RC` se asigna a cero.

```
@DTW_ASSIGN(RC, "0")
```

Otras funciones de serie incluyen DTW_CONCAT, que concatena series, y DTW_INSERT, que inserta series en una posición específica, así como muchas otras funciones de manipulación de series.

Funciones de palabra

Estas funciones le permiten manipular palabras de series de caracteres. La mayoría de estas funciones funcionan de forma similar a las funciones de serie, pero en palabras enteras. Por ejemplo, le permiten contar el número de palabras de una serie, eliminar palabras y buscar una palabra en una serie.

Por ejemplo, utilice DTW_DELWORD para suprimir el número especificado de palabras de una serie:

```
@DTW_DELWORD("Now is the time", "2", "2", result)
```

DTW_DELWORD devuelve la serie "Now time".

Otras funciones de palabra incluyen DTW_WORDLENGTH, que devuelve el número de caracteres de una palabra, y DTW_WORDPOS, que devuelve la posición de una palabra dentro de una serie.

Funciones de tabla

Puede utilizar estas funciones para generar informes o formularios utilizando los datos de una variable de tabla de Net.Data. También puede utilizar estas funciones para crear tablas de Net.Data y para manipular y recuperar valores de dichas tablas. Las variables de tabla contienen un conjunto de valores y sus nombres de columna asociados. Proporcionan un modo cómodo de pasar grupos de valores a una función.

Por ejemplo, DTW_TB_APPENDROW añade una fila a la tabla. En el ejemplo siguiente, Net.Data añade diez filas a la tabla, myTable:

```
@DTW_TB_APPENDROW(myTable, "10")
```

Adicionalmente, DTW_TB_DUMP, devuelve el contenido de una variable de tabla de macro, que se incluye entre los códigos `<pre></pre>`, con cada fila de la tabla visualizada en una línea diferente. Y DTW_TB_CHECKBOX devuelve uno o más códigos de entrada de recuadro de selección de HTML de una variable de tabla de macro.

Funciones de archivo plano

Utilice las funciones de interfaz de archivo plano (FFI) para abrir, leer y manipular datos de fuentes de archivo plano (archivos de texto), así como datos almacenados en archivos planos.

Por ejemplo, DTWF_APPEND, graba el contenido de una variable de tabla al final de un archivo y DTWF_DELETE suprime registros de un archivo.

Adicionalmente, las funciones FFI permiten el bloqueo de archivos con DTWF_CLOSE y DTWF_OPEN. DTWF_OPEN bloquea un archivo de modo que otra petición no pueda leer o actualizar el archivo. DTWF_CLOSE libera el archivo cuando Net.Data ha terminado con él, permitiendo que otras peticiones accedan al archivo.

Funciones de registro Web

Utilice las funciones de registro Web para mantener registros y las entradas que éstos contienen. Un registro Web es un archivo con una clave mantenida por Net.Data para permitirle añadir, recuperar y suprimir entradas de forma fácil.

Por ejemplo, DTWR_ADDENTRY añade entradas, mientras que DTWR_DELENTY las suprime. DTWR_LISTSUB devuelve información acerca de las entradas de registro en un parámetro de tabla OUT y

DTWR_UPDATEENTRY sustituye los valores existentes para una entrada de registro especificado por un valor nuevo.

Generación de marcación de documentos

Net.Data genera dinámicamente documentos HTML o XML que van a ser utilizados por una aplicación cliente, por ejemplo un navegador Web. Las secciones siguientes describen las diversas construcciones que puede utilizar para formatear documentos con las macros de Net.Data. Consulte el capítulo de construcciones de lenguaje del manual *Net.Data Guía de consulta* para obtener información de sintaxis específica para cada una.

Bloques HTML y XML

La aplicación cliente invoca Net.Data especificando el nombre de macro y el nombre de uno de los puntos de entrada de la macro. El punto de entrada a la macro puede ser un bloque HTML o XML. Estos bloques contienen la mayor parte de la marcación de documento y se encuentran en el lugar donde se efectúan las llamadas a las funciones. Constituyen el bloque de programa "principal" de la macro.

Dado que el bloque de puntos de entrada maneja la ejecución de la macro, tiene que haber al menos un bloque de puntos de entrada en una macro. Pueden existir múltiples bloques HTML o XML, pero sólo se ejecuta uno por petición de cliente. Y, con cada petición, se devuelve un solo documento al cliente. Para crear una aplicación que conste de muchos documentos de cliente, puede invocar Net.Data varias veces para procesar diversos bloques HTML o XML utilizando las técnicas de navegación estándares, por ejemplo enlaces y formularios.

En un bloque HTML o XML puede aparece cualquier sentencia de presentación de texto, a condición de que ésta sea válida para el cliente. Por ejemplo, los bloques HTML pueden contener HTML o JavaScript. Net.Data no ejecuta JavaScript, pero éste se envía junto con el resto de la salida de bloque HTML al cliente para su ejecución y visualización. En un bloque HTML o XML, también puede incluir llamadas de función, referencias de variables y sentencias INCLUDE. El ejemplo siguiente muestra un uso común de un bloque HTML en una macro de Net.Data:

```
%DEFINE DATABASE="MNS96"

%HTML(INPUT){
<h1>Hardware Query Form</h1>
<hr>
<form method="post" action="/cgi-bin/db2www/equip1st.d2w/report">
<d1>
<dt>What hardware do you want to list?
<dd><input type="radio" name="hware" value="MON" checked />Monitors
<dd><input type="radio" name="hware" value="PNT" />Pointing devices
<dd><input type="radio" name="hware" value="PRT" />Printers
```

```

<dd><input type="radio" name="hardware" value="SCN" />Scanners
</dl>
<hr />
<input type="submit" value="Submit" />
</form>
%}

%FUNCTION(DTW_SQL) myQuery() {
SELECT MODNO, COST, DESCRIP FROM EQPTABLE WHERE TYPE=$(hardware)
%REPORT{
<b>Here is the list you requested:</b><br />
%ROW{
<hr>
$(N1): $(V1)      $(N2): $(V2)
<p>
$(V3)
%}
%}
%}

%HTML(report){
@myQuery()
%}

```

Puede invocar la macro de Net.Data desde un enlace HTML.

```

<a href="http://www.ibm.com/cgi-bin/db2www/equiplst.d2w/input">
List of hardware</a>

```

Cuando el usuario de la aplicación pulsa este enlace, el navegador Web invoca Net.Data y Net.Data analiza la macro. Cuando Net.Data empieza a procesar el bloque HTML especificado en la invocación, en este caso HTML(input), empieza a procesar el texto que hay en el bloque. Cualquier elemento que Net.Data no reconozca como una construcción de lenguaje de macro de Net.Data, se envía al navegador para que lo visualice.

Cuando el usuario ha realizado una selección y ha pulsado el botón Submit, Net.Data ejecuta la parte ACTION del elemento HTML FORM, que especifica una llamada al bloque HTML(output) de la macro de Net.Data. Entonces Net.Data procesa el bloque HTML(output) igual que ha hecho con el bloque HTML(input).

A continuación, Net.Data procesa la llamada de función myQuery(), que a su vez invoca el bloque FUNCTION de SQL. Después de sustituir la referencia de variable \$(hardware) de la sentencia SQL por el valor devuelto del formulario de entrada, Net.Data ejecuta la consulta. En este momento, Net.Data reanuda el proceso del informe, visualizando los resultados de la consulta de acuerdo con las sentencias de presentación de texto especificadas en el bloque REPORT.

Cuando Net.Data ha completado del proceso del bloque REPORT, vuelve al bloque HTML(OUTPUT) y finaliza el proceso.

Bloques de informe

Utilice la construcción de lenguaje del bloque REPORT para formatear y visualizar la salida de datos de un bloque FUNCTION. Esta salida consta normalmente de datos de tabla, aunque se pueden especificar combinaciones válidas de texto, referencias de variables de macro y llamadas de función. En el bloque REPORT se puede especificar opcionalmente un nombre de tabla. Excepto para los entornos de lenguaje SQL y ODBC, si no especifica un nombre de tabla, Net.Data utiliza los datos de tabla de la primer tabla de salida de la lista de parámetros FUNCTION.

El bloque REPORT tiene tres partes, cada una de las cuales es opcional:

- Información de cabecera, que contiene texto que se visualiza una vez, antes de los datos de fila de tabla.
- Un bloque ROW, que contiene variables de tabla y texto que se visualizan una vez para cada fila de la tabla de resultados.
- Información de pie de página, que contiene texto que se visualiza una vez, después de los datos de fila de tabla.

Ejemplo:

```
%REPORT{
<h2>Query Results</h2>
<p>Select a name for details.
<table border=1>
  <tr>
    <td>Name</td>
    <td>Location</td></tr>
  %ROW{
    <tr>
      <td>
<a href="/cgi-bin/db2www/name.d2w/details?name=$(V1)&loc;=$(V2)">$(V1)</a>
      </td>
      <td>$(V2)</td>
    </tr>
  %}
</table>
%}
```

Directrices para el bloque REPORT

Al crear bloques REPORT, utilice las directrices siguientes:

- Para evitar visualizar cualquier salida de datos del bloque ROW, deje el bloque ROW vacío u omítalo por completo.
- Utilice variables proporcionadas por Net.Data en el bloque REPORT para acceder a los datos de la tabla de resultados de la macro de Net.Data. Estas variables se describen en el apartado “Variables de proceso de tabla” en la página 131

página 131. Para obtener detalles adicionales, consulte la sección de variables de informe del manual *Net.Data Guía de consulta*.

- Para proporcionar información de cabecera y de pie de página, proporcione el texto antes y después del bloque ROW. Net.Data procesa todo lo que encuentra antes de un bloque ROW como información de cabecera. Net.Data procesa todo lo que encuentra después de un bloque ROW como información de pie de página. Igual que con el bloque HTML, Net.Data trata como sentencias de presentación de texto todos los elementos de los bloques de cabecera, ROW y de pie de página que no se reconocen como construcciones de lenguaje de macro y envía dichas sentencias al navegador.
- Puede llamar a funciones y variables de referencia en un bloque REPORT.
- Para que Net.Data imprima un informe por omisión utilizando texto preformateado, no incluya el bloque REPORT en la macro. El ejemplo siguiente muestra el formato de informe por omisión cuando se llama a la función en un bloque HTML:

SHIPDATE	RECDATE	SHIPNO
25/05/1997	30/05/1997	1495194B
25/05/1997	28/05/1997	2942821G

- Para utilizar los códigos HTML en lugar del texto preformateado, establezca DTW_HTML_TABLE en YES.
 - Para inhabilitar la impresión del informe por omisión, establezca DTW_DEFAULT_REPORT en NO o especifique un bloque REPORT vacío.
- Por ejemplo:

```
%REPORT{%
```

Ejemplo: Personalización de un informe

El ejemplo siguiente muestra cómo se pueden personalizar formatos de informes utilizando variables especiales y códigos HTML. Visualiza los nombres, los números de teléfono y los números de FAX de la tabla CustomerTbl:

```
%DEFINE SET_TOTAL_ROWS="YES"
...
%FUNCTION(DTW_SQL) custlist() {
    SELECT Name, Phone, Fax FROM CustomerTbl
    %REPORT{
<i>Phone Query Results:</i>
<br />
=====
<br />
    %ROW{
Name: <b>$(V1)</b>
<br />
```



```

Phone: $(V2)
<br />
Fax: $(V3)
<br />
-----
<br />
    %}
    Total records retrieved: $(TOTAL_ROWS)
    %}
    %}

```

El informe resultante tiene este aspecto en el navegador Web:

```

Phone Query Results:
=====
Name: Doen, David
Phone: 422-245-1293
Fax: 422-245-7383
-----
Name: Ramirez, Paolo
Phone: 955-768-3489
Fax: 955-768-3974
-----
Name: Wu, Jianli
Phone: 525-472-1234
Fax: 525-472-1234
-----
Total records retrieved: 3

```

Net.Data ha generado el informe realizando lo siguiente:

1. Imprimiendo *Phone Query Results*: una vez al principio del informe. Este texto, junto con la línea de separación, es la parte de cabecera del bloque REPORT.
2. Sustituyendo las variables V1, V2 y V3 por sus valores de Name, Phone y Fax respectivamente para cada fila a medida que se recupera.
3. Imprimiendo la serie *Total records retrieved*: y el valor para TOTAL_ROWS una vez al final del informe.(Este texto es la parte de pie de página del bloque REPORT).

Múltiples bloques REPORT

Puede especificar múltiples bloques REPORT dentro de un solo bloque FUNCTION o MACRO FUNCTION para generar múltiples informes con una sola llamada de función.

Normalmente, se utilizan múltiples bloques REPORT con el entorno de lenguaje DTW_SQL con una función que llama a un procedimiento almacenado, el cual devuelve múltiples conjuntos de resultados (consulte el apartado “Procedimientos almacenados” en la página 171). Sin embargo, se pueden utilizar múltiples bloques REPORT con cualquier entorno de lenguaje para generar múltiples informes.

Para utilizar múltiples bloques REPORT, ponga un nombre de conjunto de resultados en el procedimiento almacenado CALL para cada conjunto de resultados. Si se devuelven más conjuntos de resultados del procedimiento almacenado que el número de bloques REPORT que ha especificado y si la función incorporada de Net.Data DTW_DEFAULT_REPORT = "MULTIPLE", se generarán informes por omisión para cada tabla que no esté asociada con un bloque de informe. Si no se especifican bloques de informe y si DTW_DEFAULT_REPORT = "YES", sólo se generará un informe por omisión. Tenga en cuenta que, para el entorno de lenguaje SQL solamente, un valor DTW_DEFAULT_REPORT de "YES" equivale a un valor de "MULTIPLE".

Ejemplos: Los ejemplos siguientes muestran modos en que se pueden utilizar múltiples bloques de informe.

Visualizar múltiples informes utilizando el formato de informe por omisión:

Ejemplo 1: Entorno de lenguaje DTW_SQL

```
%DEFINE DTW_DEFAULT_REPORT = "MULTIPLE"
%FUNCTION (dtw_sql) myStoredProc () {
    CALL myproc (table1, table2) %}
```

En este ejemplo, el procedimiento almacenado myproc devuelve dos conjuntos de resultados, que se colocan en table1 y table2. Puesto que no se ha especificado ningún bloque REPORT, se visualizan informes por omisión para ambas tablas, primero table1 y, a continuación, table2.

Ejemplo 2: Bloque MACRO_FUNCTION. En este ejemplo, se pasan dos tablas en el bloque MACRO_FUNCTION. Cuando se especifica DTW_DEFAULT_REPORT="MULTIPLE", Net.Data genera informes para ambas tablas.

```
%DEFINE DTW_DEFAULT_REPORT = "MULTIPLE"
%MACRO_FUNCTION multReport (INOUT tablename1, tablename2) {
    %}
```

En este ejemplo, se pasan dos tablas en el multReport de MACRO_FUNCTION. De nuevo, Net.Data visualiza informes por omisión para las dos tablas en el orden en que aparecen en la lista de parámetros del bloque MACRO FUNCTION, primero table1 y, a continuación, table2.

Ejemplo 3: Entorno de lenguaje DTW_REXX

```
%DEFINE DTW_DEFAULT_REPORT = "YES"
%FUNCTION (dtw_rexx) multReport (INOUT table1, table2) {
    SAY 'Generating multiple default reports...<br />'
    %}
```

En este ejemplo, se pasan dos tablas en la función REXX `multReport`. Puesto que se ha especificado `DTW_DEFAULT_REPORT="YES"`, `Net.Data` sólo visualiza un informe por omisión para la primera tabla.

Visualizar múltiples informes especificando bloques `REPORT` para el proceso de visualización:

Ejemplo 1: Bloques `REPORT` con nombre

```
%FUNCTION(dtw_sql) myStoredProc () {
    CALL myproc (table1, table2)

    %REPORT(table2) {
        ...
        %ROW { .... %}
        ...
    %}

    %REPORT(table1) {
        ...
        %row { .... %}
        ...
    %}
%}
```

En este ejemplo, se han especificado bloques `REPORT` para las dos tablas pasadas en la lista de parámetros del bloque `FUNCTION`. Estas tablas se visualizan en el orden en el que se han especificado en los bloques `REPORT`, primero `table2` y, a continuación, `table1`. Si especifica un nombre de tabla en el bloque `REPORT`, puede controlar el orden en el que se visualizan los informes.

Ejemplo 2: Bloques `REPORT` sin nombre

```
%FUNCTION(dtw_sql) myStoredProc () {
    CALL myproc

    %REPORT {
        ...
        %ROW { .... %}
        ...
    %}
    %REPORT {
        ...
        %ROW { .... %}
        ...
    %}
%}
```

En este ejemplo, se han especificado `REPORT` para las dos tablas pasadas en la lista de parámetros del bloque `FUNCTION`. Dado que no se han

especificado nombres de tabla en los bloques REPORT, se visualizan informes para las dos tablas en el orden en el que se devuelven del procedimiento almacenado.

Visualizar múltiples informes utilizando una combinación de informes por omisión y bloques REPORT:

Ejemplo: Combinación de informes por omisión y bloques REPORT

```
%DEFINE DTW_DEFAULT_REPORT = "MULTIPLE"
%FUNCTION(dtw_system) editTables (INOUT table1, table2, table3) {
    %REPORT(table2) {
        ...
        %ROW { .... %}
        ...
    %}
%}
```

En este ejemplo, sólo se especifica un bloque REPORT. Dado que el bloque especifica table2, y table2 es el segundo conjunto de resultados listados en la sentencia CALL, se utiliza el segundo conjunto de resultados para visualizar el informe. Puesto que hay menos bloques REPORT especificados que el número de conjuntos de resultados devueltos del procedimiento almacenado, se visualizan informes por omisión para el resto de conjuntos de resultados: en primer lugar, un informe por omisión para el primer conjunto de resultados, table1 y, a continuación, un informe por omisión para el tercer conjunto de resultados, table3. Se especifica una tabla de salida, table1, que se puede utilizar posteriormente para el proceso en la macro.

Directrices y restricciones para múltiples bloques REPORT: Utilice las directrices y restricciones siguientes al especificar múltiples bloques REPORT en un bloque FUNCTION o MACRO_FUNCTION.

Directrices:

- Puede especificar uno o más bloques REPORT por nombre de conjunto de resultados o de tabla. El nombre especificado para el bloque REPORT debe coincidir con un parámetro de nombre de conjunto de resultados o de nombre de tabla correspondiente en la sentencia CALL o la lista de parámetros del bloque FUNCTION.
- Especifique bloques REPORT para múltiples tablas en el orden en el que desea que se procesen.
- Para especificar el proceso por omisión cuando no hay ningún bloque REPORT especificado para una tabla, defina DTW_DEFAULT_REPORT = "MULTIPLE". Cuando Net.Data crea la página Web, visualiza informes por omisión para tablas después de visualizar los informes para las tablas que tienen bloques REPORT.

- Para evitar que Net.Data visualice tablas que no tienen bloques REPORT, establezca DTW_DEFAULT_REPORT = "NO".
- Cuando utilice la variable DTW_SAVE_TABLE_IN con una función que devuelve más de una tabla, la primera tabla devuelta de la función se asignará a la tabla DTW_SAVE_TABLE_IN.
- Se pueden utilizar múltiples bloques de informe con cualquier entorno de lenguaje.

Restricciones:

- Los valores de todas las variables de informe de una función se aplican a todos los bloques REPORT de dicha función. No se puede modificar el valor de una variable de informe para bloques REPORT individuales.
- El bloque MESSAGE debe estar ubicado antes o después de una lista de bloques REPORT, no entre los bloques REPORT.
- Las variables de tabla deben definirse dentro de la sentencia TABLE antes de pasarse a la función.
- Si el primer bloque de informe especifica un nombre de tabla, todos los bloques de informe deben especificar nombres de tabla.
- Si el primer bloque de informe no especifica un nombre de tabla, ninguno de los bloques de informe puede especificar nombres de tabla.
- El número máximo de tablas para un procedimiento almacenado individual es de 32.

Lógica condicional y repetición en bucle en una macro

Net.Data le permite incorporar la lógica condicional y la repetición en bucle en las macros de Net.Data utilizando los bloques IF y WHILE.

Los bloques IF y WHILE utilizan una lista de condiciones que le ayuda a comprobar una o más condiciones y luego a ejecutar un bloque de sentencias basándose en la salida de la comprobación de condición. La lista de condiciones contiene operadores lógicos, tales como = y <+, y términos, que están compuestos por series entrecomilladas, variables, referencias de variable y llamadas de función. Las series entrecomilladas también pueden contener referencias de variables y llamadas de función. La lista de condiciones puede anidarse.

Las secciones siguientes describen la lógica condicional y la repetición en bucle:

- “Lógica condicional: Bloques IF” en la página 154
- “Construcciones de repetición en bucle: Bloques WHILE” en la página 156

Lógica condicional: Bloques IF

Utilice el bloque IF para el proceso condicional en una macro de Net.Data. El bloque IF es similar a las sentencias IF en la mayoría de los lenguajes de alto nivel porque proporciona la posibilidad de comprobar una o más condiciones y entonces ejecutar un bloque de sentencias basándose en la salida de la comprobación de condición.

Los bloques IF pueden especificarse casi en cualquier lugar de una macro y pueden anidarse. En el capítulo de construcciones de lenguaje del manual *Net.Data Guía de consulta*, se muestra la sintaxis de un bloque IF.

Normas del bloque IF: Las normas para la sintaxis del bloque IF las determina la posición del bloque en la macro. Los elementos permitidos en el bloque ejecutable de sentencias de un bloque IF dependen de la ubicación del propio bloque.

- Cualquier elemento que sea válido en el bloque que contiene el bloque IF será válido dentro de dicho bloque IF. Por ejemplo, si especifica un bloque IF dentro de un bloque HTML, cualquier elemento que esté permitido en el bloque HTML estará permitido en el bloque IF, por ejemplo sentencias INCLUDE y bloques WHILE.

```
%HTML block
...
  %IF block
  ...
    %INCLUDE
  ...
    %WHILE
  ...
    %ENDIF
%}
```

- De modo similar, si especifica el bloque IF fuera de cualquier otro bloque de la parte de declaración de la macro de Net.Data, sólo los elementos permitidos fuera de cualquier otro bloque (por ejemplo, un bloque DEFINE o un bloque FUNCTION) estarán permitidos en el bloque IF.

```
%IF
...
  %DEFINE
...
  %FUNCTION
...
%ENDIF
```

- Cuando un bloque IF se anida dentro de un bloque IF que está fuera de cualquier otro bloque de la parte de declaración, el primero puede utilizar cualquier elemento que pueda utilizar el bloque externo. Cuando un bloque IF se anida dentro de otro bloque que está en un bloque IF, el primero adopta las normas de sintaxis del bloque en el que se encuentra.

Por ejemplo, un bloque IF anidado debe seguir las normas utilizadas cuando está dentro de un bloque HTML.

```
%IF
...
  %HTML {
...
    %IF
...
    %ENDIF
  %}
...
%ENDIF
```

Excepción: No especifique un bloque ROW en un bloque IF.

Comparación de series de bloque IF

Net.Data procesa la lista de condiciones del bloque IF en uno de dos modos basándose en el contenido de los términos que componen las condiciones. La acción por omisión es tratar todos los términos como series y realizar comparaciones de series como se especifica en las condiciones. Sin embargo, si la comparación es entre dos series que representan enteros, la comparación es numérica. Net.Data supone que una serie es numérica si ésta sólo contiene dígitos, precedidos opcionalmente por un carácter '+' o '-'. La serie no puede contener ningún carácter no dígito distinto de '+' o '-'. Net.Data no soporta la comparación numérica de números no enteros.

Ejemplos de series de enteros válidas:

```
+1234567890
-47
000812
92000
```

Ejemplos de series de enteros no válidas:

```
- 20      (contiene caracteres en blanco)
234.000   (contiene un punto)
57,987    (contiene una coma decimal)
```

Net.Data evalúa la condición IF en el momento en el que ejecuta el bloque, que puede ser diferente del momento en el que Net.Data lo lee originalmente. Por ejemplo, si especifica un bloque IF en un bloque REPORT, Net.Data no evalúa la lista de condiciones asociada con el bloque IF cuando lee la definición de bloque FUNCTION que contiene el bloque REPORT, sino que lo hace cuando llama a la función y la ejecuta. Esto es cierto para la parte de lista de condiciones del bloque IF y para el bloque de sentencias que se deben ejecutar.

Ejemplo de bloque IF: Macro que contiene bloques IF dentro de otros bloques

```

%{ This macro is called from another macro, passing the operating system
   and version variables in the form data.
%}

%IF (platform == "AS400")
  %IF (version == "V3R2")
    %INCLUDE "as400v3r2_def.hti"
  %ELIF (version == "V3R7")
    %INCLUDE "as400v3r7_def.hti"
  %ELIF (version == "V4R1")
    %INCLUDE "as400v4r1_def.hti"
  %ENDIF
%ELSE
  %INCLUDE "default_def.hti"
%ENDIF

%MACRO_FUNCTION numericCompare(IN term1, term2, OUT result) {
  %IF (term1 < term2)
    @dtw_assign(result, "-1")
  %ELIF (term1 > term2)
    @dtw_assign(result, "1")
  %ELSE
    @dtw_assign(result, "0")
  %ENDIF
%}

%HTML(report){
  %WHILE (a < "10") {
    outer while loop #$(a)<br />
    %IF (@dtw_rdivrem(a,"2") == "0")
      this is an even number loop<br />
    %ENDIF
    @DTW_ADD(a, "1", a)
  %}
%}

```

Construcciones de repetición en bucle: Bloques WHILE

Utilice el bloque WHILE para realizar la repetición en bucle en una macro de Net.Data. Al igual que el bloque IF, el bloque WHILE proporciona la posibilidad de comprobar una o más condiciones y luego ejecutar un bloque de sentencias basándose en la salida de la comprobación de condición. A diferencia del bloque IF, el bloque de sentencias puede ejecutarse cualquier número de veces basándose en la salida de la comprobación de condición.

Puede especificar bloques WHILE dentro de bloques HTML, bloques REPORT, bloques ROW, bloques MACRO_FUNCTION y bloques IF y puede anidarlos. La sintaxis de un bloque WHILE se muestra en el capítulo de construcciones de lenguaje del manual *Net.Data Guía de consulta*.

Net.Data procesa el bloque WHILE exactamente del mismo modo que procesa el bloque IF, pero vuelve a evaluar la condición después de cada ejecución del

bloque. Y, al igual que cualquier construcción de repetición en bucle condicional, es posible que el proceso entre en un bucle infinito si la condición se codifica incorrectamente.

Ejemplo: Macro con un bloque WHILE

```
%DEFINE loopCounter = "1"

%HTML(build_table) {
  %WHILE (loopCounter <= "100") {
    %{ generate table tag and column headings %}
    %IF (loopCounter == "1")
      <table border>
      <tr>
      <th>Item #
      <th>Description
    %ENDIF

    %{ generate individual rows %}
    <tr>
    <td>$(loopCounter)
    <td>@getDescription(loopCounter)

    %{ generate end table tag %}
    %IF (loopCounter == "100")
    %ENDIF

    %{ increment loop counter %}
    @DTW_ADD(loopCounter, "1", loopCounter)
  %}
%}
```

Capítulo 6. Utilización de entornos de lenguaje

Net.Data proporciona entornos de lenguaje que se utilizan para acceder a fuentes de datos y para ejecutar programas de aplicación que contienen lógica de negocio. Por ejemplo, el entorno de lenguaje SQL le permite pasar sentencias de SQL a una base de datos DB2, y el entorno de lenguaje REXX le permite invocar programas REXX. También puede utilizar el entorno de lenguaje SYSTEM para ejecutar un programa o emitir un mandato.

Con Net.Data, puede añadir entornos de lenguaje escritos por el usuario utilizando algún modo de conexión. Cada entorno de lenguaje escrito por el usuario debe soportar un conjunto estándar de interfaces definidas por Net.Data y debe implementarse como una biblioteca de enlaces dinámicos (DLL) o una biblioteca compartida. Para obtener detalles completos acerca de los entornos de lenguaje proporcionados por Net.Data y sobre cómo crear un entorno de lenguaje escrito por el usuario, consulte el manual *Net.Data Language Environment Interface Reference*.

La Figura 24 en la página 160 muestra la relación entre el servidor Web, Net.Data y los entornos de lenguaje de Net.Data.

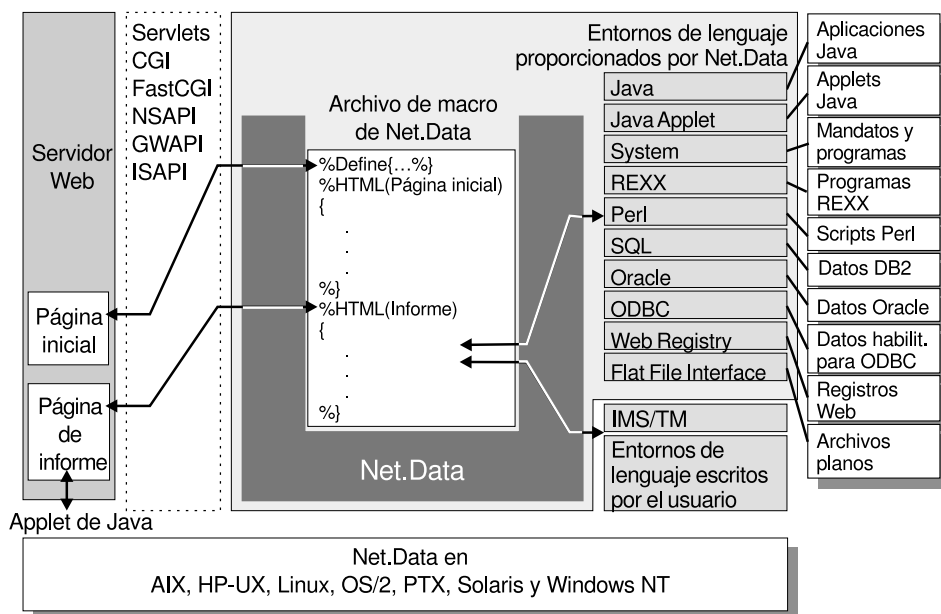


Figura 24. Entornos de lenguaje de Net.Data

Las secciones siguientes describen los entornos de lenguaje de Net.Data y el modo de utilizarlos en las macros:

- “Visión general de los entornos de lenguaje proporcionados por Net.Data” en la página 161
- “Llamada de un entorno de lenguaje” en la página 162
- “Entornos de lenguaje de datos” en la página 163
- “Entornos de lenguaje de programación” en la página 187

Para obtener información de configuración acerca de los entornos de lenguaje proporcionados por Net.Data, consulte el apartado “Configuración de los entornos de lenguaje” en la página 31.

Para obtener información sobre cómo mejorar el rendimiento cuando se utilizan los entornos de lenguaje, consulte el apartado “Optimización de los entornos de lenguaje” en la página 241.

Visión general de los entornos de lenguaje proporcionados por Net.Data

Net.Data proporciona entornos de lenguaje que le permiten acceder a los datos y los recursos de programación para la aplicación.

Net.Data proporciona dos tipos de entornos de lenguaje:

- “Entornos de lenguaje de datos” en la página 163
- “Entornos de lenguaje de programación” en la página 187

La Tabla 7 proporciona una breve descripción de cada entorno de lenguaje. Consulte el apéndice sobre sistemas operativos del manual *Net.Data Guía de consulta* para conocer qué entornos de lenguaje se soportan en diferentes sistemas operativos.

Tabla 7. Entornos de lenguaje de Net.Data

Entorno de lenguaje	Nombre de entorno	Descripción
Flat File Interface	DTW_FILE	La interfaz de archivo plano (flat file interface - FFI) proporciona funciones que soportan archivos de texto como fuentes de datos.
IMS Web	HWS_LE	El entorno de lenguaje IMS Web le permite someter una transacción IMS utilizando IMS Web y recibir la salida de la transacción en el navegador Web.
Java Applet	DTW_APPLET	El entorno de lenguaje de applet Java le permite utilizar applets Java en las aplicaciones Net.Data. Para generar un código de applet, deberá proporcionar los calificadores del código de applet y la lista de parámetros de la applet.
Java Application	DTW_JAVAPPS	Net.Data soporta las aplicaciones Java existentes con el entorno de lenguaje Java.
ODBC	DTW_ODBC	El entorno de lenguaje ODBC ejecuta sentencias de SQL mediante una interfaz ODBC para acceso a múltiples sistemas de gestión de bases de datos.
Oracle	DTW_ORA	El entorno de lenguaje Oracle le permite acceder directamente a los datos de Oracle.
Perl	DTW_PERL	El entorno de lenguaje Perl interpreta los scripts Perl internos que se especifican en un bloque FUNCTION de la macro de Net.Data o ejecuta los scripts Perl externos almacenados en archivos independientes.

Tabla 7. Entornos de lenguaje de Net.Data (continuación)

Entorno de lenguaje	Nombre de entorno	Descripción
REXX	DTW_REXX	El entorno de lenguaje REXX interpreta los programas REXX internos que se especifican en un bloque FUNCTION de la macro de Net.Data o puede ejecutar los programas REXX externos almacenados en un archivo independiente.
SQL	DTW_SQL	El entorno de lenguaje SQL ejecuta sentencias de SQL mediante DB2. Los resultados de la sentencia de SQL se pueden devolver en una variable de tabla.
System	DTW_SYSTEM	El entorno de lenguaje System soporta la ejecución de mandatos y la llamada de programas externos.
Web Registry	DTW_WEBREG	El entorno de lenguaje Web Registry (Registro Web) proporciona funciones para el almacenamiento permanente de datos relacionados con la aplicación.

Llamada de un entorno de lenguaje

Para llamar a un entorno de lenguaje:

- Utilice una sentencia FUNCTION para definir una función que llame al entorno de lenguaje.
- Utilice una llamada de función al entorno de lenguaje.

Por ejemplo:

```
%FUNCTION(DTW_SQL) custinfo() {
  select CUSTNAME, CUSTNO from ibmuser.customer
  %}
...
%HTML(REPORT) {
  @custinfo()
  %}
```

Manejo de condiciones de error

Cuando se detecta un error en una función de entorno de lenguaje, el entorno de lenguaje establece la variable RETURN_CODE de Net.Data con un código de error.

Puede utilizar los recursos siguientes para manejar las condiciones de error:

- Los entornos de lenguaje proporcionados por Net.Data devuelven códigos de error que se documentan en el manual *Net.Data Mensajes y códigos*.
- Los entornos de lenguaje de base de datos, por ejemplo el entorno de lenguaje SQL, establecen la variable RETURN_CODE en códigos de error,

llamados SQLCODE, que el sistema de gestión de bases de datos (DBMS) devuelve. Consulte la documentación de mensajes y códigos del DBMS correspondiente para obtener más información acerca de los SQLCODE utilizados por el DBMS.

Seguridad

Asegúrese de que el ID de usuario bajo el que se está ejecutando Net.Data tiene la autorización correcta para acceder a cualquier objeto al que pueda hacer referencia el destino de una sentencia de entorno de lenguaje. Por ejemplo, el entorno de lenguaje SQL ejecuta sentencias de SQL y las sentencias de SQL acceden a archivos de base de datos, de modo que el ID de usuario bajo el que se está ejecutando Net.Data debe tener autorización para los archivos de base de datos.

Entornos de lenguaje de datos

Los entornos de lenguaje de datos proporcionados por Net.Data le permiten acceder a los datos de bases de datos relacionales y jerárquicas, así como a otras fuentes de datos de una macro de Net.Data. Las secciones siguientes describen los entornos de lenguaje de datos proporcionados por Net.Data y el modo de utilizarlos en las macros de Net.Data.

- “Entornos de lenguaje de bases de datos relacionales”
- “Entorno de lenguaje Flat File Interface” en la página 182
- “Entorno de lenguaje Web Registry” en la página 184

Entornos de lenguaje de bases de datos relacionales

Net.Data proporciona entornos de lenguaje de bases de datos relacionales para ayudarle a acceder a las fuentes de datos relacionales. Las sentencias de SQL que se proporcionan para acceder a los datos relacionales se ejecutan como SQL dinámico. Para obtener más información sobre el SQL dinámico, consulte la documentación de DB2.

Las secciones siguientes describen los entornos de lenguaje y el modo de utilizarlos:

- “Entorno de lenguaje ODBC” en la página 164
- “Entorno de lenguaje Oracle” en la página 164
- “Entorno de lenguaje SQL” en la página 165
- “Gestión de transacciones en una aplicación Net.Data” en la página 166
- “Utilización de objetos grandes” en la página 168
- “Procedimientos almacenados” en la página 171
- “Codificación de los URL de DataLink en los conjuntos de resultados” en la página 178
- “Ejemplo de entorno de lenguaje de base de datos relacionals” en la página 180

Entorno de lenguaje ODBC

El entorno de lenguaje ODBC (Open Database Connectivity - Conectividad de base de datos abierta) ejecuta sentencias de SQL mediante una interfaz ODBC. ODBC se basa en la especificación de X/Open SQL CAE, que permite a una aplicación individual acceder a muchos sistemas de gestión de bases de datos.

Utilización del entorno de lenguaje ODBC:

Para utilizar el entorno de lenguaje ODBC, primero obtenga e instale un controlador ODBC y un gestor de controlador. La documentación del controlador ODBC describe cómo instalar y configurar el entorno ODBC.

Verifique que la sentencia de configuración siguiente esté en el archivo de inicialización de Net.Data, en una sola línea.

```
ENVIRONMENT (DTW_ODBC) d:/net.data/lib/dtwodbc.dll ( IN DATABASE, LOGIN, PASSWORD,  
TRANSACTION_SCOPE, START_ROW_NUM, DTW_SET_TOTAL_ROWS)
```

Restricciones:

- El entorno de lenguaje ODBC sólo soporta procedimientos almacenados cuando se conecta a DB2.
- Cuando especifique la variable DATABASE, deberá especificar la misma base de datos que la fuente de datos del archivo de inicialización de ODBC.
- Las sentencias de SQL del bloque de sentencias incorporadas pueden tener un máximo de 64 KB. DB2 Universal Database tiene las restricciones siguientes:
 - Versión 6 o posterior: 64 KB
 - Versión 5 Release 2 o anterior: 32 KB

La base de datos puede tener límites diferentes; consulte la documentación de la base de datos para determinar si ésta tiene un límite diferente.

Entorno de lenguaje Oracle

El entorno de lenguaje Oracle proporciona acceso nativo a los datos de Oracle. Puede acceder a las bases de datos de Oracle desde Net.Data cuando utilice CGI, FastCGI, NSAPI, ISAPI o GWAPI. Este entorno de lenguaje soporta Oracle 7.2, 7.3 y 8.0.

Para utilizar el entorno de lenguaje Oracle, verifique que la sentencia de configuración siguiente esté en el archivo de inicialización, en una sola línea.

```
ENVIRONMENT (DTW_ORA) /net.data/lib/dtwora.so (IN DATABASE, LOGIN, PASSWORD,  
TRANSACTION_SCOPE, START_ROW_NUM, DTW_SET_TOTAL_ROWS)
```

Consulte el apartado “Configuración del entorno de lenguaje Oracle” en la página 33 para obtener información sobre cómo realizar configuración adicional del entorno de lenguaje Oracle.

Restricciones:

- La variable DATABASE no se utiliza para acceder a las bases de datos de Oracle.
- La variable LOGIN debe contener el nombre de instancia de base de datos de Oracle. Por ejemplo, *ora73* es el nombre de instancia definido en la variable LOGIN siguiente:

LOGON=admin@ora73

- Deberá utilizar Live Connection al utilizar una interfaz distinta de CGI.
- No se soportan los procedimientos almacenados.

Entorno de lenguaje SQL

El entorno de lenguaje SQL proporciona acceso a las bases de datos DB2.

Utilice este entorno de lenguaje para obtener el rendimiento óptimo al acceder a DB2.

Para utilizar el entorno de lenguaje SQL, verifique que la sentencia de configuración siguiente esté en el archivo de inicialización, en una sola línea.

```
ENVIRONMENT (DTW_SQL) d:/net.data/lib/dtwsq1.d11 (IN DATABASE, LOGIN, PASSWORD,  
TRANSACTION_SCOPE, START_ROW_NUM, DTW_SET_TOTAL_ROWS)
```

Sentencias de SQL anidadas

Puede llamar a funciones de SQL que están dentro de otra función de SQL. Si se pasan tablas, asegúrese de que utiliza nombres de tabla exclusivos en cada una de las funciones; de lo contrario, se pueden producir resultados imprevisibles.

Ejemplo: Llama a una función de SQL desde el bloque ROW de otra función de SQL

```
%define mytable1 = %TABLE  
%define mytable2 = %TABLE  
  
%FUNCTION(DTW_SQL) sql2 (IN p1, OUT t2) {  
    select * from NETDATA.STAFFINF where projno='${p1}'  
    %REPORT {  
        %ROW { $(N1) is $(V1) %}  
    %}  
    %}  
  
%FUNCTION(DTW_SQL) sql1 (OUT t1) {  
    select * from NETDATA.STAFFINF  
    %REPORT {  
        %ROW { @sql2(V1, mytable2) %}  
    %}  
    %}  
  
%HTML(netcall1) { @sql1(mytable1) %}
```

Restricciones:

Las sentencias de SQL del bloque de sentencias incorporadas pueden tener un máximo de 64 KB. DB2 Universal Database tiene las restricciones siguientes:

- Versión 6 o posterior: 64 KB
- Versión 5 Release 2 o anterior: 32 KB

Puede que la base de datos tenga límites diferentes; consulte la documentación de la base de datos para determinar si el DBMS tiene un límite diferente.

El SQL anidado sólo se puede utilizar en los entornos de lenguaje SQL o ODBC y no se puede utilizar junto con Live Connection.

Cuando se anidan sentencias de SQL, el número máximo de conjuntos de resultados es de 32. Por ejemplo, puede anidar tres niveles, cada uno de los cuales devolverá 10 conjuntos de resultados. O anidar 32 niveles, devolviendo cada uno un conjunto de resultados.

Gestión de transacciones en una aplicación Net.Data

Cuando se modifica el contenido de una base de datos utilizando sentencias de inserción, supresión o actualización, las modificaciones no son permanentes hasta que la base de datos recibe una sentencia de compromiso de Net.Data. Si se produce un error, Net.Data envía una sentencia de retrotracción a la base de datos, deshaciendo todas las modificaciones efectuadas desde el último compromiso.

El modo en que Net.Data envía el compromiso y la posible retrotracción depende del valor de TRANSACTION_SCOPE y de que las sentencias de compromiso estén especificadas explícitamente en la macro. Los valores para TRANSACTION_SCOPE son MULTIPLE y SINGLE.

SINGLE

Especifica que Net.Data emite una sentencia de compromiso después de cada sentencia de SQL satisfactoria. Si la sentencia de SQL devuelve un error, se emite una sentencia de retrotracción. El ámbito de transacción individual asegura inmediatamente la modificación de la base de datos; sin embargo, con este ámbito, no es posible deshacer una modificación utilizando posteriormente una sentencia de retrotracción.

Para activar este método de compromiso, establezca TRANSACTION_SCOPE en SINGLE. Por ejemplo:

```
@DTW_ASSIGN(TRANSACTION_SCOPE,"SINGLE")
```

MULTIPLE

Especifica que Net.Data ejecutará todas las sentencias de SQL antes de que se emitan una sentencia de compromiso y una posible sentencia de retrotracción. Net.Data envía el compromiso al final de la petición y, si cada sentencia de SQL se emite satisfactoriamente, el compromiso hace que todas las modificaciones efectuadas en la base de datos sean permanentes. Si cualquiera de las sentencias devuelve un error, Net.Data emite una sentencia de retrotracción, que vuelve a establecer la base de datos en su estado original. MULTIPLE es el valor por omisión si no se establece TRANSACTION_SCOPE.

Puede emitir una sentencia de compromiso al final de cualquier sentencia de SQL en la macro utilizando la sentencia de SQL COMMIT. Si deja TRANSACTION_SCOPE establecido en MULTIPLE y emite sentencias de compromiso al final de los grupos de sentencias que cree que están calificadas como transacción, mantendrá, como desarrollador de aplicaciones, el control completo sobre el comportamiento de compromiso y retrotracción en la aplicación. Por ejemplo, la emisión de sentencias de compromiso después de cada actualización en la macro puede ayudar a asegurar la integridad de los datos.

Para emitir una sentencia de compromiso de SQL, puede definir una función a la que puede llamar en cualquier punto del bloque HTML:

```
%FUNCTION(DTW_SQL) user_commit() {  
    commit  
}%  
  
...  
  
%HTML {  
    ...  
    @user_commit()  
    ...  
}%
```

Restricciones:

El valor de TRANSACTION_SCOPE no se puede cambiar después de haber efectuado una conexión a la base de datos. Por consiguiente, todas las transacciones de SQL de una macro están sujetas al mismo ámbito.

Si está utilizando Net.Data como parte de Net.Commerce, tenga que cuenta que Net.Commerce tiene su propio manejo de transacciones e inhabilita el manejo de transacciones de Net.Data.

Utilización de objetos grandes

Puede almacenar archivos de objetos grandes (LOB) en bases de datos de DB2 e incorporarlos en las páginas Web dinámicas utilizando los entornos de lenguaje SQL u ODBC de Net.Data.

Cuando el entorno de lenguaje ejecuta una sentencia SELECT de SQL o un procedimiento almacenado que devuelve un LOB, no asigna el objeto a una variable de proceso de tabla $V(n)$ ni a un campo de tabla de Net.Data. En lugar de ello, almacena el LOB en un archivo creado por Net.Data y devuelve sólo el nombre del archivo en la variable de proceso de tabla $V(n)$ o un campo de tabla de Net.Data. En la macro de Net.Data, puede utilizar el nombre para hacer referencia al archivo LOB; por ejemplo, puede crear un elemento de ancla HTML con una referencia de hipertexto o un elemento de imagen que contenga un URL para el archivo. Net.Data pone el archivo que contiene el LOB en el directorio especificado por la sentencia de vía de acceso HTML_PATH, ubicada en el archivo de inicialización de Net.Data (db2www.ini). El acceso de grabación al archivo LOB está limitado al ID de usuario asociado con la petición de Net.Data que ha recuperado el LOB.

El nombre de archivo para el LOB se crea dinámicamente y tiene el formato siguiente:

nombre[*.extensión*]

Donde:

nombre Es una serie exclusiva generada dinámicamente que identifica el objeto grande

extensión

Es una serie que identifica el tipo del objeto. Para los CLOB y los DBCLOB, la extensión es .txt. Para los BLOB, el entorno de lenguaje SQL determina la extensión buscando una signatura en los primeros bytes del archivo LOB. La Tabla 8 muestra las extensiones LOB utilizadas por el entorno de lenguaje SQL:

Tabla 8. Extensiones LOB utilizadas en el entorno de lenguaje SQL

Extensión	Tipo de objeto
.bmp	imagen de bitmap
.gif	formato gráfico de imagen
.jpg	imagen JPEG (joint photographic experts group)
.tif	formato de archivo de imagen codificado
.ps	postscript
.mid	audio midi (interfaz digital de instrumentos digitales)
.aif	audio AIFF
.avi	audio de interposición visual de audio (audio visual interleave)
.au	audio básico
.ra	audio real

Tabla 8. Extensiones LOB utilizadas en el entorno de lenguaje SQL (continuación)

Extensión	Tipo de objeto
.wav	windows audio visual
.pdf	formato de documento portátil (portable document format)
.rmi	secuencia midi

Si no se reconoce el tipo de objeto para el BLOB, no se añade ninguna extensión al nombre de archivo.

Cuando Net.Data devuelve el nombre del archivo que contiene un LOB, pone como prefijo del nombre de archivo la serie /tmplobs/ utilizando la sintaxis siguiente:

`/tmplobs/nombre.[extensión]`

Este prefijo permite localizar el directorio LOB en un directorio diferente del directorio raíz de documentos del servidor Web.

Para asegurarse de que las referencias a los archivos LOB se resuelven correctamente, añada la directiva Pass siguiente al archivo de configuración del servidor Web:

```
Pass    /tmplobs/*    <víaacceso_html>/tmplobs/*
```

<víaacceso_html> es el valor especificado para la sentencia de vía de acceso HTML_PATH en el archivo de inicialización de Net.Data.

Consejo respecto a la planificación: Cada consulta que devuelve los LOB hace que se creen archivos en el directorio especificado por la variable de configuración de vía de acceso HTML_PATH. Tenga en cuenta las limitaciones del sistema al utilizar los LOB porque éstos pueden consumir recursos rápidamente. Es aconsejable limpiar el directorio periódicamente o ejecutar el daemon dtwclean. Consulte “Gestión de LOB temporales” en la página 171 para obtener más información. Se recomienda que utilice DataLinks, que elimina la necesidad de que el entorno de lenguaje SQL almacene archivos en directorios, lo que hace que el rendimiento sea mejor y que se utilicen muchos menos recursos del sistema.

Ejemplo: La aplicación siguiente utiliza un archivo de audio MPEG (.mpa). Puesto que el entorno de lenguaje SQL no reconoce este tipo de archivo, se utiliza una variable EXEC para añadir la extensión .mpa al nombre de archivo. Un usuario de esta aplicación debe pulsar el nombre de archivo para invocar el visor de archivos de audio MPEG.

```
%DEFINE{
docroot="/usr/lpp/internet/server_root/html"
myFile=%EXEC "rename $(docroot)$(filename) $(docroot)$(filename).mpa"
%}
```

```

%{ donde rename es el mandato del sistema
operativo para red denominar archivos %}
%FUNCTION(DTW_SQL) queryData() {
SELECT Name, IDPhoto, Voice FROM RepProfile
%REPORT{
    <p>Aquí está la información que ha seleccionado:</p>
%ROW{
    @DTW_ASSIGN(filename, @DTW_rSUBSTR(V3, @DTW_rLASTPOS("/", V3)))
    $(myFile)
    $(V1) 
        <a href="$(V3).mpa">Muestra de voz</a><p>
    %}
    %}
%}

%HTML (REPORT) {
@queryData()
%}

```

Si la tabla RepProfile contiene información acerca Kinson Yamamoto y Merilee Lau, la ejecución del bloque REPORT añadirá el HTML siguiente a la página Web que se está generando:

```

<p>Aquí está la información que ha seleccionado:</p>
Kinson Yamamoto 
<a href="/tmplobs/p2345n2.mpa">Muestra de voz</a><p>
Merilee Lau 
<a href="/tmplobs/p2345n4.mpa">Muestra de voz</a><p>

```

El bloque REPORT del ejemplo anterior utiliza las variables de tabla implícitas V1, V2 y V3.

- El valor de V1 es un nombre de persona, que son datos de tipo carácter.
- El valor de V2 es el nombre de un archivo GIF que contiene la fotografía de la persona. La imagen se visualiza incorporada en la página Web generada.
- El valor de V3 es el nombre de un archivo MPA que contiene una muestra de la voz de la persona. Puesto que Net.Data no reconoce el formato de archivo MPA, no añade ninguna extensión al nombre de archivo cuando crea el archivo para el LOB en el directorio especificado por HTML_PATH. Este ejemplo ilustra el uso de una variable EXEC para añadir la extensión .mpa al nombre de archivo. La muestra de voz se reproduce cuando el usuario pulsa en el texto "Muestra de voz", que es un texto de hipertexto.

Derechos de acceso para los LOB:

El directorio por omisión tmplobs para los LOB está bajo el directorio especificado por HTML_PATH en el archivo de inicialización de Net.Data proporcionado con el sistema. Cualquier ID de usuario puede acceder a él. Si se cambia el valor de HTML_PATH, asegúrese de que el ID de usuario bajo el que se está ejecutando el servidor Web tenga acceso de grabación en el

directorio especificado por HTML_PATH (consulte “HTML_PATH” en la página 25 para obtener más información). **Gestión de LOB temporales:**

Net.Data almacena los LOB temporales en un subdirectorio llamado tmplobs, bajo el directorio especificado en la variable de configuración de vía de acceso HTML_PATH. Estos archivos pueden ser grandes y deben borrarse periódicamente para mantener un rendimiento aceptable.

Net.Data proporciona un daemon llamado dtwclean que le ayuda a gestionar periódicamente el directorio tmplobs. dtwclean utiliza el puerto 7127.

Para ejecutar el daemon dtwclean: Entre el mandato siguiente desde la ventana de línea de mandatos:

```
dtwclean [-t xx] [-d|-l]
```

Donde:

- t Es un distintivo que especifica el intervalo en el que dtwclean limpia el directorio
- xx Es el intervalo, en segundos, durante el cual un archivo permanece en el directorio antes que dtwclean lo borre. Este valor no tiene ningún límite. El valor por omisión es de 3600 segundos.
- d Es un distintivo que especifica la modalidad de depuración; se visualiza información de rastreo en la ventana de mandato.
- l Es un distintivo que especifica la modalidad de anotación cronológica; la información de rastreo se imprime en un archivo de anotaciones cronológicas.

Procedimientos almacenados

Un procedimiento almacenado es un programa compilado almacenado en DB2 que puede ejecutar sentencias de SQL. En Net.Data, los procedimientos almacenados se llaman desde las funciones de Net.Data utilizando una sentencia CALL. Los parámetros de procedimiento almacenado se pasan desde la lista de parámetros de función de Net.Data. Puede utilizar procedimientos almacenados para mejorar el rendimiento y la integridad conservando las sentencias de SQL compiladas con el servidor de base de datos. Net.Data soporta la utilización de procedimientos almacenados con DB2 mediante los entornos de lenguaje SQL y ODBC.

Esta sección describe los temas siguientes:

- “Sintaxis de procedimiento almacenado” en la página 172
- “Llamada a un procedimiento almacenado” en la página 173
- “Cómo pasar parámetros” en la página 174
- “Proceso de conjuntos de resultados” en la página 174

Sintaxis de procedimiento almacenado: La sintaxis del procedimiento almacenado utiliza la sentencia FUNCTION, la sentencia CALL y, opcionalmente, un bloque REPORT.

```
%FUNCTION (DTW_SQL) nombre_función ([IN tipodatos arg1, INOUT tipodatos arg2,
    OUT nombretabla, ...]) {
    CALL procedimiento_almacenado [(nombconjresultados, ...)]
    [%REPORT [(nombconjresultados)] { %}]
    ...
    [%REPORT [(nombconjresultados)] { %}]
    [%MESSAGE %]}
%}
```

Donde:

nombre_función

Es el nombre de la función de Net.Data que inicia la llamada del procedimiento almacenado

procedimiento_almacenado

Es el nombre del procedimiento almacenado

tipodatos

Es uno de los tipos de datos de base de datos soportados por Net.Data como se muestra en la Tabla 9. Los tipos de datos especificados en la lista de parámetros deben coincidir con los tipos de datos del procedimiento almacenado. Consulta la documentación de la base de datos para obtener más información sobre estos tipos de datos.

nombretabla

Es el nombre de una tabla de Net.Data en la que se debe almacenar el conjunto de resultados (sólo se utiliza cuando el conjunto de resultados debe almacenarse en una tabla de Net.Data). Si se especifica, este nombre de parámetro debe coincidir con el nombre de parámetro asociado para *nombconjresultados*.

nombconjresultados

Es el nombre que asocia un resultado devuelto de un procedimiento almacenado con un bloque REPORT, con un nombre de tabla de la lista de parámetros de función o con ambos. El *nombconjresultados* de un bloque REPORT debe coincidir con un conjunto de resultados de la sentencia CALL.

Tabla 9. Tipos de datos soportados de procedimientos almacenados

BIGINT	DOUBLEPRECISION	SMALLINT
CHAR	FLOAT	TIME
CLOB ¹	INTEGER	TIMESTAMP
DATE	GRAPHIC	VARCHAR
DECIMAL	LONGVARCHAR	VARGRAPHIC
DOUBLE	LONGVARGRAPHIC	

Tabla 9. Tipos de datos soportados de procedimientos almacenados (continuación)

¹ CLOB sólo se puede utilizar como parámetro OUT e INOUT y Net.Data interpreta el tamaño en bytes. Por ejemplo, si especifica que una variable sea OUT CLOB(20000), se deberá utilizar un CLOB de 20 K de tamaño como parámetro de salida (out).

Llamada a un procedimiento almacenado:

1. Defina un función que inicie una llamada al procedimiento almacenado.

```
%FUNCTION (DTW_SQL) nombre_función()
```

2. Opcionalmente, especifique parámetros IN, INOUT o OUT para el procedimiento almacenado, incluyendo un nombre de variable de tabla para almacenar un conjunto de resultados en una tabla de Net.Data (sólo necesita especificar una tabla de Net.Data si desea que el conjunto de resultados se almacene en dicha tabla). También puede especificar los nombres de tablas o conjuntos de resultados, como parámetros IN o INOUT, de otro procedimiento almacenado.

```
%FUNCTION (DTW_SQL) nombre_función (IN tipodatos  
arg1, INOUT tipodatos arg2,  
OUT nombretabla...)
```

3. Utilice la sentencia CALL para identificar el nombre de procedimiento almacenado.

```
CALL procedimiento_almacenado
```

4. Si el procedimiento almacenado va a generar un conjunto de resultados, especifique opcionalmente un bloque REPORT para definir cómo visualiza Net.Data el conjunto de resultados.

```
%REPORT (nombconjresultados) {  
...  
%}
```

Ejemplo:

```
%FUNCTION (DTW_SQL) mystoredproc (IN CHAR(30) arg1) {  
    CALL myproc  
    %REPORT (mytable){  
        ...  
        %ROW { ... %}  
        ...  
    %}  
%}
```

5. Si el procedimiento almacenado va a generar más de un conjunto de resultados:

- Especifique los nombres de conjunto de resultados en la sentencia CALL.

```
CALL procedimiento_almacenado (nombconjresultados1, nombconjresultados2, ...)
```

- Especifique opcionalmente uno o más bloques REPORT para definir cómo visualiza Net.Data los conjuntos de resultados.

```
%REPORT(nombconjresultados1) {
...
%}
```

Ejemplo:

```
%FUNCTION (DTW_SQL) mystoredproc (IN CHAR(30) arg1, OUT table1) {
  CALL myproc (table1, table2)
  %REPORT(table2) {
    ...
    %ROW { ... %}
    ...
  }
%}
```

Cómo pasar parámetros: Puede pasar parámetros a un procedimiento almacenado y puede hacer que el procedimiento almacenado actualice los valores de los parámetros para que se devuelva el valor nuevo a la macro de Net.Data. El número y el tipo de parámetros de la lista de parámetros de función debe coincidir con el número y el tipo definidos para el procedimiento almacenado. Por ejemplo, si un parámetro de la lista de parámetros que se ha definido para el procedimiento almacenado es INOUT, el parámetro correspondiente de la lista de parámetros de función debe ser INOUT. Si un parámetro de la lista definida para el procedimiento almacenado es de tipo CHAR(30), el parámetro correspondiente de la lista de parámetros de función debe ser también CHAR(30).

Ejemplo 1: Pasar un valor de parámetro al procedimiento almacenado

```
%FUNCTION (DTW_SQL) mystoredproc (IN CHAR(30) valuein) {
  CALL myproc
...
}
```

Ejemplo 2: Devolver un valor de un procedimiento almacenado

```
%FUNCTION (DTW_SQL) mystoredproc (OUT VARCHAR(9) retvalue) {
  CALL myproc
...
}
```

Proceso de conjuntos de resultados: Puede devolver uno o más conjuntos de resultados de un procedimiento almacenado utilizando los entornos de lenguaje SQL u ODBC. Los conjuntos de resultados pueden almacenarse en tablas de Net.Data para procesarse adicionalmente en la macro o procesarse utilizando un bloque REPORT. Si un procedimiento almacenado genera múltiples conjuntos de resultados, deberá asociar un nombre con cada conjunto de resultados generado por el procedimiento almacenado. Esto se lleva a cabo especificando parámetros en la sentencia CALL. Entonces el nombre que especifique para un conjunto de resultados puede asociarse con un bloque REPORT o una tabla Net.Data, lo que le permite determinar cómo procesa Net.Data cada conjunto de resultados. Puede:

- Hacer que el resultado se procese en el estilo de informe por omisión de Net.Data, no definiendo un bloque de informe para el conjunto de resultados.
- Asociar un conjunto de resultados con un bloque REPORT para aplicar su propio estilo de informe. En el bloque REPORT, puede utilizar variables de Net.Data, sentencias de proceso de texto, como HTML o JavaScript, u otras funciones para especificar cómo se visualizan los datos de informe en el navegador.
- Almacenar los conjuntos de resultados en tablas de Net.Data cuando desee que Net.Data utilice los datos posteriormente en la macro. Por ejemplo, puede pasar la tabla de Net.Data a otra función para que ésta pueda utilizar los datos para cálculos y visualizar los resultados basándose en dichos cálculos.

Consulte “Directrices y restricciones para múltiples bloques REPORT” en la página 152 para conocer las directrices y las restricciones al utilizar múltiples bloques de informe.

Para utilizar un solo conjunto de resultados y utilizar el informe por omisión:

Utilice la sintaxis siguiente:

```
%FUNCTION (DTW_SQL) nombre_función
() {
    CALL procedimiento_almacenado
}%
```

Por ejemplo:

```
%FUNCTION (DTW_SQL) mystoredproc() {
    CALL myproc
}%
```

Para devolver un solo conjunto de resultados y especificar un bloque REPORT:

Utilice la sintaxis siguiente:

```
%FUNCTION (DTW_SQL) nombre_función
() {
    CALL procedimiento_almacenado [(nombconjresultados)]
    %REPORT [(nombconjresultados)] {
        ...
    }
}%
```

Por ejemplo:

```
%FUNCTION (DTW_SQL) mystoredproc () {
    CALL myproc
%REPORT {
    ...
    %ROW { ... %}
    ...
}%
%}
```

Alternativamente, se puede utilizar la sintaxis siguiente:

```
%FUNCTION (DTW_SQL) nombre_función () {
    CALL procedimiento_almacenado (nombconjresultados)

%REPORT (nombconjresultados) {
    ...
}%
%}
```

Por ejemplo:

```
%FUNCTION (DTW_SQL) mystoredproc () {
    CALL myproc (mytable1)
%REPORT (mytable1) {
    ...
    %ROW { ... %}
    ...
}%
%}
```

Para almacenar un solo conjunto de resultados en una tabla de Net.Data para procesarlo adicionalmente:

Utilice la sintaxis siguiente:

```
%FUNCTION (DTW_SQL) nombre_función (OUT nombretabla) {
    CALL procedimiento_almacenado (nombconjresultados)
}%
```

Por ejemplo:

```
%DEFINE DTW_DEFAULT_REPORT = "NO"

%FUNCTION (DTW_SQL) mystoredproc (OUT mytable1) {
    CALL myproc (mytable1)
}%
```

Observe que DTW_DEFAULT_REPORT se establece en NO para que no se genere un informe por omisión para el conjunto de resultados.

Para devolver múltiples conjuntos de resultados y visualizarlos utilizando el formato de informe por omisión:

Utilice la sintaxis siguiente:

```
%FUNCTION (DTW_SQL) nombre_función () {
    CALL procedimiento_almacenado [(nombconjresultados1, nombconjresultados2, ...)]
%}
```

Donde no se especifica ningún bloque de informe.

Por ejemplo:

```
%DEFINE DTW_DEFAULT_REPORT = "YES"
%FUNCTION (DTW_SQL) mystoredproc () {
    CALL myproc
%}
```

Para devolver múltiples conjuntos de resultados y hacer que éstos se almacenen en tablas de Net.Data para procesarlos adicionalmente:

Utilice la sintaxis siguiente:

```
%FUNCTION (DTW_SQL) nombre_función (OUT nombretabla1, nombretabla2, ...) {
    CALL procedimiento_almacenado (nombconjresultados1, nombconjresultados2, ...)
%}
```

Por ejemplo:

```
%DEFINE DTW_DEFAULT_REPORT = "NO"

%FUNCTION (DTW_SQL) mystoredproc (OUT mytable1, mytable2) {
    CALL myproc (mytable1, mytable2)
%}
```

Observe que DTW_DEFAULT_REPORT se establece en NO para que no se genere un informe por omisión para los conjuntos de resultados.

Para devolver múltiples conjuntos de resultados y especificar bloques REPORT para el proceso de visualización:

Cada conjunto de resultados se asocia con un bloque REPORTo con varios.

Utilice la sintaxis siguiente:

```
%FUNCTION (DTW_SQL) nombre_función (, ...) {
    CALL procedimiento_almacenado (nombconjresultados1, nombconjresultados2, ...)
    %REPORT (nombretabla1)
        ...
        %ROW { ... %}
        ...
    %}
    %REPORT (nombretabla2)
        ...
        %ROW { ... %}
        ...
    %}
    ...
%}
```

Por ejemplo:

```
%FUNCTION (DTW_SQL) mystoredproc () {  
    CALL myproc (mytable1, mytable2)  
  
    %REPORT(mytable1) {  
        ...  
        %ROW { ... %}  
        ...  
    %}  
  
    %REPORT(mytable2) {  
        ...  
        %ROW { ... %}  
        ...  
    %}  
%}
```

Para devolver múltiples conjuntos de resultados y especificar opciones de visualización o proceso diferentes para cada conjunto de resultados:

Puede especificar opciones de proceso diferentes para cada conjunto de resultados utilizando nombres de parámetro exclusivos. Por ejemplo:

```
%FUNCTION (DTW_SQL) mystoredproc (OUT mytable2) {  
    CALL myproc (mytable1, mytable2, mytable3)  
  
    %REPORT(mytable1)  
    ...  
    %ROW { ... %}  
    ...  
    %}  
%}
```

El conjunto de resultados mytable1 lo procesa el bloque REPORT correspondiente y se visualiza como si lo hubiera especificado la persona que ha escrito la macro. El conjunto de resultados mytable2 se almacena en la tabla de Net.Data mytable2 y se puede utilizar ahora para procesarlo adicionalmente, por ejemplo pasarlo a otra función. El conjunto de resultados mytable3 se visualiza utilizando el formato de informe por omisión de Net.Data porque no se ha especificado ningún bloque REPORT para el mismo.

Codificación de los URL de DataLink en los conjuntos de resultados

El tipo de datos DataLink es uno de los bloques de construcción básicos para ampliar los tipos de datos que pueden almacenarse en archivos de base de datos. Con DataLink, los datos reales almacenados en la columna son sólo un puntero al archivo. Este archivo puede ser de cualquier tipo; un archivo de imagen, una grabación de voz o un archivo de texto. DataLink almacena un URL para resolver la ubicación del archivo.

El tipo de datos DATALINK requiere que se utilice DataLink File Manager. Para obtener más información sobre DataLink File Manager, consulte la documentación de DataLink para el sistema operativo correspondiente. Antes de utilizar el tipo de datos DATALINK, deberá asegurarse de que el servidor Web tiene acceso al sistema de archivos gestionado por el DB2 File Manager Server.

Cuando una consulta de SQL devuelve un conjunto de resultados con DataLink, y la columna de DataLink se crea con las opciones FILE LINK CONTROL y con READ PERMISSION DB de DataLink, las vías de acceso de archivo de la columna de DataLink contienen un símbolo de acceso. DB2 utiliza el símbolo de acceso para autenticar el acceso al archivo. Sin este símbolo de acceso, todos los intentos de acceso al archivo fallarán con una violación de autorización. Sin embargo, el símbolo de acceso puede incluir caracteres que no se pueden utilizar en un URL a devolver a un navegador, por ejemplo el carácter de punto y coma (;). Por ejemplo:

```
/datalink/pics/UN1B;0YPVKG346KEBE;baibien.jpg
```

El URL no es válido porque contiene caracteres de punto y coma (;). Para que el URL sea válido, los puntos y coma deben codificarse utilizando la función incorporada de Net.Data DTW_URLESCSEQ. Sin embargo, deberá realizarse una manipulación de serie antes de aplicar esta función porque esta función también codifica barras inclinadas (/).

Puede escribir una MACRO_FUNCTION de Net.Data para automatizar la manipulación de serie y utilizar la función DTW_URLESCSEQ. Utilice esta técnica en cada macro que recupere datos de una columna de tipo de datos DATALINK.

Ejemplo 1: MACRO_FUNCTION que automatiza la codificación de los URL devueltos de DB2 UDB

```
%{ TO DO: Apply DTW_URLESCSEQ to a DATALINK URL to make it a valid URL.
    IN: DATALINK URL from DB2 File Manager column.
    RETURN: The URL with token portion is URL encoded
%}
%MACRO_FUNCTION encodeDataLink(in DLURL) {
    @DTW_rCONCAT( @DTW_rDELSTR( DLURL,
    @DTW_rADD(@DTW_rLASTPOS("/", DLURL), "1" ) ),
    @DTW_rURLESCSEQ( @DTW_rSUBSTR(DLURL,
    @DTW_rADD( @DTW_rLASTPOS("/", DLURL), "1" ) ) ) )
%}
```

Después de utilizar esta MACRO_FUNCTION, se codifica el URL correctamente y se puede hacer referencia al archivo especificado en la columna DATALINK en cualquier navegador Web.

Ejemplo 2: Macro de Net.Data que especifica la consulta de SQL que devuelve el URL DATALINK

```
%FUNCTION(DTW_SQL) myQuery(){
    select name, DLURLCOMPLETE(picture) from myTable where name like '%river%'
%REPORT{
%ROW{
    <p> $(V1) <br />
    Before Encoding: $(V2) <br />
    After Encoding: @encodeDataLink($(V2)) <br />
    Make HREF: <a href="@encodeDataLink($(V2))"> click here </a> <br /> <p>
    %}
%}
%}
```

Observe que se utiliza una función de DataLink File Manager. La función `dlurlcomplete` devuelve un URL completo.

Ejemplo de entorno de lenguaje de base de datos relacionales

Los ejemplos siguientes muestran cómo puede llamar a los entornos de lenguaje de bases de datos relacionales desde las macros:

ODBC

El ejemplo siguiente define y llama a múltiples funciones para el entorno de lenguaje ODBC.

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

%function(dtw_odbc) sql1() {
    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()
%}
```

Oracle El ejemplo siguiente muestra una macro con una definición de función DTW_ORA que consulta la base de datos de Oracle, udatabase, utilizando una referencia de variable para determinar la tabla de base de datos que se debe consultar. El bloque FUNCTION también contiene un bloque MESSAGE que maneja las condiciones de error. Cuando Net.Data procesa la macro, visualiza un informe por omisión en el navegador.

```
%DEFINE {
    LOGIN="ulogin"
    PASSWORD="upassword"
    DATABASE=""
    table= "utable"
%}

%FUNCTION(DTW_ORA) myQuery(){
select ename,job,empno,hiredate,sal,deptno from $(table) order by ename
%}
%MESSAGE{
100 : "<b>WARNING</b>: No employee were found that met your search criteria.<p>
      : continue
%}

%HTML (REPORT) {
@myQuery()
%}
```

SQL

El ejemplo siguiente muestra una macro con una definición de función DTW_SQL que llama a un procedimiento almacenado de SQL. Tiene tres parámetros de tipos de datos diferentes. El entorno de lenguaje DTW_SQL pasa cada parámetro al procedimiento almacenado de acuerdo con el tipo de datos del parámetro. Cuando el procedimiento almacenado completa el proceso, se devuelven parámetros de salida y Net.Data actualiza las variables como corresponde.

```
%{*****
      DEFINE BLOCK
*****%}
%DEFINE {
    MACRO_NAME      = "TEST ALL TYPES"
    DTW_HTML_TABLE  = "YES"
    Procedure        = "TESTTYPE"
    parm1            = "1"                %{SMALLINT      %}
    parm2            = "11"               %{INT            %}
```

```

parm3          = "1.1"          %{DECIMAL (2,1) %}
%}

%FUNCTION(DTW_SQL)    myProc
  (INOUT SMALLINT    parm1,
   INOUT INT          parm2,
   INOUT DECIMAL(2,1) parm3){
CALL $(Procedure)
%}
%HTML(REPORT) {
<head>
<title>Net.Data : SQL Stored Procedure: Example '$(MACRO_NAME)'. </title>
</head>
<body bgcolor="#bbffff" text="#000000" link="#000000">
<p><p>
Calling the function to create the stored procedure.
<p><p>
  @CRTPROC()
<hr>
<h2>
Values of the INOUT parameters
prior to calling the stored procedure:<p>
</h2>
<b>parm1 (SMALLINT)</b><br />
$(parm1)<p>
<b>parm2 (INT)</b><br />
$(parm2)<p>
<b>parm3 (DECIMAL)</b><br />
$(parm3)<p>
<p>
<hr>
<h2>
Calling the function that executes the stored procedure.
</h2>
<p><p>
  @myProc(parm1,parm2,parm3)
<hr>
<h2>
Values of the INOUT parameters after
calling the stored procedure:<p>
</h2>
<b>parm1 (SMALLINT)</b><br />
$(parm1)<p>
<b>parm2 (INT)</b><br />
$(parm2)<p>
<b>parm3 (DECIMAL)</b><br />
$(parm3)<p>
</body>
%}

```

Entorno de lenguaje Flat File Interface

Si elige utilizar archivos planos (o archivos de texto simple) como fuente de datos, utilice la FFI (flat file interface - interfaz de archivo plano) y sus funciones asociadas para abrir, cerrar, leer, grabar y suprimir archivos en el

servidor Web. El soporte de lenguaje de archivo utiliza las funciones FFI para leer o grabar en archivos del servidor Web a petición del cliente Web mediante el navegador. FFI considera el archivo como un archivo de registro, en el que cada registro equivale a una fila de una variable de tabla de macro de Net.Data y cada valor de un registro equivale a un valor de campo de una variable de tabla de macro de Net.Data. FFI lee los registros de un archivo en las filas de una tabla de macro de Net.Data y graba las filas de una tabla en los registros.

Consulte el manual *Net.Data Guía de consulta* para obtener la descripción y la sintaxis de las funciones incorporadas FFI.

Configuración del entorno de lenguaje FFI

Verifique que la sentencia de configuración siguiente esté en el archivo de inicialización, en una sola línea:

```
ENVIRONMENT (DTW_FILE) DTWFILE ( OUT RETURN_CODE )
```

Consulte el apartado “Sentencias de configuración de entorno” en la página 28 para obtener más información acerca del archivo de inicialización de Net.Data y las sentencias ENVIRONMENT de entorno de lenguaje.

Llamada de funciones incorporadas FFI

Llame a una función FFI igual que llamaría a cualquier otra función. Utilice una sentencia DEFINE para definir como variables cualquiera de los parámetros que desee pasar; por ejemplo:

```
%DEFINE {
    myFile = "c:/private/myfile"
    myTable = %TABLE
    myWait = "1500"
    myRows = "2"
%}
```

A continuación, utilice una sentencia de llamada de función para invocar la función; por ejemplo:

```
@DTWF_UPDATE(myFile, "Delimited", "|", myTable, myWait, myRows)
```

Ejemplo

En este ejemplo, Net.Data lee el contenido del archivo ffi001.dat en una tabla de Net.Data y graba el contenido de esta tabla en el archivo tmp.dat. Finalmente, Net.Data suprime el archivo tmp.dat.

```
%DEFINE {
mytable = %TABLE(ALL)
myfile = "/usr/lpp/netdata/ffi/ffi001.dat"
tmpfile = "/usr/lpp/netdata/ffi/tmp.dat"
%}
%HTML(report) {
@DTWF_READ(myfile, "ASCIITEXT", " ", mytable)
@DTW_TB_TABLE(mytable)
```

```
@DTWF_WRITE(tmpfile, "ASCIITEXT", " ", mytable)
@DTW_TB_TABLE(mytable)

@DTWF_REMOVE(tmpfile)
%}
```

Entorno de lenguaje Web Registry

El registro Web (Web registry) de Net.Data proporciona almacenamiento permanente para los datos relacionados con la aplicación. Se puede utilizar un registro Web para almacenar información de configuración y otros datos a los que se puede acceder dinámicamente durante la ejecución las aplicaciones basadas en Web. Sólo se puede acceder a los registros Web mediante las macros de Net.Data, utilizando Net.Data y el soporte incorporado de registro Web, y desde los programas CGI escritos con esta finalidad. El registro Web está disponible en un subconjunto de sistemas operativos. Consulte el manual *Net.Data Guía de consulta* para obtener la descripción y la sintaxis de la función incorporada de registro Web, así como una lista de sistemas operativos que soportan el entorno de lenguaje.

El desarrollo de páginas Web estándares requiere que los URL se coloquen directamente en la fuente HTML para la página. Esto dificulta el cambio de enlaces. La naturaleza estática también limita el tipo de enlaces que se pueden poner fácilmente en una página Web. La utilización de un registro Web para almacenar datos relacionados con aplicaciones, por ejemplo los URL, puede ayudar en la creación de páginas HTML con enlaces establecidos dinámicamente.

Los desarrolladores de aplicaciones pueden almacenar la información y mantenerla en un registro, así como los administradores de Web que tienen acceso de grabación al registro. Las aplicaciones recuperan la información de los registros asociados durante la ejecución. Esto facilita el diseño de aplicaciones flexibles y también permite el movimiento de aplicaciones y servidores. Puede utilizar macros de Net.Data para crear páginas HTML utilizando enlaces establecidos dinámicamente.

La información se almacena en un registro Web en forma de entradas de registro. Cada entrada de registro consta de un par de series de caracteres: una serie RegistryVariable (variable de registro) y una serie RegistryData (datos de registro) correspondiente. Se puede almacenar como entrada de registro cualquier información que pueda representarse mediante un par de series. Net.Data utiliza la serie variable como clave de búsqueda para localizar y recuperar entradas específicas de un registro.

La Tabla 10 visualiza un registro Web de ejemplo:

Tabla 10. Registro Web de ejemplo

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

Razones por las que considerar la posibilidad de utilizar un registro Web:

- Puede utilizar un registro Web para almacenar alias para los servidores y los URL, facilitando el cambio de ubicación de las aplicaciones y los servidores.
- Los desarrolladores de aplicaciones pueden enviar sus aplicaciones basadas en Web con datos, por ejemplo URL, predefinidos en el registro. El usuario final puede modificar los datos de registro para cambiar el comportamiento de la aplicación.
- Un registro Web puede utilizarse para efectuar búsquedas de URL basadas en el nombre de producto, el idioma nacional, el fabricante, etc.

Las entradas indexadas del registro Web son entradas cuyas series RegistryVariable tienen añadida una serie Index (índice) adicional, utilizando la sintaxis siguiente:

RegistryVariable/Index

El usuario proporciona el valor de la serie de índice en un parámetro independiente en una función incorporada diseñada para trabajar con entradas indexadas. Varias entradas de registro indexadas pueden tener el mismo valor de serie RegistryVariable, pero pueden mantener su exclusividad teniendo valores de serie Index diferentes.

Tabla 11. Registro Web indexado de ejemplo

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

Aunque las dos entradas indexadas anteriores tienen el mismo valor de serie RegistryVariable Smith, la serie de índice es diferente en cada caso. Las funciones de registro Web las tratan como dos entradas diferenciadas.

Configuración del entorno de lenguaje Web Registry

Verifique que la sentencia de configuración siguiente esté en el archivo de inicialización, en una sola línea:

```
ENVIRONMENT (DTW_WEBREG) DTWWEB ( OUT RETURN_CODE )
```

Consulte el apartado “Sentencias de configuración de entorno” en la página 28 para obtener más información acerca del archivo de inicialización de Net.Data y las sentencias ENVIRONMENT de entorno de lenguaje.

Llamada de funciones incorporadas de Web Registry

Llame a una función Web Registry igual que llamaría a cualquier otra función. Utilice una sentencia DEFINE para definir como variables cualquiera de los parámetros que desee pasar. Por ejemplo:

```
%DEFINE {  
    name = "smith"  
%}
```

A continuación, utilice una sentencia de llamada de función para invocar la función; por ejemplo:

```
@DTWR_ADDENTRY("URLLIST", name, "http://www.ibm.com/software/",  
    "WORK_URL")
```

Ejemplo

El ejemplo siguiente crea un registro Web y añade entradas. A continuación, visualiza un informe que contiene las entradas.

```
%DEFINE {  
    RegTable = %TABLE(ALL)  
%}  
  
%MESSAGE {  
    default:"<p>Function Error: Return code: $(RETURN_CODE)." :continue  
%}  
  
%FUNCTION(DTW_WEBREG) ListTable(INOUT RegTable) {  
%}  
  
%HTML(report) {  
    @DTWR_CREATEREG("MYREG")  
    @DTWR_ADDENTRY("MYREG", "Dept. 1", "Payroll")  
    @DTWR_ADDENTRY("MYREG", "Dept. 2", "Technical Support")  
    @DTWR_ADDENTRY("MYREG", "Dept. 3", "Research")  
    @DTWR_LISTREG("MYREG", RegTable)  
  
    <p>Report:<br />  
    @ListTable(RegTable)  
  
%}
```

Entornos de lenguaje de programación

Net.Data proporciona los entornos de lenguaje siguientes para que los utilice al llamar a programas externos:

- “Entorno de lenguaje Java Applet”
- “Entorno de lenguaje Java Application” en la página 195
- “Entorno de lenguaje Perl” en la página 198
- “Entorno de lenguaje REXX” en la página 201
- “Entorno de lenguaje System” en la página 205

Derechos de acceso: Asegúrese de que el ID de usuario bajo el que se ejecuta Net.Data tiene derechos de acceso para ejecutar programas, incluidos los objetos a los que los programas puedan acceder. Consulte el apartado “Cómo otorgar derechos de acceso a los archivos a los que accede Net.Data” en la página 65 para obtener más información.

Entorno de lenguaje Java Applet

El entorno de lenguaje de applet Java (Java Applet) le permite generar fácilmente códigos HTML para applets Java en las aplicaciones de Net.Data. Al llamar al entorno de lenguaje de applet Java, especifique el nombre de la applet y pase los parámetros que necesite la applet. El entorno de lenguaje procesa la macro y genera los códigos de applet HTML, que el navegador Web utiliza para ejecutar la applet.

Adicionalmente, Net.Data proporciona un conjunto de interfaces que la applet puede utilizar para acceder a los parámetros de tabla. Estas interfaces están contenidas en la clase DTW_Applet.class.

Las secciones siguientes describen cómo utilizar el entorno de lenguaje de applet Java para ejecutar las applets Java.

Configuración del entorno de lenguaje Java Applet

Verifique que la sentencia de configuración siguiente esté en el archivo de inicialización, en una sola línea:

```
ENVIRONMENT (DTW_APPLET) DTWJAVA (OUT RETURN_CODE )
```

Consulte el apartado “Sentencias de configuración de entorno” en la página 28 para obtener más información acerca del archivo de inicialización de Net.Data y las sentencias ENVIRONMENT de entorno de lenguaje.

Creación de applets Java

Antes de utilizar el entorno de lenguaje de applet Java de Net.Data, necesita determinar qué applets piensa utilizar o qué applets necesita escribir. Consulte la documentación de Java para obtener más información sobre cómo crear applets.

Generación de códigos de applet

Una llamada al entorno de lenguaje de applet se especifica con una llamada de función de Net.Data. No se necesita ninguna declaración para la llamada de función. A continuación se muestra la sintaxis para la llamada de función:

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

- DTWA_ identifica la llamada de función al entorno de lenguaje de applet.
- AppletName es el nombre de la applet para la que se generan los códigos.
- parm1 a parmN son parámetros utilizados para generar códigos PARAM.

Para escribir una macro que genere códigos de applet:

1. Defina los parámetros necesarios para la applet en la sección DEFINE de la macro. Estos parámetros incluyen cualquier atributo de código de applet, variables de Net.Data y parámetros de tabla de Net.Data que necesite como entrada para la applet. Por ejemplo:

```
%define{
DATABASE = "celdial"                <=Nombre de la base de datos
MyGraph.codebase = "/netdata-java/" <=Atributo de applet necesario
MyGraph.height = "200"              <=Atributo de applet necesario
MyGraph.width = "400"               <=Atributo de applet necesario
MyTitle = "Celdial results"         <=Nombre de la página Web
MyTable = %TABLE(all)              <=Tabla para almacenar resultados
                                   de consulta
%}
```

2. Opcional: Especifique una consulta a la base de datos para generar un conjunto de resultados como entrada para la applet. Esto es útil cuando se está utilizando una applet que genera un gráfico o una tabla. Por ejemplo:

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

3. Especifique la llamada de función en la macro Net.Data para llamar al entorno de lenguaje de applet Java e invocar la applet. La llamada de función especifica el nombre de la applet y los parámetros que desea pasar al entorno de lenguaje. Estos parámetros incluyen las variables de Net.Data y los parámetros de tabla o columna de Net.Data que necesite como entrada para la applet.

Por ejemplo:

```
%HTML(report){
@mySQL(MyTable)                                <=Una llamada a mySQL
@DTWA_MyGraph(MyTitle, DTW_COLUMN(ages) MyTable) <=Llamada de función de applet
%}
```

Atributos de código de applet: Puede especificar atributos para los códigos de applet en cualquier lugar de la macro de Net.Data. Net.Data sustituye todas las variables que tiene el formato *NombreApplet.atributo* en el código de applet como atributos. A continuación, se muestra la sintaxis para definir un atributo en un código de applet:


```
%define NombreApplet.atributo = "valor"
```

Los atributos siguientes son necesarios para todas las applets:

- *codebase*: Ubicación de la applet, que se identifica mediante un URL.
- *height*: Altura de la applet en pixels.
- *width*: Anchura de la applet en pixels.

Los atributos siguientes son opcionales:

- *align*: alineación de la applet
- *alt*: cualquier texto que deba visualizarse si el navegador interpreta el código APPLET pero no puede ejecutar applets Java
- *archive*: archivo que contiene clases y otros recursos
- *hspace*: número de pixels en cada lado de la applet
- *name*: nombre para la instancia de applet
- *object*: nombre del archivo que contiene una representación en serie de una applet
- *vspace*: número de pixels por encima y por debajo de la applet

Por ejemplo, si la applet se llama MyGraph, puede definir estos atributos necesarios como se muestra a continuación:

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

La asignación real no necesita estar en una sección DEFINE. Puede establecer el valor con la función DTW_ASSIGN. Si no define una variable para la variable *NombreApplet.código*, Net.Data añade un parámetro *code* por omisión al código de la applet. El valor del parámetro *code* es *NombreApplet.class*, donde *NombreApplet* es el nombre de la applet.

Parámetros de códigos de applet: Defina una lista de parámetros para pasarlos al entorno de lenguaje de applet Java en la llamada de función. Puede pasar parámetros que incluyan:

- Variables de Net.Data (incluidas las variables LIST)
- Tablas de Net.Data
- Columnas de tablas de Net.Data

Al pasar un parámetro, Net.Data crea un código PARAM de applet Java en la salida HTML con el nombre y valor que se asigna al parámetro. No se pueden pasar literales de serie o resultados de llamadas de función.

Parámetros de variables de Net.Data:

Puede utilizar las variables de Net.Data como parámetros. Si define una variable en el bloque DEFINE de la macro y pasa el valor de la variable en la llamada de función DTWA_NombreApplet, Net.Data genera un código PARAM que tiene el mismo nombre y valor que la variable. Por ejemplo, dada la sentencia de macro siguiente:

```
%define{  
  
...  
  
MyTitle = "This is my Title"  
%}  
  
%HTML(report){  
@DTWA_MyGraph( MyTitle, ...)  
%}
```

Net.Data produce el código PARAM de applet siguiente:

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

Parámetros de tabla de Net.Data:

Net.Data genera automáticamente un código PARAM con el nombre DTW_NUMBER_OF_TABLES cada vez que se llama al entorno de lenguaje de applet Java, especificando si la llamada de función ha pasado variables de tabla. El valor es el número de variables de tabla que Net.Data utiliza en la función. Si no se especifican variables de tabla en la llamada de función, se genera el código siguiente:

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

Puede pasar una o más variables de tabla de Net.Data como parámetros en la llamada de función. Si especifica una variable de tabla de Net.Data en una llamada de función DTWA_NombreApplet, Net.Data genera los códigos PARAM siguientes:

Código de parámetro de nombre de tabla:

Este código especifica los nombres de las tablas a pasar. El código tiene la sintaxis siguiente:

```
<param name = 'DTW_TABLE_i_NAME' value = "nombret" />
```

Donde *i* es el número de la tabla basado en el orden de la llamada de función y *nombret* es el nombre de la tabla.

Códigos de parámetros de especificación de filas y columnas:

Los códigos PARAM se generan para especificar el número de filas y columnas para una tabla determinada. Este código tiene la sintaxis siguiente:

```
<param name = 'DTW_nombret_NUMBER_OF_ROWS' value = "filas" />
<param name = 'DTW_nombret_NUMBER_OF_COLUMNS' value = "cols" />
```

Donde el nombre de la tabla es *nombret*, *filas* es el número de filas de la tabla y *cols* es el número de columnas de la tabla. Este par de códigos se genera para cada tabla exclusiva especificada en la llamada de función.

Códigos de parámetros de valor de columna:

Este código PARAM especifica el nombre de una columna determinada. Este código tiene la sintaxis siguiente:

```
<param name = 'DTW_nombret_COLUMN_NAME_j' value = "nombrec" />
```

Donde el nombre de tabla es *nombret*, *j* es el número de columna y *nombrec* es el nombre de la columna de la tabla.

Códigos de parámetros de valor de fila:

Este código PARAM especifica los valores de una fila y columna determinadas. Este código tiene la sintaxis siguiente:

```
<param name = 'DTW_nombret_nombrec_VALUE_k' value = "val" />
```

Donde el nombre de tabla es *nombret*, *nombrec* es el nombre de columna, *k* es el número de fila y *val* es el valor que coincide con el valor de la fila y la columna correspondientes.

Parámetros de columna de tabla: Puede pasar una columna de tabla como un parámetro en una llamada de función para generar códigos para una columna específica. Net.Data sólo genera los códigos de applet correspondientes para la columna especificada. Un parámetro de columna de tabla utiliza la sintaxis siguiente:

```
@DTWA_NombreApplet(DTW_COLUMN( x )Table)
```

Donde *x* es el nombre o el número de la columna de la tabla.

Los parámetros de columna de tabla utilizan los mismos códigos de applet definidos para los parámetros de tabla.

Texto alternativo para el código de applet en navegadores que no están habilitados para Java: La variable DTW_APPLET_ALTTEXT especifica el texto a visualizar en navegadores que no soportan Java o que han desactivado el soporte de Java. Por ejemplo, la definición de variable siguiente:

```
%define DTW_APPLET_ALTTEXT = "<p>Sorry, your browser is not Java-enabled.</p>"
```

produce el código HTML y el texto siguientes:

```
<p>Sorry, your browser is not Java-enabled.</p><
```

Si no se define esta variable, no se visualiza texto alternativo.

Ejemplo de applet Java

El ejemplo siguiente muestra una macro de Net.Data que llama al entorno de lenguaje de applet Java y el código de applet resultante que genera el entorno de lenguaje.

La macro de Net.Data contiene las llamadas de función siguientes al entorno de lenguaje de applet Java:

```
%define{
DATABASE = "ce1dia1"
DTW_APPLET_ALTTEXT = "<p>Sorry, your browser is not Java-enabled.</p>"
DTW_DEFAULT_REPORT = "no"
MyGraph.codebase = "/netdata-java/"
MyGraph.height = "200"
MyGraph.width = "400"
MyTitle = "This is my Title"
%}
%FUNCTION(DTW_SQL) mySQL(OUT table){
select name, ages from ibmuser.guests
%}
%HTML(report){
@mySQL(MyTable)
@DTWA_MyGraph(MyTitle, DTW_COLUMN(ages) MyTable)
%}
```

Las líneas de macro de Net.Data de la sección DEFINE especifican los atributos del código de applet:

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

El entorno de lenguaje genera un código de applet con los calificadores siguientes:

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

Net.Data devuelve los resultados de la consulta de SQL de la sección SQL de la macro de Net.Data en la tabla de salida, MyTable. Esta tabla se especifica en la sección DEFINE:

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

La llamada a la applet en la macro se especifica en la sección HTML:

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

Basándose en los parámetros de la llamada de función, Net.Data genera el código de applet completo que contiene la información sobre la tabla de

resultados, por ejemplo el número de columnas, el número de filas devueltas y las filas de resultado. Net.Data genera un código de parámetro para cada casilla de la tabla de resultado, como se muestra en el ejemplo siguiente:

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

El nombre de parámetro, *DTW_MyTable_ages_VALUE_1*, especifica la casilla (fila 1, columna ages) de la tabla, MyTable, que tiene un valor de 4. La palabra clave, DTW_COLUMN, en la llamada de función a la applet, especifica que sólo está interesado en la columna "ages" de la tabla resultante, MyTable, mostrada aquí:

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

La salida siguiente muestra el código de applet completo que genera Net.Data para el ejemplo:

```
<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>Sorry, your browser is not Java-enabled.</p>
</applet>
```

Utilización de la interfaz de applet Java de Net.Data

Net.Data proporciona un conjunto de interfaces en una clase llamada DTW_Applet.class, que puede utilizar con las applets Java para ayudar a procesar los códigos PARAM que se generan para las variables de tabla. Puede crear una applet que amplíe esta interfaz para llamar a las rutinas desde la applet.

Net.Data proporciona estas interfaces:

- **int GetNumberOfTables()** devuelve el número de tablas encontradas en el código de applet.
- **String [] GetTableNames()** devuelve una lista de los nombres de tabla encontrados en el código de applet.
- **int GetNumberOfColumns(String table_name)** devuelve el número de columnas de la tabla table_name.
- **int GetNumberOfRows(String table_name)** devuelve el número de filas de la tabla table_name.

- **String[] GetColumnNames(String table_name)** devuelve los nombres de las columnas de la tabla table_name.
- **String[][] GetTable(String table_name)** devuelve una matriz de dos dimensiones de series que contienen los valores de las filas y columnas de la tabla.

Para acceder a las interfaces, utilice la palabra clave EXTENDS en el código de applet para que la applet sea una subclase de la clase DTW_APPLET, como se muestra en el ejemplo siguiente:

```
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 + " has " + ncols +
            " columns and " + nrows + " rows.");

        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");
        }
    }
}

```

Entorno de lenguaje Java Application

Net.Data soporta las aplicaciones Java existentes con el entorno de lenguaje Java. Con el soporte para applets Java y métodos (o aplicaciones) Java, puede acceder a DB2 mediante la API Java Database Connectivity (JDBC**).

En los sitios Web siguientes encontrará detalles acerca de JDBC:

- IBM Software tiene JDK 1.1 o posterior, que es necesario para utilizar JDBC con Net.Data:
<http://www.ibm.com/software/data/db2/java/>
- JavaSoft tiene controladores JDBC adicionales, documentación de API JDBC y las últimas actualizaciones de JDBC:
<http://splash.javasoft.com/jdbc/>

Configuración del entorno de lenguaje Java

Para utilizar el entorno de lenguaje Java, necesita verificar los valores de inicialización de Net.Data y configurar el entorno de lenguaje.

Verifique que la sentencia de configuración siguiente esté en el archivo de inicialización, en una sola línea:

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

Consulte el apartado “Sentencias de configuración de entorno” en la página 28 para obtener más información acerca del archivo de inicialización de Net.Data y las sentencias ENVIRONMENT de entorno de lenguaje.

Importante: Consulte el apartado “Configuración del entorno de lenguaje Java” en la página 32 para obtener información sobre cómo configurar el entorno de lenguaje Java.

Llamada de funciones Java

El entorno de lenguaje Java proporciona una interfaz parecida a una RPC (Remote Procedure Call - Llamada de procedimiento remota). Puede emitir llamadas de función de Java desde la macro de Net.Data con series de Net.Data como parámetros y la función Java invocada puede devolver una serie. Deberá utilizar Live Connection de Net.Data cuando utilice el entorno

de lenguaje Java (consulte el apartado “Gestión de conexiones” en la página 210 para obtener más información sobre Live Connection).

Para llamar a funciones Java:

1. Escriba las funciones Java.
2. Cree una cliette de Net.Data para todas las funciones Java (las cliettes de Net.Data arrancan la Java Virtual Machine donde se ejecuta la función Java).
3. Defina una cliette en la sentencia ENVIRONMENT de Java en el archivo de configuración de Live Connection.
Cada vez que introduzca funciones Java nuevas, deberá volver a crear la cliette Java.
4. Inicie Connection Manager.
5. Ejecute la macro de Net.Data que invoca el entorno de lenguaje Java.

Creación de la función Java: Modifique el archivo de ejemplo de función Java `UserFunctions.java` o cree un archivo nuevo, según el modelo del archivo de ejemplo siguiente, llamado `myfile.java`:

```
=====myfile.java=====
import mypackage.*
public String myfctcall(...parameters from macro...)
{
    return ( mypackage.mymethod(...parameters...));
}

public String lowlevelcall(...parameters...)
{
    string result;
    .....code using many functions of your package...
    return(result)
}
```

Estructura de archivos de entorno de lenguaje Java: Net.Data crea varios directorios durante la instalación de Net.Data. Estos directorios incluyen los archivos que se necesitan para crear las funciones Java, definir la cliette y ejecutar la macro con el entorno de lenguaje Java:

- Una función Java de ejemplo llamada `UserFunctions.java`.
- Un archivo de ejemplo llamado `makeClas`. Cuando se ejecuta, este archivo crea una clase de cliette de Net.Data para la función Java.
- Un archivo de ejemplo llamado `launchjv` utilizado por la cliette Net.Data para arrancar la Java Virtual Machine y ejecutar la función Java.

La Tabla 12 en la página 197 describe los nombres de directorios y archivos para los archivos del sistema operativo correspondiente.

Tabla 12. Archivos utilizados para crear funciones Java

Sistema operativo	Nombre de archivo	Directorio
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

Definición de la cliette de entorno de lenguaje Java: Modifique el archivo de ejemplo, makeClas.bat, o cree un archivo .bat nuevo para generar una clase de cliette de Net.Data, llamada dtw_samp.class, para todas las funciones Java. El ejemplo siguiente muestra cómo el archivo de proceso por lotes, CreateServer, procesa tres funciones Java:

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

El archivo de proceso por lotes procesa los archivos siguientes, junto con el archivo stub proporcionado por Net.Data llamado Stub.java para crear dtw_samp.class.

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

La escritura de una aplicación o applet JDBC es muy similar a la escritura de una aplicación C utilizando la CLI DB2 u ODBC para acceder a una base de datos. La principal diferencia entre aplicaciones y applets es que una aplicación puede necesitar software especial para comunicarse con DB2, por ejemplo, DB2 Client Application Enabler. La applet depende de un navegador Web habilitado para Java y no necesita que se instale ningún código DB2 en el cliente.

Antes de utilizar JDBC, necesitará realizar algunas tareas de configuración en el sistema. En el sitio Web de Soporte de aplicaciones y applets JDBC de DB2 indicado a continuación, encontrará consideraciones sobre este tema:

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

Ejemplo de entorno de lenguaje Java

Después de haber creado la función Java, de haber definido la clase de cliente y de haber configurado Net.Data, puede ejecutar la macro que contiene referencias a la función Java. **Importante:** Inicie Connection Manager antes de invocar la macro de Net.Data.

En el ejemplo siguiente, la llamada de función, *myfctcall*, llama a la función de ejemplo proporcionada con Net.Data, utilizando la cliente DTW_JAVAPPS.

```
%FUNCTION (DTW_JAVAPPS) myfctcall( ....parámetros de macro ....)
```

```
%{ to call the sample provided with Net.Data %}
```

```
%FUNCTION (DTW_JAVAPPS) reverse_line(str);
```

```
%HTML(report){
```

```
you should see the string "Hello World" in reverse.
```

```
@reverse_line("Hello World")
```

```
You should have the result of your function call.
```

```
@myfctcall( ... ....)
```

```
%}
```

Entorno de lenguaje Perl

El entorno de lenguaje Perl puede interpretar scripts Perl incorporados, que se especifican en un bloque FUNCTION de la macro de Net.Data, o puede procesar scripts Perl externos, que se almacenan en archivos independientes en el servidor.

Configuración del entorno de lenguaje Perl

Verifique que la sentencia de configuración siguiente esté en el archivo de inicialización de Net.Data, en una sola línea:

```
ENVIRONMENT (DTW_PERL) DTWPERL ( OUT RETURN_CODE )
```

Consulte el apartado “Sentencias de configuración de entorno” en la página 28 para obtener más información acerca del archivo de inicialización de Net.Data y las sentencias ENVIRONMENT de entorno de lenguaje.

Usuarios japoneses: Perl puede interpretar incorrectamente como caracteres de control algunos caracteres del juego de caracteres japonés SJIS. Existe un paquete fuente abierto llamado jperl que soluciona este problema. Baje e instale el paquete y, a continuación, incluya la sentencia use I18N::Japanese.pm en la cabecera del script Perl.

Llamada de scripts Perl externos

Las llamadas a scripts Perl externos las identifica en un bloque FUNCTION una sentencia EXEC, utilizando la sintaxis siguiente:

```
%EXEC{ nombre_script_perl [parámetros opcionales] %}
```

Necesario: Asegúrese de que *nombre_script_perl*, el nombre del script Perl, se liste en una vía de acceso especificada para la variable de configuración EXEC_PATH en el archivo de inicialización de Net.Data.

```
%FUNCTION(DTW_PERL) perl1() {  
%EXEC{MyPerl.pl %}  
%}
```

Cómo pasar parámetros

Existen dos modos de pasar información a un programa invocado por el entorno de lenguaje Perl (DTW_PERL), directa o indirectamente.

Directamente

Pasar los parámetros directamente en la llamada al script Perl. Por ejemplo:

```
%DEFINE INPARAM1 = "SWITCH1"  
  
%FUNCTION(DTW_PERL) sys1() {  
%EXEC{  
    MyPerl.pl $(INPARAM1) "serie literal"  
%}  
%}
```

Se hace referencia a la variable de Net.Data INPARAM1 y se pasa dicha variable al script Perl. Los parámetros se pasan al script Perl del mismo modo que se pasan dichos parámetros al script Perl cuando se llama a este script desde la línea de mandatos. Los parámetros que se pasan al script Perl utilizando este método se consideran parámetros de tipo de entrada (los parámetros pasados al script Perl pueden ser utilizados y manipulados por dicho script, pero los cambios efectuados en los parámetros no se vuelven a reflejar en Net.Data).

Indirectamente

Pasar parámetros directamente en la llamada al script Perl utilizando uno de los métodos siguientes:

- Hacer que Net.Data pase los parámetros de entrada al script Perl como variables de entorno. Entonces el script Perl puede recuperar los parámetros mediante variables de entorno.
- Hacer que el script Perl vuelva a pasar los parámetros de salida al entorno de lenguaje grabándolos en una conexión con nombre cuyo nombre Net.Data pasa en la variable de entorno, DTWPIPE. Utilice la sintaxis siguiente para grabar datos en la conexión con nombre:

```
nombre="valor"
```

Para múltiples elementos de datos, separe cada elemento con un carácter de línea nueva o un espacio en blanco.

Si el nombre de variable tiene el mismo nombre que un parámetro de salida y utiliza la sintaxis anterior, el nuevo valor sustituye al valor actual. Si un nombre de variable no corresponde a un parámetro de salida, Net.Data lo ignora.

El ejemplo siguiente muestra cómo Net.Data pasa variables desde una macro.

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

Si el script Perl está en un archivo externo llamado today.pl, se puede escribir la misma función como en el ejemplo siguiente:

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

Puede pasar tablas de Net.Data a un script Perl llamado por el entorno de lenguaje Perl. El script Perl accede a los valores de un parámetro de tabla de macro de Net.Data por su nombre Net.Data. Las cabeceras de columna y los valores de campo están contenidos en variables identificadas con el nombre de tabla y el número de columna. Por ejemplo, en la tabla myTable, las cabeceras de columna son myTable_N_j y los valores de campo son myTable_V_i_j, donde *i* es el número de fila y *j* es el número de columna. El número de filas y columnas para la tabla son myTable_ROWS y myTable_COLS.

Bloques REPORT y MESSAGE en bloques FUNCTION

Se permiten bloques REPORT y MESSAGE como en cualquier sección FUNCTION. Dichos bloques los procesa Net.Data, no el entorno de lenguaje. Sin embargo, un script Perl puede grabar en la corriente de salida estándar texto que se deberá incluir como parte de la página Web.

Ejemplo de entorno de lenguaje Perl

El ejemplo siguiente muestra cómo Net.Data genera una tabla ejecutando el script Perl externo:

```
%define {
    c = %TABLE(20)
    rows = "5"
    columns = "5" %}
%function(DTW_PERL) genTable(in rows, in columns, out table) {
    %exec{ perl.pl
}%
```

```

%message{
default: "genTable: Unexpected Error"
%}
%}

%HTML(REPORT) {
@genTable(rows, columns, c)
return code is $(RETURN_CODE)
%}
The Perl script (perl.pl):

open(D2W,"> $ENV{DTWPIPE}");
print "genTable begins ...

";
$r = $ENV{ROWS};
$c = $ENV{COLUMNS};
print D2W "table_ROWS=\"$r\" ";
print D2W "table_COLS=\"$c\" ";
print "rows: $r
";
print "columns: $c";
for ($j=1; $j<=$c; $j++)
{
print D2W "table_N_$j=\"$COL$j\" ";
}
for ($i=1; $i<=$r; $i++)
{
for ($j=1; $j<=$c; $j++)
{
print D2W "table_V_$i","_","$j=\"$i $j\" ";
}
}
}
close(D2W);

```

Resultados: genTable genera:

```

rows: 5 columns: 5
  COL1 | COL2 | COL3 | COL4 | COL5 |
-----
[ 1 1 ] | [ 1 2 ] | [ 1 3 ] | [ 1 4 ] | [ 1 5 ] |
-----
[ 2 1 ] | [ 2 2 ] | [ 2 3 ] | [ 2 4 ] | [ 2 5 ] |
-----
[ 3 1 ] | [ 3 2 ] | [ 3 3 ] | [ 3 4 ] | [ 3 5 ] |
-----
[ 4 1 ] | [ 4 2 ] | [ 4 3 ] | [ 4 4 ] | [ 4 5 ] |
-----
[ 5 1 ] | [ 5 2 ] | [ 5 3 ] | [ 5 4 ] | [ 5 5 ] |
-----
return code is 0

```

Entorno de lenguaje REXX

El entorno de lenguaje REXX le permite ejecutar programas REXX.

Configuración del entorno de lenguaje REXX

Para utilizar el entorno de lenguaje REXX, necesita verificar los valores de inicialización de Net.Data y configurar el entorno de lenguaje.

Verifique que la sentencia de configuración siguiente esté en el archivo de inicialización, en una sola línea:

```
ENVIRONMENT (DTW_REXX)      DTWREXX      ( OUT RETURN_CODE )
```

Consulte el apartado “Sentencias de configuración de entorno” en la página 28 para obtener más información acerca del archivo de inicialización de Net.Data y las sentencias ENVIRONMENT de entorno de lenguaje.

Ejecución de programas REXX

Con el entorno de lenguaje REXX puede ejecutar programas REXX incorporados o programas REXX externos. Un programa REXX incorporado es un programa REXX que tiene la fuente del programa REXX en la macro. Un programa REXX externo tiene la fuente del programa REXX en un archivo externo.

Para ejecutar un programa REXX incorporado:

Defina una función que utilice el entorno de lenguaje REXX (DTW_REXX) y que contenga el código REXX en la sección ejecutable de entorno de lenguaje de la función.

Ejemplo: Función que contiene un programa REXX incorporado

```
%function(DTW_REXX) helloWorld() {  
    SAY 'Hello World'  
%}
```

Para ejecutar un programa REXX externo:

Defina una función que utilice el entorno de lenguaje REXX (DTW_REXX) y que incluya una vía de acceso al programa REXX que se debe ejecutar en una sentencia EXEC.

Ejemplo: Función que contiene una sentencia EXEC que apunta a un programa externo

```
%function(DTW_REXX) externalHelloWorld() {  
%EXEC{ helloworld.exe%}  
%}
```

Necesario: Asegúrese de que el nombre de archivo REXX se lista en una vía de acceso especificada para la variable de configuración EXEC_PATH en el archivo de inicialización de Net.Data. Consulte “EXEC_PATH” en la página 23 para saber cómo se define la variable de configuración EXEC_PATH.

Cómo pasar parámetros a los programas REXX

Existen dos modos de pasar información a un programa REXX invocado por el entorno de lenguaje REXX (DTW_REXX): directa e indirectamente.

Directamente

Pase parámetros directamente a un programa REXX externo utilizando la sentencia %EXEC. Por ejemplo:

```
%FUNCTION(DTW_REXX) rexx1() {  
  %EXEC{  
    CALL1.CMD $(INPARM) "serie literal"  %}  
  %}
```

Se elimina la referencia a la variable de Net.Data INPARM1 y ésta se pasa al programa REXX externo. El programa REXX puede hacer referencia a la variable utilizando la instrucción REXX PARSE ARG. Los parámetros que se pasan al programa utilizando este método se consideran parámetros de tipo entrada (el programa puede utilizar y manipular los parámetros que se le pasan, pero los cambios efectuados en los parámetros no se vuelven a reflejar en Net.Data).

Indirectamente

Pase parámetros indirectamente, por medio de la *agrupación de variables* de programa REXX. Cuando se inicia un programa REXX, el intérprete REXX crea y mantiene un espacio que contiene información acerca de todas las variables. Este espacio se denomina agrupación de variables.

Cuando se llama a una función de entorno de lenguaje REXX (DTW_REXX), el entorno de lenguaje REXX almacena los parámetros de función que son de entrada (IN) o de entrada/salida (INOUT) en la agrupación de variables antes de ejecutar el programa REXX. Cuando se invoca el programa REXX, éste puede acceder a estas variables directamente. Una vez que se ha completado satisfactoriamente el programa REXX, el entorno de lenguaje DTW_REXX determina si existen parámetros de función de salida (OUT) o INOUT. Si existen, el entorno de lenguaje recupera de la agrupación de variables el valor que corresponde al parámetro de función y lo actualiza con el nuevo valor. Cuando Net.Data recibe el control, actualiza todos los parámetros OUT o INOUT con los valores nuevos obtenidos del entorno de lenguaje REXX. Por ejemplo:

```
%DEFINE a = "3"  
%DEFINE b = "0"  
%FUNCTION(DTW_REXX) double_func(IN inp1, OUT outp1){  
  outp1 = 2*inp1  
  %}  
  
%HTML (REPORT) {  
  Value of b is $(b), @double_func(a, b) Value of b is $(b)  
  %}
```

En el ejemplo anterior, la llamada *@double_func* pasa dos parámetros, *a* y *b*. La función REXX *double_func* duplica el primer parámetro y almacena el resultado en el segundo parámetro. Cuando Net.Data invoca la macro, *b* tiene un valor de 6.

Se pueden pasar tablas de Net.Data a un programa REXX. Un programa REXX accede a los valores de un parámetro de tabla de macro de Net.Data como variables de raíz de REXX. En un programa REXX, las cabeceras de columna y los valores de campo están contenidos en variables identificadas con el nombre de tabla y el número de columna. Por ejemplo, en la tabla *myTable*, las cabeceras de columna son *myTable_N.j* y los valores de campo son *myTable_N.i.j*, donde *i* es el número de fila y *j* es el número de columna. El número de filas de la tabla es *myTable_ROWS* y el número de columnas de la tabla es *myTable_COLS*.

Mejora del rendimiento para el sistema operativo AIX:

Si tiene muchas llamadas al entorno de lenguaje REXX en el sistema AIX, considere la posibilidad de establecer la variable de entorno *RXQUEUE_OWNER_PID* en 0. Las macros que efectúan muchas llamadas al entorno de lenguaje REXX pueden generar fácilmente muchos procesos, agotando los recursos del sistema.

Puede establecer la variable de entorno de uno de estos tres modos:

- En la macro, utilizando la función incorporada *DTW_SETENV*:
`@DTW_rSETENV("RXQUEUE_OWNER_PID", "0")`
- En el archivo de entorno de sistema AIX, insertando la sentencia siguiente:
`/etc/environment: RXQUEUE_OWNER_PID = 0`

Este método afecta al comportamiento de REXX para toda la máquina.

- En el archivo de entorno de servidor Web HTTP; por ejemplo, para Domino Go Webserver, inserte la sentencia siguiente:

```
InheritEnv RXQUEUE_OWNER_PID = 0
```

Este método afecta al comportamiento de REXX para el servidor Web.

Ejemplo de entorno de lenguaje REXX

El ejemplo siguiente muestra una macro que llama a una función REXX para generar una tabla de Net.Data que tiene dos columnas y tres filas. A continuación de la llamada a la función REXX, se llama a una función incorporada, *DTW_TB_TABLE()*, para generar una tabla HTML que se devuelve al navegador.

```
%DEFINE myTable = %TABLE  
%DEFINE DTW_DEFAULT_REPORT = "NO"
```



```
%FUNCTION(DTW_REXX) genTable(out out_table) {
    out_table_ROWS = 3
    out_table_COLS = 2

    /* Set Column Headings */
    do j=1 to out_table_COLS
        out_table_N.j = 'COL'j
    end

    /* Set the fields in the row */
    do i = 1 to out_table_ROWS
        do j = 1 to out_table_COLS
            out_table_V.i.j = '[' i j ']'
        end
    end
}%

%HTML (REPORT) {
    @genTable(myTable)
    @DTW_TB_TABLE(myTable)
}%
```

Resultado:

COL1	COL2
[1 1]	[1 2]
[2 1]	[2 2]
[3 1]	[3 2]

Entorno de lenguaje System

El entorno de lenguaje System soporta la ejecución de mandatos y la llamada de programas externos.

Configuración del entorno de lenguaje System

Añada la sentencia de configuración siguiente al archivo de inicialización, en una sola línea:

```
ENVIRONMENT (DTW_SYSTEM) DTWSYS ( OUT RETURN_CODE )
```

Consulte el apartado “Sentencias de configuración de entorno” en la página 28 para obtener más información acerca del archivo de inicialización de Net.Data y las sentencias ENVIRONMENT de entorno de lenguaje.

Emisión de mandatos y llamada de programas

Para emitir un mandato, defina una función que utilice el entorno de lenguaje System (DTW_SYSTEM) que incluya una vía de acceso al mandato que se debe emitir en una sentencia EXEC. Por ejemplo:

```
%FUNCTION(DTW_SYSTEM) sys1() {
    %EXEC { ^ADDLIBLE.CMD %}
}%
```

Puede acortar la vía de acceso a los objetos ejecutables si utiliza la variable de configuración EXEC_PATH para definir las vías de acceso a los directorios que contienen los objetos (por ejemplo, mandatos y programas). Consulte “EXEC_PATH” en la página 23 para saber cómo se define la variable de configuración EXEC_PATH.

Ejemplo 1: Emite un mandato

```
%FUNCTION(DTW_SYSTEM) sys2() {  
    %EXEC { MYPGM %}  
%}
```

Ejemplo 2: Llama a un programa

```
%FUNCTION(DTW_SYSTEM) sys3() {  
    %EXEC {MYPGM.EXE %}  
%}
```

Cómo pasar parámetros a programas

Existen dos modos de pasar información a un programa invocado por el entorno de lenguaje System (DTW_SYSTEM): directa e indirectamente.

Directamente

Pase los parámetros directamente al programa en la llamada. Por ejemplo:

```
%DEFINE INPARAM1 = "SWITCH1"  
  
%FUNCTION(DTW_SYSTEM) sys1() {  
    %EXEC{  
        CALL1.CMD $(INPARAM1) "serie literal"  
    %}  
%}
```

Se hace referencia a la variable de Net.Data INPARAM1 y se pasa dicha variable al programa. Los parámetros se pasan al programa del mismo modo que se pasan cuando se llama al programa desde la línea de mandatos. Los parámetros que se pasan al programa utilizando este método se consideran parámetros de tipo entrada (el programa puede utilizar y manipular los parámetros que se le pasan, pero los cambios efectuados en los parámetros no se vuelven a reflejar en Net.Data).

Indirectamente

El entorno de lenguaje System no puede pasar o recuperar directamente variables de Net.Data, de modo que éstas quedan disponibles para los programas del modo siguiente:

- Net.Data pasa parámetros de entrada al programa como variables de entorno. Entonces el programa puede recuperar los parámetros mediante variables de entorno.
- El programa vuelve a pasar parámetros de salida al entorno de lenguaje grabándolos en una conexión con nombre cuyo nombre

pasa Net.Data en la variable de entorno DTWPIPE. Utilice la sintaxis siguiente para grabar datos en la conexión con nombre:

nombre="valor"

Para múltiples elementos de datos, separe cada elemento con un carácter de línea nueva o un espacio en blanco.

Si el nombre de variable tiene el mismo nombre que un parámetro de salida y utiliza la sintaxis anterior, el nuevo valor sustituirá el valor actual. Si un nombre de variable no corresponde a un parámetro de salida, Net.Data lo ignora.

El ejemplo siguiente muestra cómo Net.Data pasa variables desde una macro.

```
%FUNCTION(DTW_SYSTEM) sys1 (IN P1, OUT P2, P3) {  
  %EXEC {  
    UPDPGM  
  }  
}
```

Puede pasar tablas de Net.Data a un programa llamado por el entorno de lenguaje System. El programa accede a los valores de un parámetro de tabla de macro de Net.Data por su nombre de Net.Data. Las cabeceras de columna y los valores de campo están contenidos en variables identificadas con el nombre de tabla y el número de columna. Por ejemplo, en la tabla myTable, las cabeceras de columna son myTable_N_j y los valores de campo son myTable_V_i_j, donde *i* es el número de fila y *j* es el número de columna. El número de filas y columnas para la tabla son myTable_ROWS y myTable_COLS.

Se recomienda que no pase tablas con muchas filas porque el número de variables de entorno para el proceso es limitado.

Ejemplo de entorno de lenguaje System

El ejemplo siguiente muestra una macro que contiene una definición de función con tres parámetros, P1, P2 y P3. P1 es un parámetro de entrada (IN) y P2 y P3 son parámetros de salida (OUT). La función invoca un programa, UPDPGM, que actualiza el parámetro P2 con el valor de P1 y establece P3 en una serie de caracteres. Antes de procesar la sentencia del bloque %EXEC, el entorno de lenguaje DTW_SYSTEM almacena P1 y el valor correspondiente en el espacio de entorno.

```
%DEFINE {  
  MYPARM2      = "ValueOfParm2"  
  MYPARM3      = "ValueOfParm3"  
}  
%FUNCTION(DTW_SYSTEM) sys1 (IN P1, OUT P2, P3) {  
  %EXEC {
```

```

        UPDPGM
    %}
%}

%HTML(upd1) {
<p>
Passing data to a program. The current value
of MYPARM2 is "${MYPARM2}", and the current value of MYPARM3 is
"${MYPARM3}". Now we invoke the Web macro function.

@sys1("ValueOfParm1", MYPARM2, MYPARM3)

<p>
After the function call, the value of MYPARM2 is "${MYPARM2}",
and the value of MYPARM3 is "${MYPARM3}".
%}

```

Capítulo 7. Mejora del rendimiento

La mejora del rendimiento es una parte importante del ajuste del sistema. Este capítulo describe estrategias para mejorar el rendimiento de Net.Data. Se describen los temas siguientes:

- “Utilización de las API del servidor Web”
- “Utilización de FastCGI” en la página 210
- “Gestión de conexiones” en la página 210
- “Almacenamiento en antememoria de Net.Data” en la página 214
- “Establecimiento del nivel de anotación cronológica de errores” en la página 241
- “Optimización de los entornos de lenguaje” en la página 241

Además, asegúrese de que el servidor Web se ha ajustado correctamente. El rendimiento del servidor Web tiene un efecto directo en el tiempo de respuesta, independientemente de la rapidez con la que Net.Data procese una macro o una petición directa.

Utilización de las API del servidor Web

El rendimiento se puede mejorar invocando Net.Data con una API de servidor Web, por ejemplo GWAPI, en lugar de CGI. Cuando se ejecuta utilizando una API de servidor Web, Net.Data se ejecuta como una hebra dentro del proceso del servidor Web. Dado que el proceso de un servidor Web tiene múltiples hebras, se pueden procesar simultáneamente múltiples peticiones de Net.Data dentro del mismo espacio de direcciones, eliminando la actividad general generada al invocar Net.Data como un proceso CGI.

Consideración: La utilización de una API de servidor Web proporciona una mejora de rendimiento, sin aislamiento de la aplicación. Dado que Net.Data se ejecuta en un entorno de múltiples hebras, los errores introducidos en los entornos de lenguaje escritos por el usuario, las invocaciones incorrectas o incluso las interrupciones de la base de datos pueden producir problemas con el servidor Web y, potencialmente, desactivarlo. Cuando decida si va a utilizar una de las API de servidor Web, determine si la mayor prioridad para la aplicación es el rendimiento o el aislamiento de la misma.

Utilización de FastCGI

FastCGI proporciona una mejora de rendimiento con el aislamiento de aplicación de CGI. Puede utilizar Net.Data con FastCGI en todos los servidores Web que soporten FastCGI. Consulte el apartado “Configuración de Net.Data para FastCGI” en la página 42 para obtener información sobre cómo realizar la configuración para FastCGI.

Puede ajustar FastCGI para ejecutar la cantidad apropiada de procesos a fin de manejar el número de peticiones de entrada con el parámetro de configuración de procesos. Por ejemplo, si necesita manejar 100 peticiones por segundo y cada petición tarda medio segundo en procesarse, puede establecer la directiva NumProcesses en 50 en el archivo de configuración de FastCGI.

FastCGI se soporta en todos los LE; sin embargo, con Oracle y ODBC, se necesita Live Connection.

Para ajustar el número de proceso simultáneos:

1. Abra el archivo de configuración donde se define el parámetro de configuración para procesos.
 - Para Apache, este archivo es `httpd.conf`.
 - Para Lotus Domino Go, este archivo es `lgw_fcgi.conf`.
2. Cambie el valor de parámetro de configuración que especifica el número de procesos:
 - Para Apache: `Process=núm.`
 - Para ISC: `NumProcess=núm.`

Donde *núm* es el número de procesos.

Gestión de conexiones

Net.Data proporciona un componente llamado Live Connection para gestionar conexiones de base de datos y JVM (Java Virtual Machine - Máquina virtual Java). Live Connection mantiene conexiones permanentes para mejorar el rendimiento. Algunas acciones de Net.Data requieren un tiempo de arranque largo. Por ejemplo, antes de que se pueda emitir una consulta de base de datos, el proceso debe identificarse a sí mismo en el DBMS y conectarse a la base de datos. A menudo esto representa una parte significativa del tiempo de proceso necesario para las macros de Net.Data que acceden a una base de datos. Otro ejemplo de arranque de alto coste es el caso de una JVM que es necesaria para ejecutar aplicaciones Java (pero no applets Java). Debido al modo en que operan los programas CGI, estos costes de arranque se pagan en cada petición al servidor Web. Net.Data proporciona Live Connection en los sistemas operativos OS/2, Windows NT y UNIX para mantener conexiones permanentes.

Acerca de Live Connection

Live Connection puede mejorar espectacularmente el rendimiento al eliminar la actividad general del arranque. El ahorro se obtiene gracias a la ejecución continua de uno o más procesos que efectúan las funciones de arranque. Entonces estos procesos esperan para atender las peticiones. Puede ejecutar Live Connection si utiliza Net.Data como un programa CGI o FastCGI o como un plug-in de API de servidor Web.

Live Connection consta de Connection Manager y de cliettes. Las *cliettes* son procesos que Connection Manager inicia y que permanecen activos mientras se ejecuta el servidor. Las cliettes procesan los datos y se comunican con los entornos de lenguaje de Net.Data que se han especificado en el archivo de inicialización con la palabra clave CLIETTE. Cada tipo de cliette maneja una función específica de entorno de lenguaje, por ejemplo la cliette DB2, que se conecta a la base de datos DB2 y configura operaciones para realizar llamadas de SQL antes de que Net.Data procese ninguna macro de Net.Data. El archivo ejecutable se menciona en el archivo de configuración de Live Connection, dtwcm.cnf. La Figura 25 muestra la interacción entre Live Connection, la macro y los entornos de lenguaje.

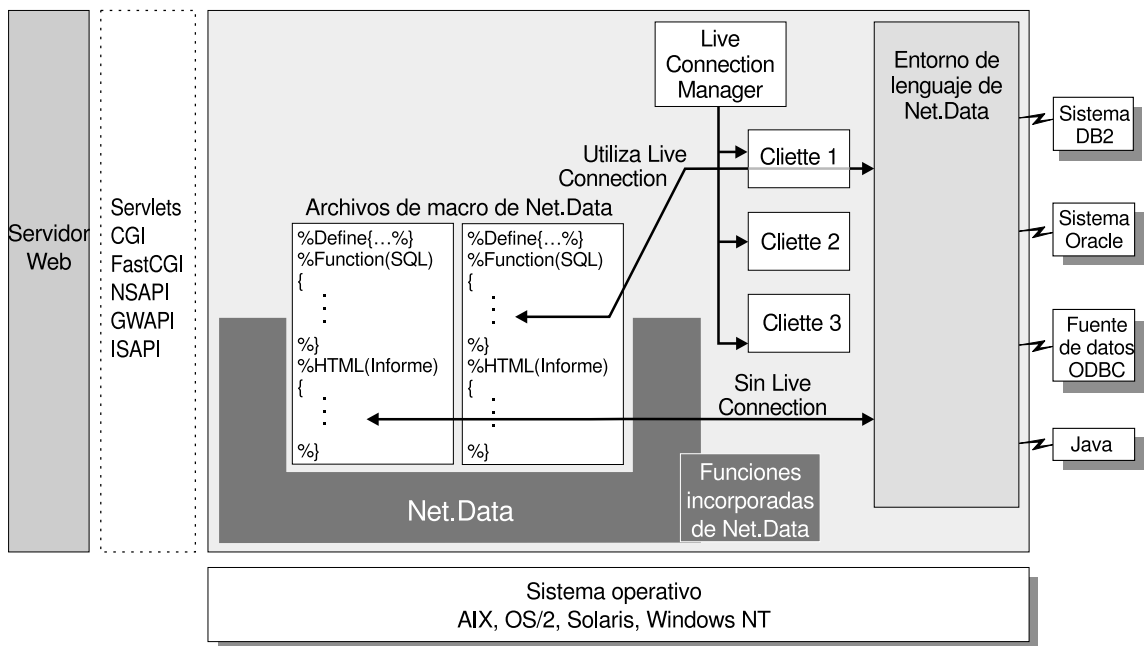


Figura 25. Live Connection con cliettes

Las secciones siguientes describen Live Connection más detalladamente. Para obtener información sobre cómo configurar Live Connection, consulte el apartado “Configuración de Live Connection” en la página 36.

Ventajas de Live Connection

Las principales ventajas de utilizar Live Connection son:

- **Mejora de rendimiento**

Es más eficiente reutilizar conexiones que establecer conexiones nuevas. Generalmente, si solicita sentencias de SQL pequeñas (por ejemplo, consultas simples en una base de datos con menos de 100.000 filas) o si la conexión a la base de datos es difícil (por ejemplo, servidores remotos), el tiempo de conexión es significativo.

- **Acceso a múltiples bases de datos**

Live Connection permite una conexión de macro de Net.Data a múltiples bases de datos al mismo tiempo. Esto es posible porque cada base de datos tiene clientes exclusivos y, por consiguiente, Net.Data simplemente se comunica con múltiples clientes.

¿Debo utilizar Live Connection?

Puede utilizar Live Connection en modalidad CGI, FastCGI o API para comunicarse con la base de datos. Además, puede beneficiarse de Live Connection si la aplicación necesita datos de múltiples bases de datos.

Live Connection, utilizado junto con un plug-in de API, mejora el rendimiento para muchos sistemas, en función de la carga y configuración de los mismos. Deberá experimentar con su propio sistema para determinar la configuración que le funciona mejor.

Muchas aplicaciones pueden mejorar el rendimiento sin utilizar Live Connection mediante el uso del mandato `ACTIVATE DATABASE` para ahorrar el tiempo necesario para establecer las conexiones de base de datos. Consulte la documentación de la base de datos para obtener detalles sobre el mandato que utiliza la base de datos. Consulte también la documentación del sistema operativo para ver si existen pasos adicionales a realizar para ayudar a mejorar el rendimiento.

Requisito: Los entornos de lenguaje ODBC y Oracle necesitan Live Connection cuando se ejecutan en las modalidades FastCGI y API.

Inicio de Connection Manager

Connection Manager (Gestor de conexiones) es un archivo ejecutable independiente que se envía con Net.Data y se denomina `dtwcm`. Inicie Connection Manager al iniciar el servidor Web.

Al iniciarse, Connection Manager lee un archivo de configuración e inicia un grupo de procesos. En cada proceso, Connection Manager empieza la

ejecución de una cliette determinada. Para obtener información sobre cómo configurar Live Connection, consulte el apartado “Configuración de Live Connection” en la página 36.

Para iniciar Connection Manager con Windows NT y OS/2:

1. Desde la línea de mandatos, cambie al directorio `<dir_inst>\connect\`.
2. Entre `dtwcm`.

Donde `<dir_inst>` es el directorio de instalación de Net.Data.

Para iniciar Connection Manager con AIX:

1. Desde la línea de mandatos, cambie al directorio
`/usr/lpp/internet/db2www/db2/`.
2. Entre `dtwcm`.

Para iniciar Connection Manager con la opción de mensajes:

Por omisión, los mensajes de Connection Manager están suprimidos. Utilice la opción `-d` al iniciar Connection Manager si desea que se visualicen los mensajes de Connection Manager.

Desde la línea de mandatos, entre: `dtwcm -d`

Después de utilizar la opción `-d`, tiene que reiniciar Connection Manager para suprimir los mensajes otra vez.

Para iniciar automáticamente Connection Manager como un servicio de Windows NT:

En Windows NT, puede especificar que Connection Manager se inicie como un servicio de Windows NT, en lugar de hacerlo desde la línea de mandatos. La ejecución de Connection Manager como un servicio de Windows NT permite que Connection Manager se inicie automáticamente cada vez que se inicia la máquina.

Consejo: Inicie Connection Manager desde la línea de mandatos antes de configurarlo para que se inicie automáticamente a fin de asegurarse de que el archivo de configuración de Live Connection es correcto.

1. En la barra de tareas de Windows NT, seleccione **Inicio->Configuración->Panel de control->Servicios**.
2. Seleccione **Net.Data Connection Manager** y, a continuación, pulse el botón **Inicio**.
3. Seleccione **Tipo de arranque automático** y, a continuación, pulse **Aceptar**.

Flujo de proceso de Net.Data y Live Connection

Después de haber configurado e iniciado la base de datos, el servidor Web y Connection Manager, el proceso de Net.Data incluye normalmente estos pasos cuando se ha habilitado Live Connection:

1. El servidor Web recibe una petición e inicia un proceso FastCGI, CGI o API para ejecutar Net.Data.
2. Net.Data empieza a procesar la macro de Net.Data.
3. Cuando Net.Data encuentra una llamada de función que utiliza Live Connection, determina qué tipo de cliette se necesita del archivo de inicialización. Para DB2, el tipo de cliette es normalmente un nombre basado en el nombre de base de datos DB2, por ejemplo DTW_SQL:CELDIAL.
4. Net.Data solicita a Connection Manager una cliette de ese tipo.
5. Connection Manager busca las cliettes disponibles de dicho tipo. Si no hay ninguna disponible, Connection Manager pone la petición en una cola y la procesa cuando está disponible el tipo de cliette correcto.
6. Cuando una cliette queda disponible, Connection Manager indica a Net.Data cómo debe comunicarse con la cliette.
7. Net.Data solicita a la cliette que procese la función.
8. Este proceso se repite desde el paso 3 hasta que se completa el proceso de la macro de Net.Data.
9. Se liberan todas las cliettes.

Si una cliette se especifica en el archivo de inicialización pero Connection Manager no se está ejecutando, Net.Data carga la DLL y procesa la macro. Si utiliza una API, es probable que reciba errores y deberá iniciar Connection Manager.

Almacenamiento en antememoria de Net.Data

El almacenamiento en antememoria ayuda a mejorar los tiempos de respuesta para el usuario de la aplicación. Net.Data almacena los resultados de una petición al servidor Web localmente para su rápida recuperación, hasta que llega el momento de renovar la información. Este capítulo describe los conceptos, las tareas y las restricciones del almacenamiento en antememoria de Net.Data.

- “Acerca del almacenamiento de páginas Web en antememoria” en la página 215
- “Acerca del almacenamiento en antememoria de Net.Data” en la página 216
- “Terminología del almacenamiento en antememoria de Net.Data” en la página 216

- “Conceptos de almacenamiento en antememoria de Net.Data” en la página 217
- “Restricciones del almacenamiento en antememoria de Net.Data” en la página 219
- “Interfaces de almacenamiento en antememoria de Net.Data” en la página 219
- “Planificación del Cache Manager” en la página 220
- “Identificadores de antememoria” en la página 221
- “Configuración de las antememorias del Cache Manager y de Net.Data” en la página 222
- “Inicio y detención del Cache Manager” en la página 230
- “Almacenamiento de páginas Web en antememoria” en la página 231
- “Mandato CACHEADM” en la página 235
- “Anotación cronológica de antememoria” en la página 238

Acerca del almacenamiento de páginas Web en antememoria

Muchos componentes de software realizan el almacenamiento en antememoria para las aplicaciones Web. He aquí unos ejemplos de aplicaciones de almacenamiento en antememoria:

- Un navegador Web guarda páginas Web y objetos relacionados, tales como archivos de imagen y audio y applets Java, localmente en memoria o en disco, para ahorrar tiempo de red cuando el usuario accede repetidamente a las mismas páginas.
- Una antememoria de servidor proxy Web guarda páginas Web y objetos relacionados en un servidor local, cerca de un grupo de usuarios, para reducir el tiempo de acceso a la red a los servidores Web remotos, por ejemplo para reducir el número de veces que los servidores Web recuperan elementos solicitados. Una antememoria de servidor proxy Web también permite el compartimiento eficiente, entre múltiples usuarios, de páginas a las que se accede comúnmente.
- Un servidor Web almacena en antememoria páginas que se recuperan frecuentemente y objetos relacionados, con el fin de ahorrar tiempo de acceso a disco cuando los usuarios recuperan repetidamente las mismas páginas.
- Un sistema de gestión de bases de datos almacena en antememoria elementos de datos, que normalmente se mantienen en disco, para ahorrar tiempo de acceso a disco cuando se recuperan repetidamente los mismos elementos de datos.

Todos estos componentes efectúan el almacenamiento en antememoria de forma independiente, pero el resultado general es la mejora de los tiempos de respuesta para los usuarios. Para determinar cuándo se debe renovar un

elemento almacenado en antememoria, los componentes Web (navegador, servidor proxy y servidor Web) tienen generalmente en consideración varias opciones que incluyen:

- Las opciones de configuración del navegador y del servidor
- El contenido de las cabeceras HTTP devueltas con las páginas Web y los elementos relacionados del servidor Web, en particular la información de fecha de caducidad

Acerca del almacenamiento en antememoria de Net.Data

El propio Net.Data proporciona su propia función de almacenamiento en antememoria para páginas a las que se accede con frecuencia y elementos de datos relacionados generados por las macros de Net.Data. Mediante la entrega de una página de la antememoria de Net.Data, ahorrará el tiempo necesario para ejecutar una macro de Net.Data y para acceder a una base de datos con el fin de crear la página.

Puede utilizar un Cache Manager por servidor. **Recomendación:** Utilice un Cache Manager para muchas instancias de Net.Data y múltiples antememorias por Cache Manager.

La Figura 26 muestra que Net.Data utiliza un Cache Manager para gestionar el almacenamiento en antememoria de la salida HTML de una macro. Esta salida puede incluir datos de una base de datos.

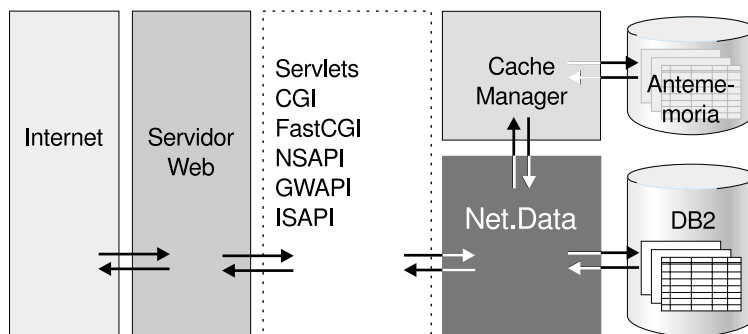


Figura 26. Almacenamiento en antememoria de Net.Data

Terminología del almacenamiento en antememoria de Net.Data

La documentación de Net.Data utiliza los términos siguientes para describir el almacenamiento en antememoria de Net.Data.

antememoria

Tipo de memoria que contiene datos a los que se ha accedido recientemente, diseñada para acelerar el acceso subsiguiente a los mismos datos. La antememoria se utiliza normalmente para mantener una copia local de datos utilizados con frecuencia a los que se puede

acceder a través de una red. En Net.Data, memoria local que contiene páginas Web HTML generadas por Net.Data para que las vuelva a usar la macro de Net.Data. Al tener las páginas almacenadas en la antememoria, Net.Data no tiene que volver a generar la información en la antememoria. Cada antememoria está gestionada por el Cache Manager, que puede ser responsable de múltiples antememorias y puede atender múltiples instancias de Net.Data.

ID de antememoria

Serie que identifica una antememoria determinada.

Cache Manager

Programa que gestiona el almacenamiento en antememoria para una máquina. Puede gestionar múltiples antememorias.

Archivo de configuración de Cache Manager

Archivo que contiene los valores utilizados por Net.Data para determinar los valores para la anotación cronológica, el rastreo, el tamaño de antememoria y otras opciones. Contiene valores para un Cache Manager y todos los archivos de antememoria gestionados por un Cache Manager determinado. El nombre de archivo es `cachemgr.cnf` cuando está empaquetado con Net.Data.

Conceptos de almacenamiento en antememoria de Net.Data

En función del número de servidores HTTP que tenga en el sistema y dependiendo de si cada servidor HTTP ejecuta su propia copia de Net.Data (utilizando archivos de configuración de Net.Data independientes), puede hacer que todas las copias de Net.Data estén asociadas con un Cache Manager o con múltiples Cache Manager. Un Cache Manager puede soportar diversas antememorias en memoria y cada antememoria tiene un identificador de antememoria llamado *ID de antememoria*. La Figura 27 en la página 218 muestra un Cache Manager funcionando con múltiples macros y gestionando dos antememorias.

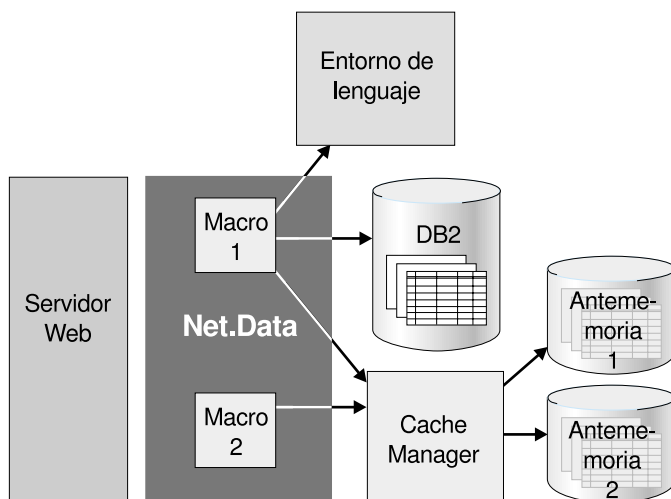


Figura 27. Cache Manager funciona con múltiples macros y antememorias

En una antememoria se puede poner cualquier número de elementos, conocidos como *páginas en antememoria*. Cada página en antememoria tiene un identificador exclusivo, por ejemplo un URL (Uniform Resource Locator - Localizador uniforme de recursos). Una página es un segmento o la totalidad de una página HTML.

Cuando Net.Data recibe una petición de datos en antememoria (por ejemplo, de la función incorporada DTW_CACHE_PAGE), se realizan los pasos siguientes:

1. Net.Data se conecta al Cache Manager.
2. Net.Data comprueba si los datos están en antememoria.
 - Si los datos están en antememoria y no están caducados, Net.Data solicita la página al Cache Manager, la envía al navegador y deja de ejecutar la macro.
 - Si los datos no están en antememoria, Net.Data continúa procesando la macro y, a continuación, envía la página HTML generada al navegador Web y al Cache Manager, donde se almacena en antememoria.
3. Net.Data se desconecta del Cache Manager

El Cache Manager almacena en antememoria la salida HTML cuando la macro realiza satisfactoriamente el proceso, asegurándose de que sólo se almacenan en antememoria las páginas Web generadas satisfactoriamente. Los datos no se almacenan en antememoria hasta después de haberse enviado al navegador, y los datos que el usuario ve son los mismos que los que están en antememoria.

Cuando Net.Data encuentra un error o sale prematuramente de la macro, Cache Manager:

- No acepta páginas parciales o defectuosas
- Conserva las páginas existentes en la antememoria

Restricciones del almacenamiento en antememoria de Net.Data

El almacenamiento en antememoria de Net.Data tiene las restricciones siguientes:

Seguridad

El Cache Manager no proporciona seguridad. Por ejemplo, si un usuario de base de datos ejecuta una macro y almacena en antememoria una página de resultados de la base de datos, otro usuario de base de datos puede recuperar la página en antememoria.

Petición directa

La invocación de petición directa de Net.Data no puede utilizar el almacenamiento en antememoria de Net.Data.

Interfaces de almacenamiento en antememoria de Net.Data

Net.Data proporciona un conjunto flexible de interfaces para que las utilice al configurar y preparar el almacenamiento en antememoria para la aplicación. La Tabla 13 describe las diversas opciones para utilizar las características del almacenamiento en antememoria de Net.Data e indica dónde se describen dichas características.

Tabla 13. Interfaces de antememoria de Net.Data

Interfaz	Descripción	Ir a ...
Opciones de configuración de Cache Manager	Puede especificar diversas opciones para el Cache Manager, por ejemplo la anotación cronológica y el rastreo, en la stanza de Cache Manager del archivo de configuración de Cache Manager.	“Definición del Cache Manager” en la página 222

Tabla 13. Interfaces de antememoria de Net.Data (continuación)

Interfaz	Descripción	Ir a ...
Opciones de configuración de antememoria	Dentro de una instancia individual de Cache Manager de Net.Data, puede definir diversas antememorias que deberán contener los elementos en antememoria. Cada antememoria tiene su propio conjunto de características, tales como el tamaño y la ubicación, así como el ID de antememoria. Estas características se definen en la stanza de antememoria del archivo de configuración de Cache Manager. Cada stanza se identifica por el ID de antememoria.	“Definición de una antememoria” en la página 225
Opciones de inicialización de Net.Data	Si Net.Data y el Cache Manager correspondiente se ejecutan en sistemas independientes, especifique el sistema y el número de puerto de Cache Manager en el archivo de inicialización de Net.Data.	“Variables de configuración de Cache Manager” en la página 15
Funciones incorporadas de antememoria de Net.Data	Puede manipular el contenido de una antememoria de Net.Data utilizando las funciones incorporadas de antememoria de Net.Data. Especifique el ID de antememoria en la función de macro apropiada para seleccionar la antememoria con las características más apropiadas.	Consulte el capítulo de funciones incorporadas del manual <i>Net.Data Guía de consulta</i>

Planificación del Cache Manager

Al planificar la utilización de las funciones de antememoria de Net.Data deberá tener en cuenta lo siguiente:

- Qué páginas se beneficiarán del almacenamiento en antememoria y qué mejoras de rendimiento se obtendrán
- Cuándo se deben almacenar en antememoria los elementos
- Cuándo se deben renovar los elementos de la antememoria y los métodos de renovación a utilizar

Para utilizar el almacenamiento en antememoria de Net.Data, necesita realizar los pasos siguientes, para lo cual es necesario saber cómo se desea utilizar el almacenamiento en antememoria.

Recomendación: Antes de aventurarse en una aplicación importante que utilice el almacenamiento en antememoria, recomendamos encarecidamente que planifique la aplicación y cree un prototipo de la misma antes de ponerla en producción.

- Instale Net.Data, lo cual incluye la función de almacenamiento en antememoria.
- Configure Cache Manager. Consulte el apartado “Configuración de las antememorias del Cache Manager y de Net.Data” en la página 222.
- Determine cómo va a poner la aplicación Net.Data en producción.
- Compruebe las diversas anotaciones cronológicas de antememoria de Net.Data para determinar si se deben realizar mejoras en el uso de la antememoria y en el modo en que ésta está configurada.

Errores de antememoria

El Cache Manager no almacena en antememoria páginas Web cuando Net.Data encuentra errores internos que le hacen salir de la macro antes de que se haya completado el proceso. El Cache Manager no almacena en antememoria páginas que estén incompletas o que contengan errores de Net.Data. Estos tipos de errores incluyen errores de sintaxis de macros y errores de SQL.

Las páginas con errores pueden almacenarse en antememoria cuando:

- Net.Data encuentra un error, continúa ejecutando la macro debido a una variable de configuración CONTINUE del bloque de mensaje y termina normalmente.
- Se producen errores fuera del ámbito de determinación de errores de Net.Data, por ejemplo retrotracciones de la base de datos.

Identificadores de antememoria

Al diseñar el almacenamiento en antememoria para la aplicación, necesitará realizar la planificación para dos tipos de identificadores.

- **Identificador para una antememoria:** Este identificador es el ID de antememoria y especifica el nombre de la stanza de archivo de configuración que define la antememoria. Puede utilizar muchas propuestas

para clasificar y denominar las antememorias. Por ejemplo, puede denominar la antememoria como la aplicación. Puede tener una antememoria para cada una de las aplicaciones de Net.Data, dándole a cada antememoria un nombre que se obtiene de la macro de Net.Data a la que atiende.

- **Identificador para la página en antememoria:** Este identificador es el ID de página en antememoria y especifica el nombre de la página que se debe almacenar en antememoria. El ID de página en antememoria puede ser cualquier serie, por ejemplo una dirección de URL. Especifique el identificador con la función incorporada `DTW_CACHE_PAGE()`. Consulte el capítulo de funciones incorporadas del manual *Net.Data Guía de consulta* para conocer la sintaxis y ver ejemplos.

Configuración de las antememorias del Cache Manager y de Net.Data

El Cache Manager gestiona una o más antememorias en el sistema. Cada una de estas antememorias incluye el contenido de las páginas HTML generadas dinámicamente. Para configurar el Cache Manager y cada una de las antememorias, actualice los valores de palabra clave en el archivo de configuración de Cache Manager, `cachemgr.cnf`.

El archivo de configuración de Cache Manager contiene dos tipos de stanzas: la stanza de Cache Manager y la stanza de Definición de antememoria. Los pasos siguientes describen cómo personalizar estos dos tipos de stanzas para la aplicación.

Definición del Cache Manager

Defina la stanza de Cache Manager especificando valores para las palabras clave permitidas. Todas las palabras clave son opcionales; no necesita especificarlas a menos que no desee aceptar el valor por omisión.

Para definir el Cache Manager:

1. Especifique el nombre del archivo de anotaciones cronológicas de Cache Manager. La anotación cronológica muestra la actividad para todas las transacciones de todas las antememorias y se proporciona para realizar la depuración y el análisis de problemas.

El valor por omisión es escribir los mensajes en la consola.

Sintaxis:

`log=víaacceso`

Donde *víaacceso* es la vía de acceso y el nombre del archivo de antememoria.

Consejo: Para especificar un archivo de anotaciones cronológicas para cada antememoria, utilice la palabra clave **tran-log** en la stanza de Definición de antememoria.

2. Especifique el número de puerto TCP/IP utilizado por el Cache Manager para las peticiones de entrada. Este número de puerto sólo se utiliza para ponerse en contacto con el Cache Manager desde una máquina remota. Este valor debe coincidir con el número de puerto especificado por la variable de configuración DTW_CACHE_PORT en el archivo de inicialización de Net.Data. El valor por omisión se determina del modo siguiente:
 - a. El Cache Manager busca en la vía de acceso */etc/services* el valor asociado con el nombre *ibm-cachemgrd*. Si se encuentra este valor, el Cache Manager lo utiliza. Si no se encuentra, utiliza el método siguiente.
 - b. El Cache Manager utiliza el puerto por omisión 7175.

Sintaxis:

`port=número_puerto`

Donde *número_puerto* es un número de puerto TCP/IP exclusivo.

3. Especifique el tiempo máximo en segundos que el Cache Manager debe permitir que se deje activa una lectura pendiente. Si se excede este tiempo, el Cache Manager desactiva la conexión.

El valor por omisión es de 30 segundos.

Sintaxis:

`connection-timeout=segundos`

Donde *segundos* es el número de segundos utilizados para el tiempo que debe estar activa una lectura pendiente.

4. Especifique si se deben anotar cronológicamente los mensajes. El valor por omisión es **no** u **off**.

Sintaxis:

`logging=yes|on|no|off`

Donde:

yes|on

Indica que la anotación cronológica es necesaria

no|off

Indica que no se debe efectuar anotación cronológica.

5. Especifique si se debe reiniciar la anotación cronológica. El valor por omisión es **no**. Si se especifica **yes**, la anotación cronológica actual se cierra cuando alcanza su tamaño máximo (consulte **log-size**, más abajo), el archivo tiene un tipo de archivo de *.old* y se abre una anotación cronológica nueva. Sólo se mantiene una generación del archivo de anotaciones cronológicas (se sobregaban los archivos *.old* existentes).

Sintaxis:

`wrap-log=yes|no`

Donde:

yes Especifica que se debe reiniciar la anotación cronológica.

no Especifica que no se debe reiniciar la anotación cronológica.

6. Especifique el tamaño máximo, en bytes, hasta el que se permite que crezca a una anotación cronológica, si se especifica que se reinicie la anotación.

El valor por omisión es de 64000.

Sintaxis:

`log-size=bytes`

Donde *bytes* es el número de bytes del tamaño máximo.

7. Especifique el nivel de los mensajes que se deben grabar en la anotación cronológica. Estos valores se establecen cuando se incluyen en la lista *definiciones_distintivos_rastreo* y no tienen ningún valor.

El valor por omisión es sólo anotar cronológicamente los mensajes de arranque y cierre de Cache Manager.

Sintaxis:

`trace-flags=definiciones_distintivos_rastreo`

Donde:

D_ALL

Habilita todos los distintivos de rastreo.

D_NONE

Inhabilita todos los distintivos de rastreo.

Ejemplo: Distintivo de rastreo que especifica que todos los distintivos de rastreo están habilitados:

`trace=flags=D_ALL`

Ejemplo de stanza: Stanza de Configuration Manager válida:

```
cache-manager {
port = 7175
connection-timeout = 60
logging = off
log = /local/netdata/cachemgr/logs/cachemgr.log
wrap-log = yes
log-size = 32KB
}
```

Definición de una antememoria

Defina la stanza de Definición de antememoria especificando valores para las palabras clave permitidas. La mayoría de las palabras clave son opcionales y no tienen que especificarse a menos que no se desee utilizar el valor por omisión.

Para definir una antememoria:

1. Especifique la vía de acceso y el nombre de directorio que deben contener las páginas de antememoria. En el arranque, el sistema de archivos que contiene este directorio debe ser como mínimo del mismo tamaño que el valor de **fssize** (vea más abajo); de lo contrario, no se inicia la antememoria. Este valor puede especificarse como un nombre de vía de acceso absoluta o como un nombre de vía de acceso relativa que corresponda a la vía de acceso en la que se ha iniciado el Cache Manager.

Necesario.

Sintaxis:

`root=nombre_víaacceso`

Donde:

`nombre_víaacceso`

Es el nombre absoluto o relativo de la vía de acceso y directorio donde están almacenadas las páginas de antememoria.

2. Especifique si la antememoria actual está activa cuando se inicia el Cache Manager.

No es necesario; el valor por omisión es **yes**. Si se establece en **no**, la antememoria se define en el Cache Manager pero no se activa. Puede activarla posteriormente con el mandato **cacheadm**.

Sintaxis:

`caching=yes|no`

Donde:

yes Indica que la antememoria debe estar activa cuando se inicie el Cache Manager.

no Indica que la antememoria no debe estar activa cuando se inicie el Cache Manager.

3. Especifique el espacio máximo que deberá ser utilizado en el sistema de archivo por las páginas de la antememoria actual. Cuando se excede la cantidad máxima de espacio, el Cache Manager suprime suficientes páginas, empezando por la más antigua, para dejar dentro del límite el espacio total ocupado por la antememoria. Puede inhabilitar de forma efectiva la depuración automática de entradas estableciendo este valor en

un número grande; sin embargo, si se excede el espacio físico del sistema de archivos, fallarán los intentos de adición de nuevas páginas a la antememoria.

No es necesario; el valor por omisión es 0 (no almacenar en antememoria en disco).

Sintaxis:

`fssize=nnB|nnKB|nnM`

Donde:

nnB Es el número de bytes; por ejemplo 5000B.

nnKB Es el número de kilobytes; por ejemplo 640KB.

nnMB Es el número de megabytes; por ejemplo 30MB.

4. Especifique la cantidad máxima de memoria que deben utilizar todas las páginas de esta antememoria. Cuando se excede la cantidad máxima de memoria, el Cache Manager suprime suficientes páginas, empezando por la más antigua, para dejar dentro de los límites la memoria total ocupada por la antememoria. Puede inhabilitar de forma efectiva la depuración automática de páginas estableciendo este valor en un número grande; sin embargo, si el proceso **cachemgrd** consume demasiada memoria, puede que el sistema operativo lo termine.

No es necesario; el valor por omisión es de 1 MB.

Sintaxis:

`mem-size=nnB|nnKB|nnMB`

Donde:

nnB Es el número de bytes; por ejemplo 5000B.

nnKB Es el número de kilobytes; por ejemplo 640KB.

nnMB Es el número de megabytes; por ejemplo 30MB.

5. Especifique el tiempo máximo durante el cual se puede mantener una página en la antememoria. Cuando se excede este valor, el Cache Manager marca la página como caducada pero no la suprime a no ser que se alcancen los límites de **fssize** (si está almacenada en antememoria en disco) o **memsize** (si está almacenada en la memoria). El Cache Manager suprime las páginas que están marcadas como caducadas antes que todas las demás páginas si se alcanzan los límites de **memsize** o **fssize**. Puede inhabilitar la comprobación de **lifetime** con la palabra clave **check_expiration**.

Necesario: No; el valor por omisión es de 5 minutos.

Sintaxis:

`lifetime=tiempo`

Donde:

nnS Es el número de segundos; por ejemplo, 600S.

nnM Es el número de minutos; por ejemplo, 20M.

nnH Es el número de horas; por ejemplo, 30H.

6. Especifique si se deben marcar como caducadas las páginas de antememoria y efectuar una comprobación de tiempo de vida.

No es necesario; el valor por omisión es **yes**, con un tiempo de vida por omisión de 60 segundos. Este valor también se puede establecer en un tiempo, indicando un valor de **yes** y declarando el tiempo máximo que un elemento puede mantenerse en antememoria. Cuando se establece en **no**, las páginas de antememoria no se marcan nunca como caducadas y no se realiza la comprobación de tiempo de vida.

Sintaxis:

check-expiration=yes|nnS|nnM|nnH|no

Donde:

yes Indica que el Cache Manager realiza la comprobación de tiempo de vida y las páginas de antememoria se marcan como caducadas.

nnS Es el número de segundos; por ejemplo, 600S.

nnM Es el número de minutos; por ejemplo, 20M.

nnH Es el número de horas; por ejemplo, 30H.

no Indica que el Cache Manager no realiza la comprobación de tiempo de vida y las páginas de antememoria no se marcan como caducadas.

7. Especifique la cantidad máxima de espacio que puede ocupar una página en antememoria dentro de la antememoria. Si una página es demasiado grande para la memoria, se comprueba la antememoria de archivo. Si existe el espacio adecuado, en lugar de ello el Cache Manager almacena en la antememoria de archivo la página de antememoria. Si la página no cabe en la antememoria de archivo, el intento de almacenamiento en antememoria falla. Si la página tiene un tamaño menor que el valor de `datum_memory_limit` (`cacheobj-memory-limit`), pero si la antememoria no tiene suficiente espacio, se suprimen de la antememoria las páginas de antememoria más antiguas para dar cabida a la página nueva.

No es necesario; el valor por omisión es de 1 KB.

Sintaxis:

datum-memory-limit (cacheobj-memory-limit)=nnB|nnKB|nnMB

Donde:

nnB Es el número de bytes; por ejemplo 5000B.

nnKB Es el número de kilobytes; por ejemplo 640KB.

nnMB Es el número de megabytes; por ejemplo 30MB.

8. Especifique la cantidad máxima de espacio que puede ocupar una página de antememoria dentro de la antememoria de archivo. Si una página tiene un tamaño menor que `datum_disk_limit`, pero no queda espacio en la antememoria de archivo, se suprimen de la antememoria de archivo las páginas de antememoria más viejas para dar cabida a la página nueva.

No es necesario; el valor por omisión es de 1 KB.

Sintaxis:

`datum-disk-limit (cacheobj-space-limit)=nnB|nnKB|nnMB`

Donde:

nnB Es el número de bytes; por ejemplo 5000B.

nnKB Es el número de kilobytes; por ejemplo 640KB.

nnMB Es el número de megabytes; por ejemplo 30MB.

9. Especifique el tiempo que debe transcurrir entre la creación de registros de estadísticas. Si se establece en 0, no se graban registros de estadísticas.

No es necesario; el valor por omisión es 0 (ninguna estadística).

Sintaxis:

`stat-interval = nnS|nnM|nnH`

Donde:

nnS Es el número de segundos; por ejemplo, 600S.

nnM Es el número de minutos; por ejemplo, 1M.

nnH Es el número de horas; por ejemplo, 3H.

10. Especifique el nombre de la vía de acceso y del archivo que se deben utilizar para anotar cronológicamente las estadísticas para la antememoria actual.

Es necesario cuando el valor para **stat-interval** es mayor que 0.

Sintaxis:

`stat-files=nombrearchivo`

Donde *nombrearchivo* es la vía de acceso y el nombre del archivo de estadísticas de anotación cronológica.

11. Especifique si se deben restablecer a 0 los contadores de estadísticas cada vez que se graban en el archivo de anotaciones cronológicas.

No es necesario; el valor por omisión es **yes**.

Sintaxis:

reset-stat-counters=yes|no

Donde:

yes Restablece los contadores de estadísticas.

no No restablece los contadores de estadísticas.

12. Defina la vía de acceso y el nombre de archivo que debe contener la anotación cronológica de transacciones para cada antememoria. Los archivos de anotaciones cronológicas de transacciones de antememoria son independientes de los archivos de anotaciones cronológicas de Cache Manager, que se utilizan para anotar cronológicamente la actividad general de Cache Manager.

Es necesario; si no se especifica, no se crea ninguna anotación cronológica de transacciones para la antememoria.

Sintaxis:

tran-log=*nombrearchivo*

Donde *nombrearchivo* es la vía de acceso y el nombre de archivo de las anotaciones cronológicas de transacciones para cada antememoria.

13. Especifica si se debe activar la anotación cronológica de transacciones para la antememoria cuando el Cache Manager arranca por primera vez. Este parámetro se ignora a no ser que se especifique un archivo de anotaciones cronológicas de transacciones válido a través del parámetro tran-log. Puede activar la anotación cronológica de transacciones mientras se está ejecutando el daemon de Cache Manager utilizando el mandato **cacheadm** si se ha especificado un valor de tran-log válido en el archivo de configuración de Cache Manager.

No es necesario; el valor por omisión es **no**.

Sintaxis:

tran-logging=yes|on|no|off

Donde:

yes|on

Indica que la anotación cronológica es necesaria.

no|off

Indica que no se debe efectuar anotación cronológica.

14. Especifique si se debe reiniciar la anotación cronológica de transacciones. No es necesario; el valor por omisión es **yes**. Si se especifica como **yes**, la anotación cronológica actual se cierra cuando alcanza su tamaño máximo (vea **tran-log-size**), tiene el tipo de archivo de .old y se abre una

anotación cronológica nueva. Sólo se mantiene una generación de la anotación cronológica (se sobregraban los archivos .old existentes).

Sintaxis:

`wrap-tran-log=yes|no`

Donde:

yes Indica reiniciar la anotación cronológica.

no Indica no reiniciar la anotación cronológica.

15. Especifique el tamaño máximo en bytes hasta el que se permite que crezca a una anotación cronológica de transacciones, si se especifica **wrap-tran-log**.

No es necesario; el valor por omisión es 64000.

Sintaxis:

`tran-log-size=bytes`

Donde *bytes* es el número de bytes del tamaño máximo.

Ejemplo de stanza: Stanza válida de Definición de antememoria para una antememoria:

```
cache0
{
root = /locale/netdata/cachemgr/caches/cache0
caching = on
mem-size = 10MB
fs-size = 1MB
datum-memory-limit = 200KB
datum-disk-limit = 1MB
lifetime = 6000000
check-expiration = 999999
tran-logging = no
tran-log-size = 10000
wrap-tran-log = yes
tran-log = /ocale/netdata/cachemgr/logs/tran.log
}
```

Inicio y detención del Cache Manager

Las secciones siguientes describen cómo iniciar y detener el Cache Manager.

- “Inicio del Cache Manager”
- “Detención del Cache Manager” en la página 231

Inicio del Cache Manager

Utilice el mandato `cachemgrd` para iniciar el daemon de Cache Manager.

Sintaxis:

►►—cachemgrd—-c—*archivo_config*——————►►

Parámetros:

cachemgrd

Palabra clave de mandato.

archivo_config

Especifica el nombre del archivo donde se definen el Cache Manager y cada una de las antememorias gestionadas por el mismo. El archivo de configuración enviado con el producto Net.Data es cachemgr.cnf.

Ejemplo:

```
cachemgrd -c myconfig.cfg
```

Detención del Cache Manager

Utilice el mandato `cacheadm` para detener el Cache Manager.

Sintaxis:

►►—cacheadm—┬──────────────────┬──────────────────┬──────────────────┬──────────────────►►
 hostname—*nombre_sistpral*—┬──────────────────┬──────────────────┬──────────────────┬──────────────────
 port—*núm_puerto*—┬──────────────────┬──────────────────┬──────────────────┬──────────────────

Parámetros:

cacheadm

Palabra clave de mandato.

nombre_sistpral

Especifica el nombre de la máquina donde se ejecuta la antememoria, si es diferente de la máquina donde se emite el mandato `cacheadm`.

núm_puerto

Especifica el número de puerto de antememoria, si el número es diferente del valor por omisión (7175).

terminate

Especifica detener el Cache Manager.

Ejemplo:

```
cacheadm hostname host1 port 7178 terminate
```

Almacenamiento de páginas Web en antememoria

Puede utilizar la función incorporada `DTW_CACHE_PAGE` para almacenar una página Web en antememoria. Cuando Net.Data ve la función

DTW_CACHE_PAGE en la macro, se pone en contacto con el Cache Manager y empieza a guardar la salida HTML para la macro en la memoria. Después de que Net.Data haya procesado satisfactoriamente una macro, se envía la salida HTML al navegador y el Cache Manager almacena en antememoria la salida en una transacción, como se muestra en la Figura 28.

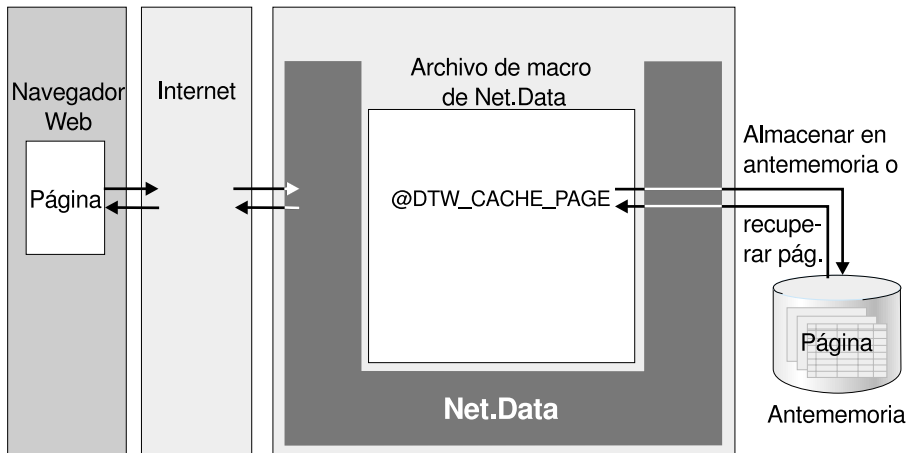


Figura 28. La función DTW_CACHE_PAGE inicia el almacenamiento en antememoria

Almacenamiento de una página en antememoria

Especifique que las páginas generadas por Net.Data se graben en la antememoria utilizando la función incorporada de Net.Data DTW_CACHE_PAGE().

La función DTW_CACHE_PAGE() almacena en antememoria toda la salida de la macro a continuación de la sentencia de función, una vez que ha determinado que la página todavía no existe en la antememoria o ha caducado. Si la página no existe en la antememoria o excede la antigüedad especificada, Net.Data vuelve a enviar la página de salida al navegador, genera páginas de salida nuevas a partir de la ejecución de la macro y almacena la página en la antememoria.

Si el Cache Manager encuentra la página en antememoria y ésta es aún actual, visualiza el contenido de la antememoria y Net.Data sale de la macro. Este comportamiento asegura que no se realice ningún proceso innecesario después de que se haya recuperado la página Web de la antememoria.

Consejo respecto al rendimiento: Para minimizar el coste de ejecución de la macro, escriba DTW_CACHE_PAGE() como la primera o una de las primeras sentencias de una macro.

Para almacenar una página en antememoria:

1. En el bloque HTML de una macro, antes de la codificación HTML, inserte la sentencia de función siguiente:

```
@DTW_CACHE_PAGE("id_antememoria", id_página_en_antememoria, "antigüedad", status)
```

Utilice esta función para especificar que Net.Data debe almacenar en antememoria toda la salida HTML de la macro que sigue a esta sentencia. Ponga esta sentencia al principio de la macro si desea almacenar en antememoria toda la salida HTML.

Parámetros:

id_antememoria

Serie que identifica la antememoria en la que se colocará la página. Puede asociar los ID de antememoria con macros o grupos de macros.

id_página_en_antememoria

Serie que contiene un identificador utilizado para identificar la página en la antememoria en una petición de antememoria @DTW_CACHE_PAGE subsiguiente, por ejemplo el URL de la página.

antigüedad

Variable de serie que contiene una duración en segundos que especifica cuándo se considera que una página está anticuada. Si la página solicitada ha estado en la antememoria más tiempo que el indicado por el valor de *antigüedad*, Net.Data ejecuta la macro, vuelve a generar la página y almacena en antememoria la página generada, sustituyendo a la página anticuada. Si la página solicitada ha estado en la antememoria un tiempo inferior o igual al indicado por el valor de *antigüedad*, Net.Data recupera la página de la antememoria y la envía al navegador. En este caso, Net.Data finaliza la ejecución de la macro inmediatamente.

status Variable de serie devuelta por Net.Data para indicar si la página se ha almacenado en antememoria satisfactoriamente.

Ejemplo:

```
%HTML(cache_example) {  
  %IF (customer == "Joe Smith")  
    @DTW_CACHE_PAGE("mymacro.d2w", "http://www.mypage.org", "-1", status)  
  %ENDIF  
  ...  
  <html>  
  
  <head>  
    <:title>This is the page title</title>  
  </head>
```

```

<body>
  <center>
    <h3>This is the Main Heading</h3>
    <p>It is $(time). Have a nice day!
  </body>

</html>
%}

```

Almacenamiento en antememoria avanzado: cómo determinar dinámicamente si se debe almacenar en antememoria

La función DTW_CACHE_PAGE() inicia el almacenamiento en antememoria desde su ubicación en la macro. Normalmente, la función se coloca al principio de la macro para obtener un mejor rendimiento y asegurar que se almacena en antememoria toda la salida HTML.

Para aplicaciones de almacenamiento en antememoria avanzado, puede poner la función DTW_CACHE_PAGE() en las secciones de salida HTML cuando necesite tomar la decisión de almacenar en antememoria en un punto específico durante el proceso, en lugar de hacerlo al principio de la macro. Por ejemplo, puede que necesite tomar la decisión de almacenar en antememoria basándose en el número de filas devueltas de una consulta o una llamada de función.

Ejemplo: Pone la función en el bloque HTML porque la decisión de almacenar en antememoria depende del tamaño esperado de la salida HTML

```

% DEFINE { ...%}

...

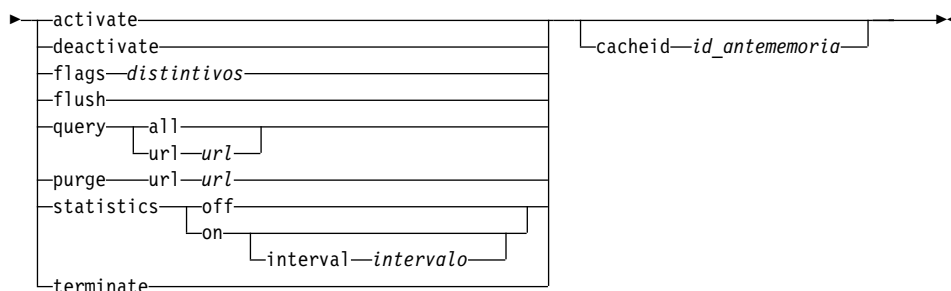
%FUNCTION(DTW_SQL) count_rows(){
  select count(*) from customer
%REPORT{
  %ROW{
    @DTW_ASSIGN(ALL_ROWS, V1)
  %}
  %}
  %}

%FUNCTION(DTW_SQL) all_customers(){
  select * from customer
%}

%HTML(OUTPUT) {
<html>
<head>
  <title>This is the customer list
</head>
<body>

@count_rows()

```

Parámetros:

activate

Activa una antememoria especificada. Si la antememoria ya está activada, el Cache Manager no hace nada.

id_antememoria

Variable de serie que identifica la antememoria en la que está ubicada la página. Por ejemplo: cache1.

deactivate

Desactiva una antememoria especificada. Si la antememoria ya está inactiva, el Cache Manager no hace nada. Todas las operaciones pendientes se completan y no se aceptan otras nuevas. Cuando se ha completado la última operación, el Cache Manager marca la antememoria como inactiva.

flags

Especifica si los distintivos listados deben activarse o desactivarse.

D_ALL

Activa todos los distintivos de rastreo.

D_NONE

Desactiva todos los distintivos de rastreo.

flush

Desecha una antememoria, especificada por el parámetro *id_antememoria*, que es necesario para este parámetro. Este parámetro suprime incondicionalmente todos los elementos de la antememoria especificada.

nombre_sistpral

Especifica el nombre de la máquina donde se ejecuta la antememoria, si es diferente de la máquina donde se emite el mandato cacheadm. Por ejemplo: myhost.

núm_puerto

Especifica el número de puerto de antememoria, si el número es diferente del valor por omisión (7175). Este número debe ser exclusivo en el sistema.

purge

Especifica que debe eliminarse de la antememoria una página específica. Si se especifica *url*, el Cache Manager elimina la página con una clave que coincida con *url*. Si se especifica dependencia, el Cache Manager elimina todos los elementos con la dependencia asociada y graba sus claves en *stdout*, la corriente de salida estándar.

query

Devuelve datos de almacenamiento en antememoria, en función de los parámetros que se especifiquen:

- Devuelve información acerca de una antememoria, si sólo se especifica el ID de antememoria.
- Devuelve información acerca de una página específica en antememoria, si se especifica *url*.
- Devuelve información acerca de todas las páginas, si se especifica *all*.

Otros programas utilizan la opción *all* para formatear o interpretar los resultados. Cada línea contiene la información siguiente:

- Clave de la página
- Antigüedad de la página
- Longitud de la página
- Fecha de creación de la página
- Fecha de caducidad de la página
- Fecha en la que se ha hecho referencia a la página por última vez

Todas las fechas están en formato de hora entera estándar de UNIX.

Consejo respecto al rendimiento: La opción *cache query all* puede afectar al rendimiento y deberá utilizarse de forma limitada.

statistics

Habilita o inhabilita la anotación cronológica de la recopilación de estadísticas para una antememoria específica y necesita el parámetro *id_antememoria*. Si se especifica un intervalo con el parámetro *statistics* establecido en *on*, *Net.Data* establece o restablece el intervalo entre las actualizaciones en el número especificado de segundos.

terminate

Especifica detener el Cache Manager.

tranlogging

Habilita o inhabilita la anotación cronológica de transacciones para una

antememoria específica y necesita el parámetro *id_antememoria*. Este parámetro sólo entra en vigor si se especifica una anotación cronológica de transacciones válida para la antememoria en el archivo de configuración de Cache Manager con el parámetro *tran-log*.

url La dirección de URL (Universal Relative Location - Ubicación universal relativa) que especifica la ubicación del archivo en un servidor Web. Por ejemplo: <http://www.ibm.com/mydir/page1>.

Anotación cronológica de antememoria

Se conservan y, opcionalmente, se graban en la anotación cronológica de antememoria varios tipos de estadísticas relacionadas con el funcionamiento interno. Se puede mantener una anotación cronológica independiente para cada antememoria o bien se pueden grabar todas las estadísticas en la misma anotación cronológica. Esta sección describe los temas de anotación cronológica de antememoria siguientes:

- “Configuración de la anotación cronológica”
- “Formato de la anotación cronológica de antememoria”

Configuración de la anotación cronológica

Para anotar cronológicamente las estadísticas, debe configurar el archivo de configuración de Cache Manager.

Para configurar la anotación cronológica:

Especifique las palabras clave *stat-files* y *stat-interval* en la stanza de antememoria del archivo de configuración de Cache Manager.

Puede modificar los valores de las estadísticas sin detener, reconfigurar ni reiniciar el Cache Manager.

Para modificar los valores de recopilación de estadísticas:

Especifique el mandato *cacheadm statistics*. Sin embargo, tenga en cuenta que los cambios efectuados con el mandato *cacheadm statistics* no se guardan cuando se reinicia el Cache Manager.

Formato de la anotación cronológica de antememoria

La anotación cronológica de estadísticas es un archivo ASCII sencillo que se puede procesar o importar mediante hojas de cálculo o programas de bases de datos. Se graban tres tipos de registros:

- Los registros de inicialización documentan el inicio de la recopilación de estadísticas para una antememoria determinada. Estos registros tienen el formato siguiente:

mm/dd/aa hh:mm:ss id Initialization: interval n seconds

Donde:

mm/dd/aa Es la fecha (expresada en mes, día y año) en la que se inicia la recopilación de estadísticas

hh:mm:ss Es la hora (expresada en horas, minutos y segundos) en la que se inicia la recopilación de estadísticas

id Es el nombre de la antememoria asociada con el registro

n Es el intervalo de recogida

- Los registros de terminación documentan la terminación de la recopilación de estadísticas para una antememoria determinada. Estos registros tienen el formato siguiente:

mm/dd/aa hh:mm:ss id Termination

Donde:

mm/dd/aa Es la fecha (expresada en mes, día y año) en la que se detiene la recopilación de estadísticas

hh:mm:ss Es la hora (expresada en horas, minutos y segundos) en la que se detiene la recopilación de estadísticas

id Es el nombre de la antememoria asociada con el registro

- Los registros de estadísticas son un conjunto de números, delimitado por espacios en blanco, que muestran la actividad en la antememoria. Estos registros tienen el formato siguiente:

mm/dd/aa hh:mm:ss id estadísticas

Donde:

mm/dd/aa Es la fecha (expresada en mes, día y año) en la que se crea la recopilación de estadísticas

hh:mm:ss Es la hora (expresada en horas, minutos y segundos) en la que se crea la recopilación de estadísticas

id Es el nombre de la antememoria asociada con el registro

<estadísticas> Es una lista, delimitada por espacios en blanco, de las estadísticas reunidas para esta antememoria como se especifica en la Tabla 14 en la página 240:

Tabla 14. Lista de estadísticas

Número de campo	Contenido	Descripción	Contadores reinicializados a cero
1	lecturas	Número de operaciones de lectura realizadas en la antememoria	Sí
2	grabaciones	Número de operaciones de grabación realizadas en la antememoria	Sí
3	cierres	Número de operaciones de cierre realizadas en objetos de la antememoria	Sí
4	lectura abierta	Número de operaciones de lectura abierta realizadas en objetos de la antememoria	Sí
5	grabación abierta	Número de operaciones de grabación abierta realizadas en objetos de la antememoria	Sí
6	consulta de grabación abierta	Número de operaciones de consulta de grabación abierta realizadas en objetos de la antememoria	Sí
7	aciertos de lectura	Número de aciertos de lectura en objetos de la antememoria	Sí
8	aciertos de grabación	Número de aciertos de grabación en objetos de la antememoria	Sí
9	aciertos de consulta de grabación	Número de aciertos de consulta de grabación en objetos de la antememoria	Sí
10	inicializaciones	Número de sesiones nuevas establecidas con esta antememoria	Sí
11	terminaciones	Número de sesiones terminadas con esta antememoria	Sí
12	eliminaciones	Número de objetos suprimidos de esta antememoria	No
13	memoria utilizada	Cantidad de memoria utilizada por los objetos de la parte de memoria de la antememoria	No
14	disco utilizado	Cantidad de espacio en disco utilizado por los objetos de la parte de disco de la antememoria	No
15	memoria disponible	Cantidad de memoria aún disponible para que la utilicen los objetos de la parte de memoria de la antememoria	No

Tabla 14. Lista de estadísticas (continuación)

Número de campo	Contenido	Descripción	Contadores reinicializados a cero
16	disco disponible	Cantidad de espacio en disco aún disponible para que lo utilicen los objetos de la parte de disco de la antememoria	No
17	cuenta de objetos de memoria	Número de objetos en la parte de memoria de la antememoria	No
18	cuenta de objetos de archivo	Número de objetos de la parte de disco de la antememoria	No
19	cuenta de sesiones	Número de sesiones actualmente activas en la antememoria	No

Establecimiento del nivel de anotación cronológica de errores

Net.Data proporciona una anotación cronológica de errores para que pueda supervisar errores o problemas de rendimiento en el sistema Net.Data.

Cuando utilice la anotación cronológica de errores de Net.Data, puede que note un efecto en el rendimiento del sistema si se graban muchos mensajes en las anotaciones cronológicas de errores. Por ejemplo, cada vez que un usuario accede a una macro que Net.Data no puede encontrar, Net.Data pasa un mensaje como salida a la anotación cronológica de errores.

Para reducir el efecto en el rendimiento, compruebe el nivel de anotación cronológica de errores establecido en una macro de Net.Data con la palabra clave DTW_LOG_LEVEL. Si el nivel está establecido en WARNING, considere la posibilidad de reducir el nivel a ERROR para obtener un pequeño aumento de rendimiento o a OFF para obtener un mayor aumento de rendimiento.

Optimización de los entornos de lenguaje

Las secciones siguientes describen técnicas que puede utilizar para mejorar el rendimiento al utilizar los entornos de lenguaje proporcionados por Net.Data.

- “Entorno de lenguaje REXX”
- “Entorno de lenguaje SQL” en la página 242
- “Entornos de lenguaje System y Perl” en la página 244

Entorno de lenguaje REXX

Utilice los consejos siguientes para mejorar el rendimiento de la aplicación Net.Data:

- Combine los programas REXX donde sea posible. Un número menor de programas más grandes proporciona un rendimiento mejor que un número mayor de programas más pequeños porque el intérprete de REXX se inicializa cada vez que se llama a una función de entorno de lenguaje REXX en la macro.
- Almacene el programa REXX en un archivo externo en lugar de incluirlo incorporado en la macro de Net.Data.
- Para los programas REXX externos, haga referencia a las variables globales en la línea de mandatos en la sentencia %EXEC.
- Pase los parámetros de sólo entrada directamente a un programa REXX definiendo variables de Net.Data globales y haciendo referencia a las variables. Para los programas REXX incorporados, haga referencia a las variables globales directamente en la fuente REXX.

Entorno de lenguaje SQL

En las secciones que vienen a continuación, se describen algunas técnicas de rendimiento acerca de la base de datos y del entorno de lenguaje SQL. Para conocer las consideraciones acerca del rendimiento de DB2, visite la Web en la dirección: <http://review.ibm.com/software/data/db2/performance>

Técnicas de bases de datos

El resumen siguiente describe a grandes trazos algunas de las técnicas de bases de datos más simples que pueden mejorar el acceso a la base de datos:

- Active la base de datos. Si se emite el mandato `db2 activate database nombreBasedatos`, las conexiones a la base de datos se realizan en un tiempo significativamente menor. Consulte el manual *DB2 Administration Guide* para obtener más información sobre el mandato `DB2 activate database`.
- Evite la conversión numérica. Cuando se están comparando un valor de columna y un valor literal, intente especificar los mismos atributos y tipos de datos. DB2 no utiliza un índice para la columna mencionada si el valor literal tiene una precisión mayor que la precisión de la columna. Si los dos elementos que se están comparando tienen tipos de datos diferentes, DB2 tendrá que convertir un valor u otro, lo que puede producir imprecisiones (debido a la precisión limitada de la máquina).

Por ejemplo, `EDUCLVL` es un valor entero de media palabra (`SMALLINT`). Especifique

```
... WHERE EDUCLVL < 11 AND EDUCLVL >= 2
```

En lugar de:

```
... WHERE EDUCLVL < 1.1E1 AND EDUCLVL > 1.3
```

- Evite rellenar series de caracteres. Intente utilizar la misma longitud de datos cuando compare un valor de columna de serie de caracteres de longitud fija con un valor literal. DB2 no utiliza ningún índice si la longitud del valor literal es mayor que la longitud de la columna.

Por ejemplo, EMPNO es CHAR(6) y DEPTNO es CHAR(3). Especifique:

```
... WHERE EMPNO > '000300' AND DEPTNO < 'E20'
```

En lugar de:

```
... WHERE EMPNO > '000300 ' AND DEPTNO < 'E20 '
```

- Evite el uso de patrones LIKE que empiecen por % o _. El signo de porcentaje (%) y el subrayado (_), cuando se utilizan en el patrón de un predicado LIKE, especifican una serie de caracteres que es similar al valor de columna de las filas que desea seleccionar. Cuando se utilizan para indicar caracteres en medio o al final de una serie de caracteres, los patrones LIKE pueden beneficiarse de los índices. Por ejemplo:

```
... WHERE LASTNAME LIKE 'J%SON%'
```

Sin embargo, cuando se utilizan al principio de una serie de caracteres, los patrones LIKE pueden impedir que DB2 utilice índices que pueden estar definidos en la columna LASTNAME para limitar el número de filas exploradas. Por ejemplo:

```
... WHERE LASTNAME LIKE '%SON'
```

Evite utilizar estos símbolos al principio de las series de caracteres, especialmente si está accediendo a una tabla especialmente grande.

Técnicas de entorno de lenguaje SQL

- Si un conjunto de resultados contiene un gran número de filas, puede especificar un subconjunto del conjunto de resultados que se devuelve al navegador utilizando START_ROW_NUM y RPT_MAX_ROWS. START_ROW_NUM especifica en qué fila debe empezar el subconjunto devuelto y RPT_MAX_ROWS especifica el número de filas que se deben devolver a la página. Entonces START_ROW_NUM puede utilizarse en un enlace para visualizar la siguiente página de resultados.
Tenga en cuenta que Net.Data vuelve a emitir la consulta para cada página porque la posición del cursor no se mantiene entre peticiones.
- Considere la posibilidad de llamar a un procedimiento almacenado que utilice SQL estático. El SQL dinámico está preparado en la ejecución, mientras que el SQL estático está preparado en la fase de precompilación. El entorno de lenguaje SQL utiliza SQL dinámico, que le permite ejecutar sentencias de SQL en la ejecución del programa. Dado que la preparación de las sentencias requiere tiempo de proceso adicional, el SQL estático puede ser más eficiente.
- Cuando pase parámetros al entorno de lenguaje SQL, considere la posibilidad de aprovecharse de la antememoria de preparación de DB2. En primer lugar, configure DB2 para preparar las sentencias conjuntamente con paquetes de almacenamiento en antememoria. A continuación, en la macro, establezca la variable de Net.Data DTW_USE_DB2_PREPARE_CACHE en

YES. Este valor vuelve a escribir la sentencia de SQL utilizando marcadores de parámetro, aprovechándose así de la antememoria de preparación de DB2, eliminando la necesidad de buscar conjuntos de resultados repetidos y, de este modo, mejorando el rendimiento.

Puesto que Net.Data utiliza la sustitución de variables para optimizar la consulta, tenga cuidado de manejar las comillas simples correctamente en las sentencias de SQL. Por ejemplo, si la columna devuelve una serie, utilice la función `DTW_ADDQUOTE()` para eludir las comillas simples de la serie.

Entornos de lenguaje System y Perl

Pase los parámetros de sólo entrada directamente al programa que el entorno de lenguaje System o Perl está invocando. Realice dicha acción definiendo variables globales de Net.Data y haciendo referencia a ellas. Para scripts Perl y programas externos, haga referencia a las variables en la línea de mandatos en la sentencia `%EXEC`. Para scripts Perl incorporados, haga referencia a las variables directamente en la fuente Perl.

Capítulo 8. Anotaciones cronológicas de Net.Data

Net.Data proporciona varias anotaciones cronológicas para que las utilice al supervisar el rendimiento de Net.Data y al resolver errores. Las anotaciones cronológicas de Net.Data incluyen:

Anotación cronológica de mensajes de error de Net.Data

Anotación cronológica que contiene todos los mensajes de error de Net.Data.

Anotación cronológica de Live Connection

Anotación cronológica que contiene los mensajes de error de Live Connection y la comunicación entre Net.Data, Live Connection y la base de datos DB2. La anotación cronológica está disponible para los entornos de lenguaje DTW_SQL y DTW_ODBC para las bases de datos DB2.

- “Anotación cronológica de mensajes de error de Net.Data”
- “Anotación cronológica de mensajes de error y de cliente de Live Connection” en la página 248

Anotación cronológica de mensajes de error de Net.Data

Net.Data graba los mensajes de error en el archivo de anotaciones de cronológicas de errores de Net.Data, `netdata.log`. El tamaño máximo de la anotación cronológica de errores la fija Net.Data en 500 KB, lo que equivale a 3000 entradas de anotación cronológica aproximadamente.

Puede examinar periódicamente el archivo de anotaciones cronológicas de errores o las copias archivadas para determinar si existen problemas en el sistema Net.Data.

%ara activar la anotación cronológica de errores de Net.Data:

- Establezca la variable de configuración de anotación cronológica de Net.Data, `DTW_LOG_DIR`:

`DTW_LOG_DIR víaacceso`

Donde *vía*acceso es el directorio donde desea que se almacene el archivo de anotaciones cronológicas de errores.

- En la macro, establezca la variable de anotación cronológica de Net.Data, `DTW_LOG_LEVEL`:

`@DTW_ASSIGN(DTW_LOG_LEVEL, "nivel")`

Donde *nivel* es el nivel de anotación cronológica. Puede tener los valores siguientes:

off Net.Data no anota errores. Éste es el valor por omisión.

error Net.Data anota los mensajes de error.

Esta sección describe los temas de anotación cronológica siguientes:

- “Planificación de la anotación cronológica de errores de Net.Data”
- “Control del nivel de anotación cronológica de Net.Data” en la página 247
- “Tipos de mensajes de error de Net.Data no anotados cronológicamente” en la página 247
- “Tamaño y rotación del archivo de anotaciones cronológicas de errores de Net.Data” en la página 247
- “Formato de la anotación cronológica de errores de Net.Data” en la página 248

Planificación de la anotación cronológica de errores de Net.Data

Al anotar cronológicamente los errores, necesita planificar los puntos siguientes:

- Determinación del espacio en disco:

Si piensa utilizar la anotación cronológica de errores, deberá dejar espacio en disco adicional para las anotaciones de errores.

- Configuración de Net.Data:

Si piensa controlar la anotación cronológica de errores para todo el sistema Net.Data, establezca una variable de configuración en el archivo de inicialización de Net.Data: DTW_LOG_DIR

Esta variable es necesaria para la anotación cronológica de errores, incluso si ha establecido la variable DTW_LOG_LEVEL en error o aviso (warning) en la macro. Consulte el apartado “DTW_LOG_DIR y DTW_LOG_LEVEL: Variables de anotación cronológica de errores” en la página 18 para obtener información sobre cómo actualizar el archivo de inicialización.

- Escritura de macros de Net.Data:

Establezca el nivel de anotación cronológica con la palabra clave DTW_LOG_LEVEL en la macro.

- Ejecución de Net.Data:

Si está utilizando la anotación cronológica de errores, puede buscar los errores del sistema Net.Data en las anotaciones cronológicas de errores y en los archivos archivadores.

- Ajuste:

Tenga presente que la anotación cronológica puede afectar al rendimiento. Consulte el apartado “Establecimiento del nivel de anotación cronológica de errores” en la página 241 para obtener información sobre temas de rendimiento.

Control del nivel de anotación cronológica de Net.Data

Puede especificar el nivel de anotación cronológica con la variable DTW_LOG_LEVEL. Defina esta palabra clave en la macro de Net.Data. La variable tiene tres valores:

off Net.Data no anota errores. Éste es el valor por omisión.

error Net.Data anota los mensajes de error.

warning
Net.Data anota los avisos, así como los mensajes de error.

Tipos de mensajes de error de Net.Data no anotados cronológicamente

Net.Data no anota cronológicamente los errores manejados explícitamente por una sección MESSAGE de la macro.

Tamaño y rotación del archivo de anotaciones cronológicas de errores de Net.Data

El tamaño máximo del archivo de anotaciones cronológicas puede ser de 500 KB. Con este tamaño, se incluirán 3000 entradas de anotación cronológica aproximadamente.

Cuando el archivo de anotaciones cronológicas alcance el tamaño máximo, se archivará en `netdata.logMMMDDAAAA_nn`

Donde:

MMM Es el mes, de enero a diciembre (Jan-Dec)

DD Es la fecha

AAAA Es el año

nn Número de 01 a 99, que identifica de forma exclusiva cada archivo archivador para un día determinado.

La anotación cronológica continúa en el archivo original.

Formato de la anotación cronológica de errores de Net.Data

Las entradas del archivo de anotaciones cronológicas tienen el formato siguiente:

[DD/MMM/AAAA:HH:MM:SS] [MACRO] [BLOCK] [núm_PID] [núm_TID]mensaje_error

Parámetros:

DD Fecha

MMM Mes, de enero a diciembre (Jan-Dec)

AAAA Año

HH Hora (de 00 a 23)

MM Minuto (de 00 a 59)

SS Número de segundos (de 00 a 59)

MACRO

Macro que ha generado el mensaje de error

BLOCK

Nombre del bloque HTML que ha generado el mensaje de error.

núm_PID

Número de ID del proceso que ha generado el mensaje de error. Este ID es necesario porque se pueden grabar múltiples procesos de Net.Data en el archivo de anotaciones cronológicas.

núm_TID

Número de ID de la hebra que ha generado el mensaje de error. Este ID es necesario porque se pueden grabar múltiples hebras del mismo proceso de Net.Data en el archivo de anotaciones cronológicas.

mensaje_error

Texto del mensaje de error

Anotación cronológica de mensajes de error y de cliente de Live Connection

Live Connection registra los mensajes y la comunicación entre Live Connection, Net.Data y la base de datos DB2 en el archivo de anotaciones cronológicas de Live Connection. El tamaño máximo de la anotación cronológica la fija Net.Data en 1 MB, lo que equivale a 1200 entradas de anotación cronológica aproximadamente.

Puede examinar periódicamente el archivo de anotaciones cronológicas o las copias archivadas para determinar si existen problemas con las clientas o la base de datos DB2.

Para activar la anotación cronológica de Live Connection:

Inicie Connection Manager con el atributo -l:

```
dtwcm -l  
[nivel]
```

Donde *nivel* es el nivel de anotación cronológica. Puede tener los valores siguientes:

normal

Live Connection anota todas las actividades de las cliettes, las sentencias de SQL y los mensajes de estado de DB2 relacionados así como los mensajes de error de Live Connection

minimal

Live Connection anota solamente la información esencial, por ejemplo las consultas de base de datos y el número de filas del conjunto de resultados.

Esta sección describe los temas de anotación cronológica siguientes:

- “Planificación de la anotación cronológica de Live Connection”
- “Control del nivel de anotación cronológica de Live Connection” en la página 250
- “Tipos de mensajes de Live Connection no anotados cronológicamente” en la página 250
- “Nombres de archivos de anotaciones cronológicas de Live Connection” en la página 250
- “Tamaño y rotación del archivo de anotaciones cronológicas de Live Connection” en la página 251
- “Formato de anotación cronológica de Live Connection” en la página 252

Planificación de la anotación cronológica de Live Connection

Cuando se anotan cronológicamente los errores, es necesario planificar los puntos siguientes:

- Determinación del espacio en disco:
Si piensa utilizar la anotación cronológica de errores, deberá dejar espacio en disco adicional para los archivos de anotaciones cronológicas.

- Ejecución de Connection Manager:

Active la anotación cronológica entrando un atributo en el mandato dtwcm. Consulte el apartado “Anotación cronológica de mensajes de error y de cliette de Live Connection” en la página 248 para conocer la sintaxis.

Si está utilizando la anotación cronológica, puede buscar en las anotaciones cronológicas y los archivos archivadores los errores de las cliettes.

- **Ajuste:**
Tenga presente que la anotación cronológica puede afectar al rendimiento. Consulte el apartado “Establecimiento del nivel de anotación cronológica de errores” en la página 241 para obtener información sobre temas de rendimiento y el apartado “Control del nivel de anotación cronológica de Live Connection”.

Control del nivel de anotación cronológica de Live Connection

Puede especificar el nivel de anotación cronológica en el mandato dtwcm mientras invoca Connection Manager. El atributo -l del mandato dtwcm tiene dos valores:

normal

Live Connection anota todas las actividades de las cliettes, las sentencias de SQL y los mensajes de estado de DB2 relacionados así como los mensajes de error de Live Connection

minimal

Live Connection sólo anota los mensajes esenciales. Esta opción proporciona menos mensajes en la anotación cronológica.

Tipos de mensajes de Live Connection no anotados cronológicamente

Live Connection no anota cronológicamente los errores de Net.Data ni los errores manejados explícitamente por una sección MESSAGE de la macro.

Nombres de archivos de anotaciones cronológicas de Live Connection

Live Connection crea un archivo de anotaciones cronológicas para Connection Manager y para cada cliette. La lista siguiente describe los formatos de archivo:

Archivo de Connection Manager

Formato:

conman-id_proceso-DDMMMAAAHHMMSS.log

Parámetros:

id_proceso

Identificador del proceso de Connection Manager

DD

Fecha

MMM

Mes, de enero a diciembre (Jan-Dec)

AAAA

Año

HH

Hora, reloj de 24 horas

MM

Minutos

SS Segundos

Ejemplo:

conman-513-01Feb1999095639.log

Archivo de cliette

Formato:

cliectt-id_proceso-DDMMMAAAHHMMSS.log

Parámetros:

id_proceso

Identificador de la hebra de cliette

DD

Fecha

MMM

Mes, de enero a diciembre (Jan-Dec)

AAAA

Año

HH

Hora, reloj de 24 horas

MM

Minutos

SS Segundos

Ejemplo:

cliectt-592-01Feb1999095647.log

Tamaño y rotación del archivo de anotaciones cronológicas de Live Connection

El tamaño máximo del archivo de anotaciones cronológicas puede ser de 1 MB. Con este tamaño, se incluirán 6.000 entradas de anotación cronológica aproximadamente. Cuando el archivo de anotaciones cronológicas alcance el tamaño máximo, el proceso cerrará el archivo de anotaciones cronológicas original, creará un archivo de anotaciones cronológicas nuevo y continuará anotando en el archivo nuevo.

Los archivos de anotaciones cronológicas están ubicados en el mismo directorio que dtwcm y dtwcdb2

Formato de anotación cronológica de Live Connection

Las entradas del archivo de anotaciones cronológicas tienen el formato siguiente:

--*tipo_proceso*-DD/MMM/AAAA:HH:MM:SS-núm_PID--
texto_mensaje

Parámetros:

tipo_proceso

dtwcm o cliet, en función de si ha anotado el mensaje Connection Manager o una cliette.

DD Fecha

MMM Mes, de enero a diciembre (Jan-Dec)

AAAA Año

HH Hora (de 00 a 23)

MM Minuto (de 00 a 59)

SS Número de segundos (de 00 a 59)

núm_PID

Número de ID del proceso que ha generado el mensaje.

texto_mensaje

Texto del mensaje.

Ejemplo 1: Entrada de anotación cronológica de Connection Manager.

```
--dtwcm-02/Mar/1999:13:43:07-330--
Creating connection manager ...successfully
Reading configuration info ...
Completing initialization ...
Initializing cm server ... successfully
Initializing NLS environment ... successfully
Detecting cliette ./dtwcdb2 for DTW_SQL:CELDIAL:
  Min process(es) = 1,
  Priv Port = 7100.
Starting 1 cliettes for DTW_SQL:CELDIAL.
Started: ./dtwcdb2 7128 7100 7200 DTW_SQL:CELDIAL LOG_MAX , pid: 213
1 cliettes for DTW_SQL:CELDIAL started.
...
```

Ejemplo 2: Entrada de anotación cronológica de cliette.

```
--cliет-02/Mar/1999:13:43:08-335--
Cliette starting ...
Cliette: DTW_SQL:SAMPLE, database: SAMPLE, user: *USE_DEFAULT
```



```
Making a new connection to database: SAMPLE, user: *USE_DEFAULT.  
Calling SQLAllocHandle for environment ...  
Calling SQLAllocHandle for connection ...  
Calling SQLSetConnectAttr ...  
Calling SQLConnect ...  
Connecting to database: SAMPLE sucessfully.  
Telling CM the cliette is ready ...  
Ready and waiting for command from CM ...
```

Apéndice A. Bibliografía

Biblioteca técnica de Net.Data

La Biblioteca técnica de Net.Data está disponible en el sitio Web de Net.Data en

<http://www.ibm.com/software/data/net.data/library.html>

Documento	Descripción
<ul style="list-style-type: none">• <i>Net.Data Guía de administración y programación para OS/390</i>• <i>Net.Data Guía de administración y programación para OS/2, Windows NT y UNIX</i>• <i>Net.Data Guía de administración y programación para OS/400</i>	Contiene información de conceptos y tareas sobre cómo instalar, configurar e invocar Net.Data. También describe cómo escribir macros de Net.Data, utilizar técnicas de rendimiento de Net.Data, utilizar entornos de lenguaje de Net.Data, gestionar conexiones y utilizar anotaciones cronológicas y rastreos de Net.Data para resolver problemas y ajustar el rendimiento.
<i>Net.Data Guía de consulta</i>	Describe el lenguaje de macros, las variables y funciones incorporadas de Net.Data.
<i>Net.Data Language Environment Interface Reference</i>	Describe la interfaz de entorno de lenguaje de Net.Data.
<i>Net.Data Mensajes y códigos</i>	Lista mensajes de error y códigos de retorno de Net.Data.

Apéndice B. Net.Data para AIX

En el archivo README que se envía con Net.Data, se incluyen detalles para AIX. El archivo README incluye la información siguiente:

- Requisitos
- Instalación
- Configuración
- Desinstalación

Carga de bibliotecas compartidas para los entornos de lenguaje

Al crear un entorno de lenguaje en la plataforma AIX, necesita cargar las bibliotecas compartidas. En AIX, el entorno de lenguaje es necesario para proporcionar una rutina que Net.Data llama y que devuelve las direcciones de las rutinas de interfaz de entorno de lenguaje tales como `dtw_initialize()` y `dtw_execute()`.

Net.Data utiliza la estructura `dtw_fp` para recuperar de un entorno de lenguaje de AIX los punteros a las rutinas de interfaz de entorno de lenguaje, y dicha estructura tiene este formato:

```
typedef struct dtw_fp {  
    int (* dtw_initialize_fp)(); /* puntero de función dtw_initialize */  
    int (* dtw_execute_fp)(); /* puntero de función dtw_execute */  
    int (* dtw_cleanup_fp)(); /* puntero de función dtw_cleanup */  
} dtw_fp_t;
```

Net.Data pasa esta estructura al entorno de lenguaje como un parámetro de la rutina `dtw_getFp()` cuando se carga la biblioteca compartida.

La estructura `dtw_fp` se pasa como parámetro único. Esta estructura contiene un campo para cada interfaz soportada y el establecimiento de estos campos es responsabilidad del entorno de lenguaje. Si el entorno de lenguaje está proporcionando la interfaz especificada, establece el campo en el puntero de función de dicha interfaz. Si no está proporcionando la interfaz especificada, establece el campo en NULL. La rutina `dtw_getFp()` de la plantilla de programa muestra una implementación correcta de esta rutina.

Para que Net.Data obtenga el puntero a esta rutina cuando se carga la biblioteca compartida, la rutina `dtw_getFp` debe ser el primer punto de entrada especificado en el archivo de exportación de la biblioteca compartida. A continuación se muestra un archivo de exportación de ejemplo para una biblioteca llamada `dtwsampshr.o` que soporta todas las rutinas de interfaz de entorno de lenguaje:

```
#!dtwsampshr.o
dtw_getFp
dtw_initialize
dtw_execute
dtw_cleanup
```

Mejora del rendimiento en el entorno REXX

Si tiene muchas llamadas al entorno de lenguaje REXX en el sistema AIX, considere la posibilidad de establecer la variable de entorno `RXQUEUE_OWNER_PID` en 0. Las macros que efectúan muchas llamadas al entorno de lenguaje REXX pueden generar fácilmente muchos procesos, agotando los recursos del sistema.

Puede establecer la variable de entorno de uno de estos tres modos:

- En la macro, utilizando la función incorporada `DTW_SETENV`:

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

- En el archivo de entorno del sistema AIX:

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

Este método afecta al comportamiento de REXX para toda la máquina.

- En el archivo de entorno de servidor Web HTTP; por ejemplo, para Domino Go Webserver, inserte la sentencia siguiente:

```
InheritEnv RXQUEUE_OWNER_PID = 0
```

Este método afecta al comportamiento de REXX para el servidor Web.

Consideraciones acerca de NLS

Net.Data se ejecuta utilizando la misma página de códigos que el servidor Web. Para que Net.Data utilice la página de códigos correcta para el entorno nacional correspondiente, el servidor Web debe estar utilizando la página de códigos correcta. Por ejemplo, si está utilizando un IBM Internet Connection Server y desea utilizar una página de códigos coreana, detenga el servidor y reinicielo utilizando el entorno nacional coreano:

```
stopsrc httpd
startsrc -s httpd -e "LC_ALL=ko_KR"
```

Si la macro contiene caracteres UTF-8 o si está conectándose a una base de datos DB2 que contiene datos Unicode, establezca la variable de configuración `DTW_UNICODE` en el archivo INI en UTF8. Net.Data soporta actualmente en la macro los caracteres UTF-8, pero no los caracteres UTF-16. Sin embargo, Net.Data puede procesar datos de la base de datos DB2 que estén codificados en UTF-8 o UCS-2. La salida de Net.Data es siempre en UTF-8.

Consejo respecto al rendimiento: Si está ejecutando en un entorno nacional de doble byte y las funciones incorporadas de trabajo o serie siempre procesan series de caracteres de un solo byte, establezca DTW_MBMODE en NO para ahorrarse la conversión innecesaria.

Apéndice C. Asistentes de Net.Data

Los asistentes de Net.Data están diseñados para proporcionarle un modo rápido y fácil de crear aplicaciones Net.Data personalizadas. Simplemente seleccione un asistente, responda unas preguntas y Net.Data le creará una aplicación personalizada.

Net.Data proporciona las SmartGuides siguientes para que las utilice mientras aprende a crear macros y a utilizar las características de Net.Data:

Drilldown (Sondeo)

Esta SmartGuide toma las tablas de base de datos existentes y crea una aplicación de sondeo habilitada para la Web que le permite acceder a diferentes niveles de detalle de los datos. Opcionalmente, la SmartGuide Drilldown puede conectarse a la base de datos para reunir información de la misma. Puede personalizar un máximo de cinco informes Web. La macro generada utiliza la información de las claves principal y foránea almacenada en la base de datos para enlazar los informes Web automáticamente.

Stored Procedure (Procedimiento almacenado)

Esta SmartGuide se conecta a la base de datos y recupera una lista de todos los procedimientos almacenados que están registrados en la base de datos. Seleccione un procedimiento almacenado y la SmartGuide generará una macro de Net.Data que llamará al procedimiento almacenado. Entonces puede modificar la macro generada o integrarla en las aplicaciones de Net.Data existentes.

Contacts (Contactos)

Esta SmartGuide genera un listín habilitado para la Web para almacenar nombres, direcciones, números de teléfono e información adicional importante sobre contactos. Viene con una función de búsqueda que proporciona el acceso rápido a los contactos. La aplicación de contactos generada puede incluir un informe breve o uno detallado. Adicionalmente, puede incluir un informe personalizado.

Consulte el sitio Web de Net.Data en <http://www.ibm.com/software/data/net.data> para obtener la versión más reciente del paquete de SmartGuide de Net.Data.

Este apéndice describe los temas siguientes:

- “Antes de empezar” en la página 262
- “Ejecución de los Asistentes” en la página 262

Antes de empezar

Para ejecutar las SmartGuide y generar las macros de Net.Data, deberá tener instalado el software siguiente:

- Java Development Kit (JDK) o Java Runtime Environment (JRE) 1.1.x
- Net.Data Versión 2 o posterior
- IBM Universal Database (UDB) 5.0 o posterior
- REXX (necesario para la SmartGuide Drilldown)

Ejecución de los Asistentes

Los asistentes de Net.Data se inician desde la línea de mandatos y están contenidos en el archivo NetDataSmartGuides.jar.

Para iniciar un asistente con el JDK (Java Development Kit):

1. Añada la línea siguiente a la variable de entorno CLASSPATH.

*[Víaacceso]*NetDataSmartGuides.jar

donde *[Víaacceso]* es la vía de acceso opcional al archivo NetDataSmartGuides.jar.

2. Si desea utilizar la característica de conexión de base de datos del asistente Drilldown y Stored Procedure, añada el controlador UDB JDBC a la variable de entorno CLASSPATH.

Para los sistemas operativos Windows NT y OS/2, añada la línea siguiente a la variable de entorno CLASSPATH:

*[Víaacceso]*NetDataSmartGuides.jar;*[VíaaccesoInstalaciónUDB]*\\java\db2java.zip

Para los sistemas operativos UNIX, añada la línea siguiente a la variable de entorno CLASSPATH:

*[Víaacceso]*NetDataSmartGuides.jar:*[VíaaccesoInstalaciónUDB]*\\java\db2java.zip

Donde *[Víaacceso]* es la vía de acceso opcional al archivo NetDataSmartGuides.jar y *[VíaaccesoInstalaciónUDB]* es la vía de acceso a la instalación de UDB, por ejemplo C:\\SQLLIB.

3. Inicie el asistente.

- Para los sistemas operativos UNIX, escriba el mandato siguiente:

java TaskGuide LaunchPad.class

- Para los sistemas operativos Windows NT y OS/2, ejecute el archivo siguiente:

SmartGuides.cmd

Se abre un área de ejecución con la lista de asistentes disponibles, como se muestra en la Figura 29.

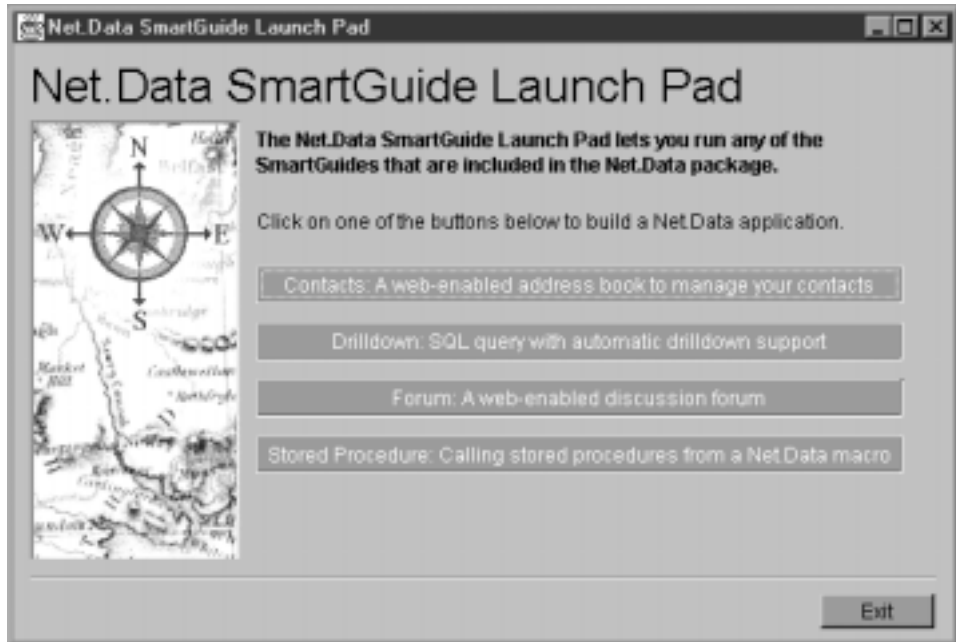


Figura 29. Área de ejecución de asistentes

4. Pulse el nombre del asistente que desea ejecutar.

Para iniciar un asistente con el JRE (Java Runtime Environment):

1. Para el sistema operativo Windows NT, elija **Inicio->Programas->Net.Data->SmartGuides**, que ejecuta un archivo de proceso por lotes llamado SMARTGUIDES.BAT. Para otros sistemas operativos, escriba el mandato siguiente para ejecutar los asistentes:

```
jre -cp [Víaacceso]NetDataSmartGuides.jar TaskGuide LaunchPad.class
```

donde [Víaacceso] es la vía de acceso opcional al archivo NetDataSmartGuides.jar.

Se abre un área de ejecución con la lista de asistentes disponibles, como se muestra en la Figura 29.

2. Si desea utilizar la característica de conexión de base de datos del asistente Drilldown y Stored Procedure, escriba el mandato siguiente:

Para los sistemas operativos Windows NT y OS/2:

```
jre -cp [Víaacceso]NetDataSmartGuides.jar;[VíaaccesoInstalaciónUDB]
\java\db2java.zip TaskGuide LaunchPad.class
```

Para los sistemas operativos UNIX:

```
jre -cp [Víaacceso]NetDataSmartGuides.jar:[VíaaccesoInstalaciónUDB]  
      \java\db2java.zip TaskGuide LaunchPad.class
```

Donde *[Víaacceso]* es la vía de acceso opcional al archivo NetDataSmartGuides.jar y *[VíaaccesoInstalaciónUDB]* es la vía de acceso a la instalación de UDB, por ejemplo C:\SQLLIB.

3. Pulse el nombre del asistente que desea ejecutar.

Apéndice D. Creación de sentencias de SQL con Net.Data SQL Assist

Net.Data SQL Assist es un creador de sentencias de SQL basado en Java que proporciona una GUI fácil de usar para guiarle durante el proceso de creación de sentencias de SQL. Con Net.Data SQL Assist puede:

- Crear sentencias SELECT, INSERT, UPDATE y DELETE (incluyendo SELECT DISTINCT)
- Crear múltiples condiciones utilizando la búsqueda de valor, AND u OR y campos de entrada sensibles al tipo
- Definir UNIONES de tabla (interna, externa derecha y externa izquierda)
- Seleccionar columnas a visualizar
- Seleccionar el orden de clasificación
- Entrar variables definidas por el usuario que se deberán utilizar en las condiciones, los valores y la clasificación

Después de crear la sentencia de SQL, puede:

- Guardar la sentencia de SQL como un archivo
- Generar y guardar una macro que contenga la sentencia de SQL
- Copiar la sentencia de SQL o la macro en el área común

Este apéndice describe los temas siguientes:

- “Antes de empezar”
- “Ejecución de Net.Data SQL Assist” en la página 266

Antes de empezar

Para ejecutar Net.Data SQL Assist, deberá tener instalado el software siguiente:

- Java Development Kit (JDK) o Java Runtime Environment (JRE) 1.1.x
- Una base de datos habilitada para JDBC

Consulte la documentación de la base de datos para obtener detalles adicionales sobre cómo acceder a la fuente de datos mediante JDBC y sobre cualquier otro arranque de servidor que pueda ser necesario. Por ejemplo, cuando se accede a una fuente de datos DB2 UDB v5.0 de forma remota, el servidor de base de datos debe tener el servidor JDBC (db2jstrt) en ejecución.

Ejecución de Net.Data SQL Assist

Net.Data SQL Assist se inicia desde la línea de mandatos y está contenido en el archivo `{dir_inst}/assist/NetDataAssist.jar`.

Para iniciar Net.Data Assist con el JDK (Java Development Kit):

Entre el mandato siguiente para iniciar Net.Data Assist:

```
java -classpath  
%CLASSPATH%;{dir_inst}/assist/NetDataAssist.jar NetDataAssist
```

Para iniciar Net.Data Assist con el JRE (Java Runtime Environment):

Entre el mandato para iniciar Net.Data Assist:

```
jre -cp  
%CLASSPATH%;{dir_inst}/assist/NetDataAssist.jar NetDataAssist
```

Pulse el botón **Next** para navegar por las ventanas utilizadas para iniciar la sesión, crear una sentencia de SQL y generar una macro de Net.Data.

Apéndice E. Utilización de los plug-in de NetObjects Fusion (NOF) con servlets de Net.Data

Net.Data proporciona un plug-in de NetObjects Fusion para las servlets de Net.Data. Las servlets de Net.Data se describen en el apartado “Servlets de Net.Data” en la página 95.

Puede utilizar NetObjects Fusion (NOF) para integrar las macros de Net.Data existentes, que proporcionarán una integración mejor con la gestión de sitios Web y una interfaz gráfica de usuario fácil de utilizar.

Este apéndice describe los temas siguientes:

- “Acerca del plug-in de NetObjects Fusion”
- “Instalación del plug-in de NetObjects Fusion” en la página 268
- “Configuración del plug-in de Net.Data para NetObjects Fusion” en la página 268
- “Publicación de servlets con el plug-in NOF” en la página 272

Acerca del plug-in de NetObjects Fusion

El plug-in NetDataServlet.NFX funciona con las servlets de Net.Data. Este plug-in NOF soporta la invocación de una macro de Net.Data existente o de una sola función de Net.Data. El plug-in genera el HTML para invocar Net.Data como una servlet o una SSI (inclusión del extremo del servidor). Cuando el servidor Web invoca Net.Data, se ejecuta la macro o función de Net.Data. Utilice el plug-in de servlet de Net.Data para incorporar cualquiera de las servlets de macro y función en un sitio Web gestionado por NOF, lo cual se describe en la Figura 30 en la página 268. El plug-in proporciona muchos de los parámetros de macro o función básicos, con los valores por omisión seleccionados para automatizar la creación de una macro. Para utilizar el plug-in, deberá instalar y configurar NOF y los archivos de plug-in.

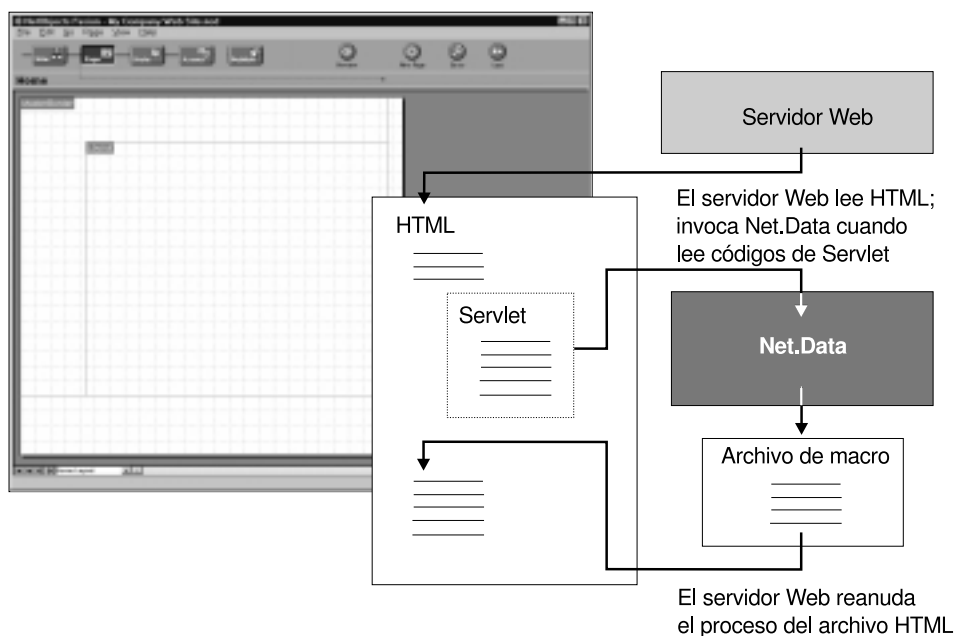


Figura 30. Plug-in de servlets de Net.Data

Instalación del plug-in de NetObjects Fusion

Requisitos de software y hardware:


Los plug-in NOF necesitan NetObjects Fusion Versión 2.0 o posterior.

Para instalar: Copie <dir_inst>\fx\NetDataServlet.nfx y <dir_inst>\fx\NetDataServlet.gif en el directorio <NetObjects fusion>\components\.

Configuración del plug-in de Net.Data para NetObjects Fusion

Puede cambiar las propiedades de la servlet con la que está trabajando utilizando NOF.

1. Abra NetObjects Fusion.
2. En la paleta **Tools** de NetObjects Fusion (NOF), seleccione el botón

NetObjects Components:  Se visualizan los botones del plug-in en la parte inferior de la paleta **Tools**.

3. De estos seis botones de la paleta **Tools**, seleccione el botón **NetObjects**

Components: 

4. En la cuadrícula de NOF, marque el área para especificar dónde desea poner el plug-in seleccionado. Éste es el lugar donde se visualizarán los resultados de la servlet. Se abre la ventana Installed Components, visualizando una lista de plug-in entre los que elegir. Si el plug-in de servlet no está en la lista, utilice los campos de vía de acceso y nombre de archivo para especificar el nombre de archivo de plug-in, NetDataServlet.NFX, a utilizar con la servlet de función o macro
5. Seleccione el plug-in de servlet en la lista y pulse el pulsador **OK**.
El plug-in se convierte en un objeto en la cuadrícula de NOF.

Modificación de las propiedades del plug-in

Puede modificar las servlets de función y macro utilizando el plug-in de servlet de Net.Data.

Para modificar servlet de Net.Data con NOF:

1. En la cuadrícula de NOF, marque el área para especificar dónde desea poner el plug-in seleccionado. Éste es el lugar donde se visualizarán los resultados de la servlet. Se abre la ventana Installed Components, visualizando una lista de plug-in entre los que elegir. Si el plug-in de servlet no está en la lista, utilice los campos de vía de acceso y nombre de archivo para especificar el nombre de archivo de plug-in, NetDataServlet.NFX, a utilizar con la servlet de función o macro.
2. Seleccione el plug-in de servlet de Net.Data en la lista y pulse el pulsador **OK**.
El plug-in se convierte en un objeto en la cuadrícula de NOF.
3. Seleccione y personalice las propiedades del plug-in de servlet de Net.Data:
 - a. Seleccione el plug-in de servlet de Net.Data en la cuadrícula de NOF. Se abre la paleta **Properties** de NOF, visualizando las propiedades del plug-in como se muestra en la Figura 31 en la página 270.



Figura 31. Paleta Properties de Servlet de Net.Data

Propiedades para las servlets de Net.Data:

Puede personalizar las propiedades siguientes:

Servlet name (Nombre de servlet)

Seleccione el nombre de la servlet que desea invocar: función o macro. En función del nombre de servlet que seleccione, se visualizarán propiedades diferentes.

Servlet type (Tipo de servlet)

Seleccione el tipo de servlet que desea: SSI, HREF o FORM Submit Button. En función del tipo de servlet que desee, se visualizarán propiedades diferentes.

Submit label (Etiqueta de sometimiento)

Si selecciona un tipo FORM Submit Button, especifique el texto para la etiqueta de sometimiento. De lo contrario, esta propiedad no se visualizará.

Server URL (URL de servidor)

Si selecciona un tipo de servlet HREF, especifique el URL de servidor en el servidor Web habilitado para servlet. Si selecciona SSI, esta propiedad no se visualizará.

Macro name (Nombre de macro)

Si selecciona el nombre de servlet de macro, especifique el nombre de una macro de Net.Data existente a ejecutar. De lo contrario, esta propiedad no se visualizará.

HTML block name (Nombre de bloque HTML)

Si selecciona el nombre de servlet de macro, especifique el nombre del bloque HTML en la macro de Net.Data a ejecutar. De lo contrario, esta propiedad no se visualizará.

Function type (Tipo de función)

Si selecciona el nombre de servlet de función, seleccione el tipo de función a ejecutar: Function (Función) o SQL. De lo contrario, esta propiedad no se visualizará.

Language env (Entorno de lenguaje)

Si selecciona el nombre de servlet de función, especifique el entorno de lenguaje de Net.Data a utilizar. De lo contrario, esta propiedad no se visualizará.

Statement (Sentencia)

Si selecciona el nombre de servlet de función, especifique la sentencia a ejecutar. De lo contrario, esta propiedad no se visualizará.

Database (Base de datos)

Si selecciona el nombre de servlet de función, especifique el nombre de la base de datos a utilizar. De lo contrario, esta propiedad no se visualizará.

of other parameters (Número de otros parámetros)

Especifique el número de otros parámetros que se deben pasar a Net.Data (máximo: 25). Para cada parámetro, entre el nombre del parámetro y su valor (opcional).

Before HREF text (Antes del texto HREF)

Si selecciona un tipo de servlet HREF, especifique el texto que debe aparecer antes del texto que debe aparecer antes del código HTML <a href>. De lo contrario, esta propiedad no se visualizará (opcional).

Inside HREF text (Dentro del texto HREF)

Si selecciona un tipo de servlet HREF, especifique el texto que debe aparecer dentro del código HTML <a href>. De lo contrario, esta propiedad no se visualizará (opcional).

After HREF text (Después del texto HREF)

Si selecciona un tipo de servlet HREF, especifique el texto que

debe aparecer después del texto que debe aparecer después del código HTML <a href>. De lo contrario, esta propiedad no se visualizará (opcional).

SQL Reminder! (¡Recordatorio de SQL!)

Si selecciona un tipo de servlet HREF y especifica un tipo de función SQL, se visualiza un mensaje con un recordatorio que indica que la sentencia de SQL HREF debe utilizar un carácter de signo más (+) para los caracteres de espacio (.). Este texto no se puede modificar ni se visualiza después de que se haya publicado la página. De lo contrario, esta propiedad no se visualizará.

- b. Después de haber definido las propiedades para la página, pulse el pulsador **Publish** para crear y publicar las páginas Web con el plug-in NOF de servlet de Net.Data.

Nota: Si selecciona un tipo de servlet SSI, la extensión del archivo de página Web debe ser .shtml. Puede establecer este valor como el valor por omisión de página desde la paleta de NOF **Properties**, seleccionando la pestaña de cuaderno **Page**, pulsando el pulsador **Custom names** y entrando .shtml en el campo **Extension Type**.

Publicación de servlets con el plug-in NOF

Después de haber establecido las propiedades para la página, pulse el pulsador **Publish** para crear y publicar las páginas Web con el plug-in.

Apéndice F. Macro de ejemplo de Net.Data

Esta aplicación de macro de ejemplo visualiza una lista de nombres de empleados a partir de la cual el usuario de aplicación puede obtener información adicional acerca de un empleado individual seleccionando el nombre del empleado en la lista. La macro utiliza el entorno de lenguaje SQL para consultar en la tabla EMPLOYEE los nombres de empleado y la información acerca de un empleado específico.

La macro utiliza un archivo de inclusión (include), que contiene el bloque DEFINE para la macro.

La Figura 32 en la página 274 muestra la macro de ejemplo. La Figura 33 en la página 276 muestra el archivo de inclusión.

```

%{***** Sample Macro *****)
*   FileName = sqlsamp1.d2w
*   Description:
*       This Net.Data macro queries...
*       - The EMPLOYEE table to create a selection list of
*         employees for display at a browser
*       - The EMPLOYEE table to obtain additional information
*         about an individual employee
*
*****}
%{*****}
*   Include for global DEFINES -
*****}
%INCLUDE "sqlsamp1.hti"
%{*****}
*   Function: queryDB           Language Environment: SQL
*   Description: Queries the table designated by the variable myTable and
*   creates a selection list from the result. The value of the variable
*   myTable is specified in the include file sqlsamp1.hti.
*****}
%FUNCTION(DTW_SQL) queryDB() {
    SELECT FIRSTNME FROM $(myTable)
%MESSAGE {
    -204: {<p><b>ERROR -204: Table $(myTable) not found. </b>
        <p>Be sure the correct include file is being used.</p>
        %} : exit
    +default: "WARNING $(RETURN_CODE)" : continue
    -default: "Unexpected ERROR $(RETURN_CODE)" : exit
}%}

%}

%REPORT {
<select name=emp_name>
%ROW{
<option>$(V1)
}%}
</select>
}%}
%}

%{*****}
*   Function: fname           Language Environment: SQL
*   Description: Queries the table designated by the variable myTable for
*   additional information about the employee identified by the
*   variable emp_name.
*****}
%FUNCTION(DTW_SQL) fname(){
SELECT FIRSTNME, PHONENO, JOB FROM $(myTable) WHERE FIRSTNME='$(emp_name)'
%MESSAGE {
    -204: "Error -204: Table not found "
    -104: "Error -104: Syntax error"
    100: "Warning 100: No records" : continue
    +default: "Warning $(RETURN_CODE)" : continue
    -default: "Unexpected SQL error" : exit
}%}
%}
%}

```

```
%{*****
*   HTML block: INPUT                               Title: Dynamic Query Selection   *
*                                                                                       *
*   Description: Queries the EMPLOYEE table to create a selection list *
*                  of the employees for display at the browser *
*****%}
%HTML(INPUT) {
<html>
<head>
<title>Generate Employee Selection List</title>
</head>
<body>
<h3>$(exampleTitle)</h3>
<p>This example queries a table and uses the result to create
a selection list using a <em>%REPORT</em> block.
<hr>
<form method="post" action="report">
@queryDB(<input type="submit" value="Select Employee" />
</form>
<hr />
</body>
</html>
%}
```

Figura 32. Macro de ejemplo (Pieza 2 de 3)

```
%{*****
*   HTML block:   REPORT                               *
*   Description: Queries the EMPLOYEE table to obtain additional information *
*                  about an individual employee *
*****%}
%HTML(REPORT) {
<html>
<head>
<title>Obtain Employee Information</title>
</head>
<body>
<h3>You selected employee name = $(emp_name)</h3>
<p>Here is the information for that employee:
<pre>
@fname()
</pre>
<hr><a href="input">Return to previous page</a>
</body>
</html>
%}

%{      End of Net.Data macro 1 %}
```

Figura 32. Macro de ejemplo (Pieza 3 de 3)

```

=====
%{***** Include File *****}
*   FileName = sqlsamp1.hti                               *
*   Description:                                           *
*       This include file provides global DEFINES for the sqlsamp1.d2w   *
*       Net.Data macro.                                       *
*****%}
#define {
    emp_name    = ""
    reposition = sign
    exampleTitle = "Sample Macro"
    myTable = "EMPLOYEE"
    DATABASE = "sample"
%}

%{    End of include file  %}

```

Figura 33. Archivo de inclusión

Avisos

Es posible que IBM no comercialice en todos los países algunos productos, servicios o características descritos en este manual. Consulte al representante local de IBM para obtener información sobre los productos y servicios que actualmente pueden adquirirse en su zona geográfica. Cualquier referencia a un producto, programa o servicio de IBM no pretende afirmar ni implicar que sólo se puede utilizar dicho producto, programa o servicio de IBM. En su lugar se puede utilizar cualquier producto, programa o servicio funcionalmente equivalente que no infrinja ninguno de los derechos de propiedad intelectual de IBM. Sin embargo, es responsabilidad del usuario evaluar y verificar el funcionamiento de cualquier producto, programa o servicio que no sea de IBM.

IBM puede tener patentes o solicitudes de patentes en tramitación que afecten al tema tratado en este documento. La posesión de este documento no confiere ninguna licencia sobre dichas patentes. Puede realizar consultas sobre licencias escribiendo a:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
E.E.U.U.

En el caso de consultas sobre licencias referentes a información de doble byte (DBCS), consulte al Departamento de Propiedad Intelectual de IBM en su país o envíe consultas por escrito a:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japón

El párrafo siguiente no es aplicable al Reino Unido ni a ningún país en el que tales disposiciones sean incompatibles con la legislación local:
INTERNATIONAL BUSINESS MACHINES CORPORATION PROPORCIONA ESTA PUBLICACIÓN "TAL CUAL", SIN GARANTÍA DE NINGUNA CLASE, NI EXPLÍCITA NI IMPLÍCITA, INCLUIDAS, PERO SIN LIMITARSE A ELLAS, LAS GARANTÍAS IMPLÍCITAS DE NO VULNERACIÓN DE DERECHOS, COMERCIALIZACIÓN O IDONEIDAD PARA UN FIN DETERMINADO. Algunos estados no permiten la exclusión de garantías expresas o implícitas en determinadas transacciones, por lo que es posible que esta declaración no sea aplicable en su caso.

Esta publicación puede contener inexactitudes técnicas o errores tipográficos. Periódicamente se efectúan cambios en la información aquí contenida; dichos cambios se incorporarán a las nuevas ediciones de la publicación. IBM puede efectuar, en cualquier momento y sin previo aviso, mejoras y/o cambios en los productos y/o programas descritos en esta publicación.

Las referencias hechas en esta publicación a sitios Web que no son de IBM se proporcionan sólo para la comodidad del usuario y no constituyen un aval de esos sitios Web. La información contenida en esos sitios Web no forma parte de la información del presente producto IBM y el usuario es responsable de la utilización de esos sitios Web.

Cuando envía información a IBM, IBM puede utilizar o distribuir dicha información en la forma en que IBM considere adecuada, sin contraer por ello ninguna obligación con el remitente.

Los licenciarios de este programa que deseen obtener información sobre él con el fin de habilitar: (i) el intercambio de información entre programas creados de forma independiente y otros programas (incluido este) y (ii) el uso mutuo de la información intercambiada, deben ponerse en contacto con:

IBM Canada Limited
Office of the Lab Director
1150 Eglinton Ave. East
North York, Ontario
M3C 1H7
CANADÁ

Dicha información puede estar disponible, sujeta a los términos y condiciones apropiados, incluido en algunos casos, el pago de una tarifa.

El programa bajo licencia descrito en este manual y todo el material bajo licencia asociado a él, los proporciona IBM según los términos del Convenio del Cliente IBM, el Convenio Internacional de Licencia de Programas de IBM o cualquier convenio equivalente entre el usuario e IBM.

Los datos de rendimiento contenidos en este documento se obtuvieron en un entorno controlado. Por tanto, los resultados obtenidos en otros entornos operativos pueden variar significativamente. Algunas mediciones pueden haberse hecho en sistemas experimentales y no es seguro que estas mediciones sean las mismas en los sistemas disponibles comercialmente. Además, algunas mediciones pueden haberse calculado mediante extrapolación. Los resultados reales pueden variar. Los usuarios del presente manual deben verificar los datos aplicables para su entorno específico.

La información referente a productos que no son de IBM se ha obtenido de los proveedores de esos productos, de sus anuncios publicados o de otras

fuentes disponibles públicamente. IBM no ha probado esos productos y no puede confirmar la exactitud del rendimiento, la compatibilidad ni cualquier otra afirmación referente a productos no IBM. Las preguntas sobre las prestaciones de productos no IBM deben dirigirse a los proveedores de esos productos.

Todas las declaraciones de intenciones de IBM están sujetas a cambio o cancelación sin previo aviso, y sólo representan objetivos.

Esta publicación puede contener ejemplos de datos e informes que se utilizan en operaciones comerciales diarias. Para ilustrarlos de la forma más completa posible, los ejemplos incluyen nombre de personas, empresas, marcas y productos. Todos estos nombres son ficticios y cualquier similitud con nombres y direcciones utilizados por una empresa real es totalmente no intencionada.

LICENCIA DE COPYRIGHT:

Este manual puede contener programas de aplicación de ejemplo escritos en lenguaje fuente, que muestra técnicas de programación en diversas plataformas operativas. Puede copiar, modificar y distribuir estos programas de ejemplo de la forma que desee, sin pago alguno a IBM, con los fines de desarrollar, utilizar, comercializar o distribuir programas de aplicación de acuerdo con la interfaz de programación de aplicaciones correspondiente a la plataforma operativa para la que están escritos los programas de ejemplo. Estos ejemplos no se han probado exhaustivamente bajo todas las condiciones. Por tanto, IBM no puede asegurar ni implicar la fiabilidad, utilidad o función de estos programas.

Cada copia o porción de estos programas de ejemplo o cualquier trabajo derivado debe incluir una nota de copyright como la siguiente:

© (nombre de la empresa) (año). Partes de este código derivan de programas de ejemplo de IBM Corp. © Copyright IBM Corp. _especifique el año o años_. Reservados todos los derechos.

Marcas registradas

Los términos siguientes, que pueden estar indicados por un asterisco (*), son marcas registradas de International Business Machines Corporation en los Estados Unidos y/o en otros países.

ACF/VTAM	IBM
AISPO	IMS
AIX	IMS/ESA
AIX/6000	LAN DistanceMVS
AIXwindows	MVS/ESA
AnyNet	MVS/XA
APPN	Net.Data
AS/400	OS/2
BookManager	OS/390
CICS	OS/400
C Set++	PowerPC
C/370	QBIC
DATABASE 2	QMF
DataHub	RACF
DataJoiner	RISC System/6000
DataPropagator	RS/6000
DataRefresher	S/370
DB2	SP
DB2 Connect	SQL/DS
DB2 Extenders	SQL/400
DB2 OLAP Server	System/370
DB2 Universal Database	System/390
Distributed Relational Database Architecture	SystemView
DRDA	VisualAge
eNetwork	VM/ESA
Extended Services	VSE/ESA
FFST	VTAM
First Failure Support Technology	WebExplorer
	WIN-OS/2

Los términos siguientes son marcas registradas de otras empresas:

Microsoft, Windows y Windows NT son marcas registradas de Microsoft Corporation.

Java, y las marcas registradas y logotipos basados en Java y Solaris, son marcas registradas de Sun Microsystems, Inc. en los Estados Unidos y/o en otros países.

Tivoli y NetView son marcas registradas de Tivoli Systems Inc. en los Estados Unidos y/o en otros países.

UNIX es una marca registrada en los Estados Unidos y/o en otros países bajo licencia exclusiva de X/Open Company Limited.

Otros nombres de empresas, productos o servicios, que pueden estar indicados por un doble asterisco (**), pueden ser marcas registradas o marcas de servicio de otras empresas.

Glosario

almacenar en antememoria. Proceso de almacenamiento local de resultados utilizados con frecuencia de una petición al servidor Web para su rápida recuperación, hasta que llega el momento de renovar la información.

antememoria. Parte de la memoria o del espacio en disco que contiene datos a los que se ha accedido recientemente, que ha sido diseñada para acelerar el acceso subsiguiente a los mismos datos. La antememoria se utiliza con frecuencia para mantener una copia local de datos utilizados con frecuencia a los que se puede acceder a través de una red.

API. Véase .

applet. Programa Java incluido en una página HTML. Las applets funcionan con navegadores habilitados para Java, por ejemplo Netscape Navigator, y se cargan cuando se procesa la página HTML.

base de datos. Conjunto de tablas o conjunto de espacios de tabla y espacios de índice.

BLOB. Gran objeto binario.

Cache Manager. Programa que gestiona una antememoria para una máquina. Puede gestionar múltiples antememorias.

CGI. Common Gateway Interface (Interfaz de pasarela común).

cliette. Proceso de larga ejecución en Net.Data Live Connection que sirve peticiones del servidor Web. El Connection Manager planifica los procesos de cliette para servir dichas peticiones.

CLOB. Objeto grande de caracteres.

Common Gateway Interface (CGI). Procedimiento estandarizado para que un servidor Web pase el control a un programa de aplicación y reciba datos como respuesta.

Connection Manager. Archivo ejecutable, dtwcm, en Net.Data que es necesario para soportar Live Connection.

control de compromiso. Establecimiento de un límite en el proceso bajo el que se está ejecutando Net.Data donde las operaciones en los recursos forman parte de una unidad de trabajo.

cookie. Paquete de información enviada por un servidor HTTP a un navegador Web y que luego el navegador vuelve a enviar cada vez que accede a dicho servidor. Los cookies pueden contener cualquier información arbitraria que elija el servidor y se utilizan para mantener el estado entre las transacciones HTTP que, de lo contrario, no tendrían ningún estado. *Free Online Dictionary of Computing*

cortafuegos. Sistema con software que protege una red interna del acceso externo no autorizado.

DATALINK. Tipo de datos de DB2 que permite las referencias lógicas de la base de datos a un archivo almacenado fuera de ésta.

DBCLOB. Objeto grande de caracteres de doble byte.

DBMS. Sistema de gestión de bases de datos.

directorio de trabajo actual. Directorio por omisión de un proceso desde el cual se resuelven todos los nombres de vías de acceso relativas.

Domino Go Webserver. Servidor Web ofrecido por Lotus Corp. e IBM, que proporciona conexiones seguras y regulares. GWAPI es la interfaz que se proporciona con este servidor.

entorno de lenguaje. Módulo que proporciona acceso desde una macro de Net.Data a una fuente de datos externa como, por ejemplo, DB2 o a un lenguaje de programación como, por ejemplo, Perl.

GWAPI. API de Go Webserver.

HTML. Hypertext markup language (lenguaje de marcación de hipertexto).

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

hypertext markup language. Lenguaje de códigos utilizado para escribir documentos Web.

hypertext transfer protocol. Protocolo de comunicaciones utilizado en un servidor Web y un navegador.

interfaz de archivo plano. Conjunto de funciones incorporadas de Net.Data que le permiten leer y grabar datos de archivos de texto simple.

interfaz de programación de aplicaciones. Interfaz funcional proporcionada por el sistema operativo o por un programa bajo licencia que se pueda solicitar por separado que permite a un programa de aplicación escrito en un lenguaje de alto nivel utilizar funciones o datos específicos del sistema operativo o del programa bajo licencia. Net.Data soporta las API de servidor Web de propiedad siguientes para obtener un rendimiento mejorado en los procesos CGI: GWAPI, ISAPI y NSAPI.

Internet. Red de sistemas TCP/IP pública internacional.

Intranet. Red TCP/IP dentro del cortafuegos de una compañía.

ISAPI. API de Internet Server de Microsoft.

Java. Lenguaje de programación orientado a objetos independiente del sistema operativo, especialmente útil para las aplicaciones de Internet.

Live Connection. Componente de Net.Data que consta de un Connection Manager y de múltiples cliettes. Live Connection gestiona la reutilización de la base de datos y de las conexiones de máquinas virtuales Java.

LOB. Objeto grande.

middleware. Software que actúa como mediador entre un programa de aplicación y una red. Gestiona la interacción entre un programa de aplicación cliente y un servidor a través de la red.

nombre de vía de acceso. Indica al sistema cómo localizar un objeto. El nombre de vía de acceso se expresa como una secuencia de nombres de directorios seguidos del nombre del objeto. Los directorios individuales y el nombre de objeto están separados por un carácter de barra inclinada (/) o de barra inclinada invertida (\).

nombre de vía de acceso relativa. Nombre de vía de acceso que no empieza en el nivel más alto o directorio "raíz". El sistema supone que el nombre de vía de acceso empieza en el directorio de trabajo actual del proceso.

NSAPI. API de Netscape.

null (nulo). Valor especial que indica la ausencia de información.

Perl. Lenguaje de programación interpretado.

permanencia. Estado de conservación de un valor asignado para una transacción entera, donde una transacción abarca múltiples invocaciones de Net.Data. Sólo las variables pueden ser permanentes. Además, las operaciones en los recursos afectados por el control de compromiso se conservan activos hasta que se realiza de forma explícita un compromiso o una retrotracción o hasta que se completa la transacción.

puerto. Número de 16 bits utilizado para comunicarse entre TCP/IP y una aplicación o un protocolo de nivel superior.

registro. Depósito donde se pueden almacenar y recuperar series.

servidor Web. Software servidor HTTP de ejecución de sistema, por ejemplo Internet Connection.

sistema de gestión de bases de datos (database management system - DBMS). Sistema de software que controla la creación, la organización y la modificación de una base de datos y el acceso a los datos almacenados en ella.

TCP/IP. Transmission Control Protocol / Internet Protocol (Protocolo de control de transmisión / Protocolo Internet).

tipo de datos. Atributo de columnas y literales.

transacción. Invocación de Net.Data. Si se utiliza Net.Data permanente, una transacción puede abarcar múltiples invocaciones de Net.Data.

Transmission Control Protocol / Internet Protocol. Conjunto de protocolos de comunicaciones que soportan funciones de conectividad de igual a igual para redes locales y redes de área amplia.

unidad de trabajo. Secuencia recuperable de operaciones que se tratan como una operación atómica. Todas las operaciones dentro de la unidad de trabajo se pueden completar (comprometer) o deshacer (retrotraer) como si las operaciones fueran una sola operación. Sólo se pueden comprometer o retrotraer las operaciones efectuadas en recursos que están afectados por el control de compromiso.

uniform resource locator. Dirección que nombra un servidor HTTP y opcionalmente un directorio y nombre de archivo, por ejemplo:
<http://www.ibm.com/software/data/net.data/index.html>.

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

vía de acceso. Ruta de búsqueda utilizada para localizar archivos.

vía de acceso absoluta. Nombre completo de vía de acceso de un objeto. Los nombres de vías de acceso absolutas empiezan en el nivel más alto o directorio "raíz" (que se identifica por el carácter de barra inclinada (/) o de barra inclinada invertida (\)).

Índice

A

acceder a bases de datos ODBC 164
acceder a bases de datos Oracle 164
acceder a DB2 165
acceso, derechos de
 para archivos Net.Data 65
 para entornos de lenguaje 163
administración, herramienta
 cifrar contraseñas de cliente de
 base de datos, clientes 58
configurar Net.Data
 antes de empezar 50
 clientes 54
 puertos de Live
 Connection 53
 sentencias de variables de
 configuración 63
 sentencias de vía de
 acceso 51
 visión general 50
ENVIRONMENT, sentencias 59
instalar bibliotecas de ejecución
 de Java 50
AIX, Apéndice: Net.Data para 257
almacenar en antememoria
 anotación cronológica 223, 235
 aplicaciones de ejemplo 215
 consultar una antememoria
 específica 235
 desechar 235
 detener 235
 determinar configuración 217
 distintivos 235
 interfaces 219
 introducción 216
 mandato cacheadm 235
 planificar 220
 restricciones 219
 reunir estadísticas 235
 terminología 216
 una página 232
ámbito de identificador
 bloque FUNCTION 119
 bloque REPORT 120
 bloque ROW 120
 global 119
 macro 119
ámbito del identificador 119

anotación cronológica de errores
archivo de anotaciones
 cronológicas
 activar 246, 249
 especificar ubicación 18
 formato 248, 252
 tamaño 245, 248
 variable de nivel 18
 variable de ubicación 18
consideraciones acerca del
 rendimiento 241
descripción 245, 248
DTW_LOG_DIR 18, 246
DTW_LOG_LEVEL 64, 247
nivel de anotación cronológica
 atributo de invocación 250
 efecto en el rendimiento 241
 especificar 64, 247, 250
 variable 64, 247
nombres de archivos de Live
 Connection 250
planificar 246, 249
antememoria
 activar la actual 225
 definición 216
 especificar antigüedad de las
 páginas 226
 especificar espacio para
 páginas 225
 especificar memoria para 226
 identificadores 217, 218, 221
 stanza, configurar 225
 vía de acceso 225
Apache Web Server, instalar 42
API del servidor Web
 consideraciones acerca del
 rendimiento 209
 mejorar el rendimiento con 209
applets Java, entorno de lenguaje
 entorno de lenguaje 187
archivo de anotaciones cronológicas
 activar 18, 246, 249
 Cache Manager 222, 223, 224
 controlar nivel 246, 249
 formato 248, 252
 Live Connection, nombres 250
 para cada antememoria 229
 rotación 247, 251

archivo de anotaciones cronológicas
 (*continuación*)
 tamaño máximo 245, 247, 248,
 251
archivo de anotaciones cronológicas
 de transacciones de
 antememoria 229
archivo de inicialización
 ENVIRONMENT, sentencias 28
archivo plano, entorno de lenguaje
 de interfaz de
 visión general 182
archivo plano, funciones de 144
archivos
 especificar derechos de acceso a
 Net.Data 65
 subir 23, 84
autenticación, seguridad 70
autorización
 especificar derechos de acceso a
 archivos Net.Data 65
 seguridad 71

B

base de datos
 clientes, configurar 54
beans
 configurar para Net.Data 46
bibliotecas compartidas
 cargar para entornos de lenguaje
 en AIX 257
blanco adicional, variable para
 eliminar espacio en 19
BLOB 168
bloques, macro 110
bloques IF 154

C

Cache Manager
 archivo de anotaciones
 cronológicas
 activar 223
 denominar 222
 distintivos de rastreo 224
 para cada antememoria 229
 archivo de configuración 8, 217,
 222
 definición 217
 definir 222
 definir una antememoria 225

- Cache Manager (*continuación*)
 - detener 231
 - iniciar 230
 - puerto 222
 - stanza, configurar 222
 - tiempo de espera de conexión 223
 - variables de configuración 15
- cacheadm
 - detener el Cache Manager 231
 - sintaxis 235
- caracteres, juegos de 18, 20
- cifrado
 - contraseñas de cliette de base de datos 58
- cifrado de red 70
- cliettes
 - configurar con la herramienta de administración
 - añadir 55
 - cifrar contraseñas de base de datos 58
 - información de base de datos 57
 - modificar 56
 - probar inicio de sesión de base de datos DB2 58
 - suprimir 58
 - descripción 211
 - entorno de lenguaje Java 197
 - nombres de archivos ejecutables 38
- CLOB 168
- codificar URL de DataLink en conjuntos de resultados 178
- COMMIT 166
- condicional
 - lógica, bloques IF 154
- condicionales
 - variables 125
- condiciones de error, entornos de lenguaje 162
- configurar Cache Manager 222, 225
- configurar Net.Data
 - archivo de configuración de Cache Manager
 - descripción 8
 - puertos 15
 - stanzas 222, 225
 - archivo de configuración de Live Connection 37
 - actualizar 36
 - descripción 8
 - archivo de inicialización
 - actualizar 12
- configurar Net.Data (*continuación*)
 - archivo de inicialización (*continuación*)
 - descripción 7
 - ENVIRONMENT, sentencias 28
 - sentencias de variables de configuración 13
 - sentencias de vía de acceso 22
 - comparación de archivos de control 9
 - comparación de métodos 5
 - configurar entornos de lenguaje 31
 - derechos de acceso a archivos Net.Data 65
 - FastCGI 42
 - herramienta de administración
 - antes de empezar 50
 - cliettes 54
 - configurar sentencias de variables 63
 - ENVIRONMENT, sentencias 59
 - instalar bibliotecas de ejecución de Java 50
 - puertos 53
 - sentencias de vía de acceso 51
 - visión general 50
 - manual frente a herramienta de administración 5
 - para uso con beans Java 46
 - para uso con las API del servidor Web 46
 - para uso con servlets Java 46
 - visión general 5
- conjuntos de resultados
 - múltiples
 - directrices y restricciones 152
 - informes por omisión 176
 - procesar, procedimientos almacenados 174
 - uno solo 175
- Connection Manager
 - activar anotación cronológica de Live Connection 249
 - descripción 211
 - iniciar
 - AIX 213
 - con la opción de mensajes 213
 - OS/2 y Windows NT 213
- contraseña e inicio de sesión, configurar cliettes 39
- cortafuegos 67
- D**
 - daemon dtwclean, gestionar LOB temporales 171
 - DATALINK, tipo de datos
 - codificación de URL 179
 - DataLink File Manager 178
 - DB2INSTANCE 16
 - DBCLOB 168
 - DBCS 259
 - declaración de estructura de macro, parte 107
 - DEFINE, bloque
 - definir variables 120
 - descripción 110
 - definir variables
 - códigos de formulario HTML SELECT, INPUT y TEXTAREA 121
 - datos de serie de consulta 122
 - sentencia o bloque DEFINE 120
 - dinámicamente, generar nombres de variable 122
 - dir_inst 50
 - directorio inicial
 - configurar con la herramienta de administración 64
 - configurar en el archivo de inicialización 18, 50
 - distintivos de rastreo, para Cache Manager 224
 - diversas, variables 130
 - Domino Go Webserver (GWAPI)
 - configurar para Net.Data 47
 - Domino Go Webserver, instalar 42
 - DTW_APPLET 187
 - DTW_CACHE_HOST 15
 - DTW_CACHE_PAGE 232
 - DTW_CACHE_PORT 15
 - DTW_CM_PORT 16
 - DTW_DEFAULT_ERROR_MESSAGE 17
 - DTW_DEFAULT_REPORT 149
 - DTW_DIRECT_REQUEST 17
 - DTW_FFI 182
 - DTW_INST_DIR 18, 64
 - DTW_JAVAPPS 195
 - DTW_LOG_DIR 18
 - DTW_LOG_LEVEL 18, 64, 241, 247
 - DTW_MBMODE 18, 259
 - DTW_ODBC 164
 - DTW_ORA 164
 - DTW_PERL 198
 - DTW_REMOVE_WS 19

- DTW_REXX 201
- DTW_SHOWSQL 19
- DTW_SMTP_SERVER 20
- DTW_SQL 165
- DTW_SYSTEM 205
- DTW_UNICODE 20, 258
- DTW_UPLOAD_DIR 23, 84
- DTW_VARIABLE_SCOPE 22
- DTW_WEBREG 184
- dtwcm, mandato 213
- E**
- ejecutables, variables 126
- ejecutar mandatos 205
- ejecutar sentencias de SQL 164, 165
- ejemplo, macro 272
- enlaces
 - en páginas Web para invocar Net.Data 83
 - invocar Net.Data 82, 91
- entorno, variables de 125
- entorno de lenguaje REXX
 - llamar a programas 202
 - pasar parámetros 203
 - rendimiento en AIX 204
 - visión general 201
- entorno de lenguaje System
 - emitir mandatos 205
 - llamar a programas 205
 - pasar parámetros 206
 - visión general 205
- entorno nacional 258
- entornos de lenguaje
 - aplicaciones Java 195
 - applet Java 187
 - configurar 31
 - configurar con la herramienta de administración
 - añadir 60
 - modificar 61
 - suprimir 63
 - configurar en el archivo de inicialización 28
 - configurar sentencias ENVIRONMENT 28, 59
 - ejemplos 28
 - interfaz de archivo plano 182
 - llamar 162
 - manejar condiciones de error 162
 - ODBC 164
 - Oracle 164
 - Perl 198
 - registro Web 184
 - seguridad 163
 - soportados 161

- entornos de lenguaje (*continuación*)
 - SQL 165
- entornos de lenguaje de datos 163
- ENVIRONMENT, sentencias
 - configurar en el archivo de inicialización 28
 - descripción 28, 59
 - DLL o nombre de biblioteca 29
 - ejemplo 30
 - lista de parámetros 29
 - nombre de cliette 29
 - sintaxis 28
 - tipo de entorno de lenguaje 29
- espacios en blanco adicionales, variable para eliminar 19
- EXEC_PATH 23, 51

- F**
- FastCGI
 - configurar Net.Data 42
 - configurar para Net.Data
 - instalar Apache Web Server 42
 - instalar Domino Go Webserver 42
 - consideraciones acerca del rendimiento 210
 - determinar procesos simultáneos 210
 - entornos de lenguaje soportados 42, 210
- FFI, entorno de lenguaje
 - llamar a funciones incorporadas 183
- FFI_PATH 24, 51
- formatear salida de datos 147
- formularios
 - en páginas Web para invocar Net.Data 83
 - invocar Net.Data 82, 91
 - utilizando el tipo de entrada FILE 84
- función, llamadas de incorporada 140
- sintaxis 140
- funciones
 - archivo plano 144
 - definidas por el usuario 134
 - definir 134
 - descripción 133
 - FUNCTION, sintaxis de bloque 134
 - llamar 140
 - llamar procedimientos almacenados 171
 - matemáticas 143

- funciones (*continuación*)
 - palabra 143
 - registro Web 144
 - serie 143
 - sintaxis de bloque
 - MACRO_FUNCTION 134
 - tabla 144
 - uso general 142
- FUNCTION, bloque
 - ámbito del identificador 119
 - descripción 111
 - formatear salida 147
 - llamar funciones 140
- FunctionServlet
 - descripción 96
 - ejecutar 99
 - plug-in NOF 267
- G**
- generar applets Java 187
- gestión de conexiones
 - configurar 36
 - rendimiento 210
- gestionar LOB temporales 171
- global, ámbito de identificador 119
- glosario 281
- GWAPI
 - configurar para Net.Data 47
 - invocando Net.Data 93
- H**
- habilitación de petición directa (DTW_DIRECT_REQUEST) 17
- hacer referencia a variables 122
- HTML
 - bloques
 - descripción 112
 - ejemplo 145
 - invocar Net.Data 145
 - procesar 146
 - códigos para tablas 148
 - datos no reconocidos como 146
 - enlaces
 - acerca de 83
 - invocar Net.Data 82, 91
 - FORM Submit, botón 146
 - formularios
 - acerca de 83
 - códigos SELECT, INPUT y TEXTAREA, definir variables 121
 - invocar Net.Data 82, 91
 - generar en una macro 145
 - URL, invocar Net.Data 91
- HTML_PATH 25, 51

- I**
- ID de antememoria
 - definición 217, 218
 - planificar 221
- impresión, inhabilitar para informes por omisión 148
- IMS Web, entorno de lenguaje
 - configurar 31
- INCLUDE_PATH 25, 51
- información de cabecera, bloque
 - REPORT 148
- información pie de página, bloque
 - REPORT 148
- informe, personalizar formatos de 148
- informe, variables 132
- informes
 - generar múltiples con una llamada de función 149
 - por omisión 149
- informes por omisión
 - especificar para procedimientos almacenados 175, 176
 - imprimir 148
- inicialización, archivo
 - actualizar 12
 - descripción 7
 - ejemplo 11
 - formato 13
 - sentencias de variables de configuración 13
 - sentencias de vía de acceso 22
- iniciar Net.Data 79
- inicio de sesión y contraseña, configurar cliettes 39
- instalación, variable de configuración de directorio
 - configurar con la herramienta de administración 64
 - configurar en el archivo de inicialización 18
- invocar applets 187
- invocar Net.Data
 - bloques HTML 145
 - con una macro 81
 - enlaces 82, 91
 - FastCGI 46
 - formularios 82, 91
 - GWAPI 93
 - ISAPI 93
 - NSAPI 94
 - petición de macro 79
 - petición directa 79
 - sin ninguna macro 86
 - sintaxis 80

- invocar Net.Data (*continuación*)
 - URL 82, 91
 - utilizando CGI 79
 - utilizando las API de servidor
 - Web 92
 - visión general 79
- ISAPI
 - configurar para Net.Data 47
 - invocando Net.Data 93

J

- Java, applets
 - clases 193
 - crear 187
 - generar códigos 187
 - invocar 187
- Java, beans
 - configurar para Net.Data 46
- Java, configurar cliettes 39
- Java, entorno de lenguaje
 - crear cliettes 197
 - crear funciones 196
 - estructura de archivos 196
 - invocar 198
 - llamar funciones 195
- Java, entorno de lenguaje de aplicaciones
 - visión general 195
- Java Application, entorno de lenguaje
 - configurar 32
- Java servlets
 - configurar para Net.Data 46
- juego de caracteres de doble byte 259

L

- lenguaje, entornos
 - cargar bibliotecas compartidas en
 - AIX 257
 - REXX 201
 - System 205
 - variables 133
- lista, variables 128
- Live Connection
 - archivo de configuración
 - actualizar 36
 - cliettes de base de datos 38
 - cliettes Java 39
 - descripción 8
 - ejemplo 12
 - formato 36
 - inicio de sesión y contraseña 39
 - nombre 37
 - nombre de base de datos 39

- Live Connection (*continuación*)
 - archivo de configuración (*continuación*)
 - número de procesos 38, 39
 - tipo de proceso 38
 - cliettes
 - archivos de configuración 9
 - configurar con la herramienta de administración 54
 - determinar si se debe utilizar 212
 - flujo de proceso 214
 - iniciar Connection Manager 212
 - mejorar el rendimiento con 210
 - puertos
 - configurar con la herramienta de administración 53
 - configurar en el archivo de inicialización 37
 - ventajas 212
- Live Connection, anotación cronológica
 - activar 249
 - archivo de anotaciones cronológicas
 - formato 252
 - tamaño 248
 - controlar nivel 249
 - descripción 248
 - nivel de anotación cronológica
 - atributo de invocación 250
 - especificar 250
 - nombres de archivos 250
 - planificar 249
- LOB 168

LL

- llamar
 - aplicaciones Java 195
 - entornos de lenguaje 162
 - funciones 140
 - funciones incorporadas de Web Registry 186
 - funciones incorporadas FFI 183
 - procedimientos
 - almacenados 171, 173
 - programas, System 205
 - programas REXX 201, 202
 - scripts Perl 198

M

- macro, petición de
 - descripción 79
 - ejemplos 81
 - sintaxis 81

- MACRO_FUNCTION, bloque
 - llamar funciones 140
 - sintaxis 134
- MACRO_PATH 27, 51
- macros
 - ámbito del identificador 119
 - anatomía 108
 - bloque DEFINE 110
 - bloque FUNCTION 111
 - bloque HTML 112
 - bloques 110
 - bloques IF 154
 - bloques WHILE 156
 - desarrollar 107
 - descripción 1
 - ejemplo 10, 109
 - funciones 133
 - generar HTML 145
 - lógica condicional 154
 - navegación dentro y entre 112
 - parte de declaración 107
 - parte de presentación 107
 - plug-in NOF 267
 - repetir en bucle 156
 - variables 118
- MacroServlet
 - descripción 96
 - ejecutar 97
 - plug-in NOF 267
- matemáticas, funciones 143
- MAX_PROCESS 38, 39, 57
- MBCS para funciones, soporte 18
- mejorar el rendimiento 209
- MESSAGE, bloque
 - ámbito 138
 - descripción 138
 - ejemplo 139
 - procesar 138
 - sintaxis 138
- MIN_PROCESS 38, 39, 57
- múltiples bloques de informe 149

N

- navegación, dentro y entre
 - macros 112
- Net.Data
 - archivos, derechos de acceso 65
 - configurar 5
 - invocar 79
 - macros, desarrollar 107
 - mecanismos de seguridad 71
 - visión general 1
- Net.Data, macros. Véase macros. 1
- Net.Data, servlets
 - FunctionServlet 96
 - MacroServlet 96

- Net.Data, servlets (*continuación*)
 - plug-in NOF
 - configurar 268
 - descripción 267
 - modificar propiedades 272
 - publicar servlets 272
- NetObjects Fusion (NOF), plug-in
 - configurar 268
 - descripción 267
 - instalar 268
 - modificar propiedades de
 - servlet 272
 - para servlets de función y
 - macro 267
 - publicar 272
 - requisitos de hardware y
 - software 268
- NOF (NetObjects Fusion),
 - plug-in 267
- NSAPI
 - configurar para Net.Data 48
 - invocando Net.Data 94

O

- objetos grandes (LOB)
 - descripción 168
 - formatos válidos 169
 - temporales, gestionar 171
 - tipos soportados 168
- ocultas, variables
 - esconder nombres de
 - variables 127
- ODBC, entorno de lenguaje
 - restricciones 164
 - variables 164
 - visión general 164
- Oracle, entorno de lenguaje
 - configurar 33
 - restricciones 164
 - visión general 164

P

- página de códigos 258
- páginas Web, almacenar en
 - antememoria 232
- palabra, funciones de 143
- partes de una macro
 - declaración 107
 - presentación 107
- pasar parámetros
 - entorno de lenguaje System 206
- procedimientos
 - almacenados 174
 - programas REXX 203
 - scripts Perl 199

- Perl, entorno de lenguaje
 - bloques REPORT y
 - MESSAGE 200
 - llamar a funciones
 - incorporadas 198
 - pasar parámetros 199
 - visión general 198
- petición directa
 - descripción 79
 - ejemplos 90
 - restricciones de almacenamiento
 - en antememoria 219
 - sintaxis 86
- plano, fuentes de datos de
 - archivo 182
- plug-in NetObjects Fusion 267
- procedimientos almacenados
 - conjuntos de resultados
 - individuales 175
 - informes por omisión 175, 176
 - llamar desde macro 171
 - múltiples conjuntos de
 - resultados 176
 - pasar parámetros 174
 - pasos 173
 - procesar conjuntos de
 - resultados 174
 - REPORT, bloques 175, 177
 - tablas Net.Data 176, 177
 - tipos de datos válidos 172
- procesar conjuntos de resultados,
 - procedimientos almacenados 174
- proteger activos 67
- publicar servlets con plug-in
 - NOF 272
- puertos
 - Cache Manager 15, 222
 - Live Connection
 - archivo de configuración 37
 - configurar con la herramienta
 - de administración 53

R

- registro Web, funciones de 144
- registros 184
- relacionales, entornos de lenguaje de
 - bases de datos 163
- rendimiento
 - anotación cronológica de
 - errores 241
 - API del servidor Web 209
 - cache query all 237
 - entorno de lenguaje Perl 244
 - entorno de lenguaje REXX 241
 - entorno de lenguaje SQL 242

- rendimiento (*continuación*)
 - entorno de lenguaje System 244
 - entorno REXX 204, 258
 - FastCGI 210
 - Live Connection 210
 - optimizar entornos de lenguaje 241
- repetición en bucle, bloques
 - WHILE 156
- REPORT, bloque
 - procedimientos almacenados 175
- REPORT, bloques
 - ámbito 120
 - descripción 147
 - directrices para múltiples 152
 - ejemplos 150
 - formatear salida de datos 147
 - información de cabecera y pie de página 148
 - informes por omisión 149
 - múltiples 149
 - procedimientos almacenados 177
 - restricciones 152
- REPORT y MESSAGE, bloques
 - scripts Perl 200
- RETURN_CODE, variable 138, 162
- REXX, mejorar el rendimiento 258
- ROW, ámbito de identificador de bloque 120

S

- seguridad
 - almacenar en antememoria 219
 - autenticación 70
 - autorización 71
 - cifrado de red 70
 - cifrar contraseñas de cliente de base de datos 58
 - cortafuegos 67
 - entornos de lenguaje 163
 - especificar derechos de acceso 65, 163
 - mecanismos de Net.Data 71
 - visión general 67
- sentencias de variables de configuración
 - configurar
 - con la herramienta de administración 63
 - configurar en el archivo de inicialización 13
 - DB2INSTANCE 16
 - descripción 13
 - directorio inicial (dir_inst) 18

- sentencias de variables de configuración (*continuación*)
 - DTW_CACHE_HOST 15
 - DTW_CACHE_PORT 15
 - DTW_CM_PORT 16
 - DTW_DEFAULT_ERROR_MESSAGE 17
 - DTW_DIRECT_REQUEST 17
 - DTW_INST_DIR 18, 64
 - DTW_LOG_DIR 18
 - DTW_LOG_LEVEL 18, 64
 - DTW_MBMODE 18
 - DTW_REMOVE_WS 19
 - DTW_SHOWSQL 19
 - DTW_SMTP_SERVER 20
 - DTW_UNICODE 20
 - DTW_VARIABLE_SCOPE 22

- sentencias de vía de acceso
 - configurar con la herramienta de administración
 - añadir 52
 - modificar 52
 - suprimir 53
 - configurar en el archivo de inicialización 22
 - directrices de actualización 22
 - DTW_UPLOAD_DIR 23
 - EXEC_PATH 23
 - FFI_PATH 24
 - HTML_PATH 25
 - INCLUDE_PATH 25
 - MACRO_PATH 27
 - proteger activos 71

- serie, funciones de 143

- servlets

- configurar para Net.Data 46
- descripción 95
- documentación de API 96
- ejecutar 96
- Net.Data
 - configurar plug-in 268
 - función 96
 - macro 96
 - modificar propiedades con plug-in 272
 - NetObjects Fusion, plug-in 267
 - plug-in NOF 267
 - publicar con plug-in NOF 272

- sopORTE de idioma nativo para funciones 18

- SQL, entorno de lenguaje
 - restricciones 165
 - variables 165
 - visión general 165

- SQLCODE 162, 163

- stanza

- antememoria, configurar 225
- Cache Manager, configurar 222
- subir archivos 23, 84

T

- tabla, funciones de 144
- tabla, variables 129
- tabla, variables de proceso de 131
- tablas de Net.Data, procedimientos almacenados 176, 177
- tamaños de símbolo 118
- temporales, gestionar LOB 171
- tiempo de espera de conexión, Cache manager 223
- tipos de datos
 - DATALINK 178
 - LOB 168
 - para procedimientos almacenados 172
- tipos de variables 124
- TRANSACTION_SCOPE 166

U

- Unicode 258
- Unicode, variable
 - con DTW_MBMODE 18, 20
- URL
 - definir variables 122
 - invocar Net.Data 82, 91
- uso general, funciones de 142
- usuario, funciones definidas por el 134
- UTF-8 258
- utilizar las API de servidor Web invocando Net.Data 92

V

- variables
 - ámbito 119
 - condicionales 125
 - configuración, sentencias
 - archivo de inicialización 13
 - DB2, instancia (DB2INSTANCE) 16
 - descripción 13
 - directorio inicial 18, 64
 - editar máscaras (DTW_CM_PORT) 16
 - eliminar espacios en blanco adicionales (DTW_REMOVE_WS) 19
 - errores, nivel de anotación cronológica (DTW_LOG_LEVEL) 18, 64

variables (continuación)

errores, ubicación de
 anotación cronológica de
 (DTW_LOG_DIR) 18
 habilitación de mensajes de
 error por omisión
 (DTW_DEFAULT_ERROR_MESSAGE) 17
 habilitación de petición
 directa
 (DTW_DIRECT_REQUEST) 17
 habilitación de SHOWSQL
 (DTW_SHOWSQL) 19
 herramienta de
 administración 63
 instalación, directorio
 (DTW_INST_DIR) 18, 64
 nombre de máquina de
 antememoria
 (DTW_CACHE_HOST) 15
 puerto de Cache Manager
 (DTW_CACHE_PORT) 15
 servidor SMTP
 (DTW_SMTP_SERVER) 20
 servidor SMTP de correo
 electrónico
 (DTW_SMTP_SERVER) 20
 soporte de idioma nativo
 (DTW_MBMODE) 18
 variable de ámbito de variable
 (DTW_VARIABLE_SCOPE) 22
 variable de Unicode
 (DTW_UNICODE) 20
 definir 120
 descripción 118
 diversas 130
 ejecutables 126
 entorno 125
 entorno de lenguaje 133
 generar nombres
 dinámicamente 122
 hacer referencia 122
 informe 132
 lista 128
 ocultas 127
 proceso de tabla 131
 referencias generadas
 dinámicamente 122
 tabla 129
 tamaños de símbolo 118
 tipos 118, 124
 variables ocultas
 proteger activos 71

W

Web, API de servidor
 configurar para Net.Data
 descripción 46
 GWAPI 47
 ISAPI 47
 NSAPI 48
 consideraciones 92
 descripciones 92
 invocando Net.Data
 GWAPI 93
 ISAPI 93
 NSAPI 94
 Web, servidor
 configurar para API de servidor
 Web 46
 configurar para FastCGI 42
 Web Registry, entorno de lenguaje
 llamar a funciones
 incorporadas 186
 visión general 184
 WHILE, bloques 156

Cómo ponerse en contacto con IBM

Si tiene un problema técnico, repase y lleve a cabo las acciones que se sugieren en la *Guía de resolución de problemas* antes de ponerse en contacto con el Centro de Asistencia al Cliente de DB2. Dicha guía sugiere información que puede reunir para ayudar al Centro de Asistencia a proporcionarle un mejor servicio.

Para obtener información o para solicitar cualquiera de los productos de DB2 Universal Database, consulte a un representante de IBM de una sucursal local o a un concesionario autorizado de IBM.

Si vive en los Estados Unidos, puede llamar a uno de los números siguientes:

- 1-800-237-5511 para obtener soporte técnico
- 1-888-426-4343 para obtener información sobre las opciones de servicio técnico disponibles

Información del producto

Si vive en los Estados Unidos, puede llamar a uno de los números siguientes:

- 1-800-IBM-CALL (1-800-426-2255) o 1-800-3IBM-OS2 (1-800-342-6672) para solicitar productos u obtener información general.
- 1-800-879-2755 para solicitar publicaciones.

<http://www.ibm.com/software/data/>

Las páginas World Wide Web de DB2 ofrecen información actual sobre DB2 referente a novedades, descripciones de productos, planes de formación, etc.

<http://www.ibm.com/software/data/db2/library/>

La biblioteca técnica de servicio y de productos DB2 ofrece acceso a preguntas frecuentemente formuladas (FAQ), arreglos de programa, manuales e información técnica actualizada sobre DB2.

Nota: Puede que esta información sólo esté disponible en inglés.

<http://www.elink.ibm.com/pbl/pbl/>

El sitio Web para el pedido de publicaciones internacionales proporciona información sobre cómo hacer pedidos de manuales.

<http://www.ibm.com/education/certify/>

El Programa de homologación profesional contenido en el sitio Web de IBM proporciona información de prueba de homologación para diversos productos de IBM, incluido DB2.

ftp.software.ibm.com

Conéctese como anónimo. En el directorio /ps/products/db2 encontrará programas de demostración, arreglos de programa, información y herramientas referentes a DB2 y a muchos otros productos.

comp.databases.ibm-db2, bit.listserv.db2-l

En estos foros de discusión de Internet los usuarios pueden explicar sus experiencias con los productos DB2.

En Compuserve: GO IBMDB2

Entre este mandato para acceder a los foros referentes a la familia de productos DB2. Todos los productos DB2 tienen soporte a través de estos foros.

Para conocer cómo ponerse en contacto con IBM desde fuera de los Estados Unidos, consulte el Apéndice A del manual *IBM Software Support Handbook*. Para acceder a este documento, vaya a la página Web siguiente: <http://www.ibm.com/support/> y luego seleccione el enlace "IBM Software Support Handbook", cerca del final de la página.

Nota: En algunos países, los distribuidores autorizados de IBM deben ponerse en contacto con su organización de soporte en lugar de acudir al Centro de Soporte de IBM.



Impreso en España