IBM Net.Data für OS/2, Windows NT und UNIX

Verwaltung und Programmierung

Version 7

IBM Net.Data für OS/2, Windows NT und UNIX

Verwaltung und Programmierung

Version 7

Anmerkung

Vor der Verwendung dieser Informationen und des darin beschriebenen Produkts sollten die allgemeinen Informationen unter „Bemerkungen” auf Seite 269 gelesen werden.

- Die IBM Homepage finden Sie im Internet unter: **ibm.com**
- IBM und das IBM Logo sind eingetragene Marken der International Business Machines Corporation.
- Das e-business Symbol ist eine Marke der International Business Machines Corporation
- Infoprint ist eine eingetragene Marke der IBM.
- ActionMedia, LANDesk, MMX, Pentium und ProShare sind Marken der Intel Corporation in den USA und/oder anderen Ländern.
- C-bus ist eine Marke der Corollary, Inc. in den USA und/oder anderen Ländern.
- Java und alle Java-basierenden Marken und Logos sind Marken der Sun Microsystems, Inc. in den USA und/oder anderen Ländern.
- Microsoft Windows, Windows NT und das Windows-Logo sind Marken der Microsoft Corporation in den USA und/oder anderen Ländern.
- PC Direct ist eine Marke der Ziff Communications Company in den USA und/oder anderen Ländern.
- SET und das SET-Logo sind Marken der SET Secure Electronic Transaction LLC.
- UNIX ist eine eingetragene Marke der Open Group in den USA und/oder anderen Ländern.
- Marken anderer Unternehmen/Hersteller werden anerkannt.

Änderungen in der IBM Terminologie

Die ständige Weiterentwicklung der deutschen Sprache nimmt auch Einfluß auf die IBM Terminologie. Durch die daraus resultierende Umstellung der IBM Terminologie, kann es u. U. vorkommen, dass in diesem Handbuch sowohl alte als auch neue Termini gleichbedeutend verwendet werden. Dies ist der Fall, wenn auf ältere existierende Handbuchausschnitte und/oder Programmteile zurückgegriffen wird.

Diese Ausgabe gilt bis auf weiteres für IBM Net.Data für OS/2, Windows NT und UNIX, einer Funktion der Version 7.2 von DB2 Universal Database und alle nachfolgenden Releases und Änderungen.

Diese Veröffentlichung ist eine Übersetzung des Handbuchs
IBM Net.Data for OS/2, Windows NT, and UNIX Administration and Programming Guide,
herausgegeben von International Business Machines Corporation, USA

Informationen, die nur für bestimmte Länder Gültigkeit haben und für Deutschland, Österreich und die Schweiz nicht zutreffen, wurden in dieser Veröffentlichung im Originaltext übernommen.

Möglicherweise sind nicht alle in dieser Übersetzung aufgeführten Produkte in Deutschland angekündigt und verfügbar; vor Entscheidungen empfiehlt sich der Kontakt mit der zuständigen IBM Geschäftsstelle.

Änderung des Textes bleibt vorbehalten.

Herausgegeben von:
SW TSC Germany
Kst. 2877
Juni 2001

Inhaltsverzeichnis

Vorwort

Vielen Dank, dass Sie sich für Net.Data, das IBM Entwicklungs-Tool zum Erstellen dynamischer Webseiten, entschieden haben! Mit Hilfe von Net.Data können Sie schnell Webseiten mit dynamischem Inhalt entwickeln, indem Sie Daten aus einer Vielzahl von Datenquellen integrieren und die Leistungsstärke der Ihnen bereits bekannten Programmiersprachen ausschöpfen.

Informationen zu Net.Data

Mit Net.Data können Sie unter Verwendung von Daten aus Verwaltungssystemen für relationale und nichtrelationale Datenbanken (DBMS - Database Management Systems), einschließlich DB2-, IMS- und ODBC-fähiger Datenbanken sowie unter Verwendung von Anwendungen, die in Programmiersprachen wie Java, JavaScript, Perl, C, C++ und REXX geschrieben wurden, dynamische Webseiten erstellen.

Net.Data ist ein Makroumwandler, der als Middleware auf einer Webserver-Maschine ausgeführt wird. Sie können Net.Data-Anwendungsprogramme (sogenannte *Makros*) schreiben, die von Net.Data interpretiert und für die Erstellung dynamischer Webseiten mit angepasstem Inhalt verwendet werden. Grundlage hierfür sind die Eingabe vom Benutzer, der aktuelle Status Ihrer Datenbanken, andere Datenquellen, vorhandene Geschäftslogik und andere Faktoren, die Sie in den Makroentwurf aufnehmen.

Eine Anforderung in Form einer URL-Adresse (URL - Uniform Resource Locator) wird von einem Browser, wie Netscape Navigator oder Internet Explorer, an einen Webserver gesendet, der die Anforderung zur Ausführung an Net-.Data weiterleitet. Net.Data lokalisiert das Makro, führt es aus und erstellt eine Webseite, die basierend auf den von Ihnen definierten Funktionen angepasst wird. Diese Funktionen können folgende Aktionen ausführen:

- Einbinden von Geschäftslogik in Perl-Scripts, C- und C++- oder REXX-Programme
- Zugreifen auf Datenbanken wie DB2
- Zugreifen auf andere Datenquellen wie unstrukturierte Textdateien

Net.Data gibt diese Webseite an den Webserver weiter, der die Seite seinerseits über das Netzwerk zur Anzeige im Browser weiterleitet.

Net.Data kann in Server-Umgebungen verwendet werden, die zur Verwendung von Schnittstellen wie HTTP (HyperText Transfer Protocol) und CGI (Common Gateway Interface) konfiguriert sind. HTTP ist eine dem Industrie-

standard entsprechende Schnittstelle für die Interaktion zwischen einem Browser und einem Webserver, und CGI ist eine dem Industriestandard entsprechende Schnittstelle für den Webserver-Aufruf von Gateway-Anwendungen wie Net.Data. Net.Data unterstützt außerdem eine Vielzahl von Webserver-APIs (Application Programming Interfaces - Anwendungsprogrammierschnittstellen) für verbesserte Leistung. Die Net.Data-Produktfamilie bietet unter OS/400, OS/390, Windows NT, AIX, OS/2, HP-UX, Sun Solaris, Linux und Dynix/PTX ein ähnliches Leistungsspektrum. Net.Data unterstützt außerdem FastCGI und die wesentlichen Webserver-APIs für viele Betriebssysteme.

Ein grafisches Verwaltungs-Tool hilft Ihnen bei der Verwaltung von Net.Data-Konfigurationseinstellungen für die Betriebssysteme AIX, Windows NT und OS/2. Das Verwaltungs-Tool unterstützt Sie auch bei der Angabe von Sicherheitseinstellungen für Ihre Verbindungen zu Datenbanken, die Direktverbindung verwenden.

Für einfachen Zugriff auf Daten von Ihrer Datenbank stellt Net.Data eine Vielzahl von Tools bereit, darunter Plug-Ins für NetObjects Fusion und Assistenten für auf Java basierende Entwicklungen. Diese Tools arbeiten mit den Net.Data-Java-Servlets in der Java-Umgebung und ermöglichen Ihnen das Erstellen von Anwendungen, die betriebssystemübergreifend übertragbar sind. Plug-Ins für NetObjects Fusion ermöglichen Ihnen die Verwendung des Webentwicklungs-Tools von NetObjects Fusion zum Erstellen komplexer Anwendungen mit dynamischen Daten relationaler Datenquellen. Net.Data-Assistenten stellen ein grafisches Tool bereit, das Sie durch das Erstellen grundlegender Net.Data-Makros führt.

Neue Funktionen in Version 7.2

Net.Data Version 7.2 bietet die vollständige Funktionalität früherer Net.Data-Releases und noch vieles mehr! Net.Data für OS/2, Windows NT und UNIX bietet in Version 7.2 die folgenden zusätzlichen Funktionen:

- Die Möglichkeit, SQL-Funktionen aus den Blöcken REPORT und ROW anderer SQL-Funktionen aufzurufen, wie z. B. verschachtelten Funktionsaufrufen bei der Verwendung der Direktverbindung
- Verbesserte Leistung durch Ausnutzen der Vorteile der DB2-Cache-Anweisung. Sie können jetzt Parametermarken in die SQL-Anweisungen Ihres Makros stellen, was Ihnen eine effektive Verwendung des Anwendungs-Cache ermöglicht.
- Neue integrierte Net.Data-Funktionen: DTWF_COPY(), DTWF_EXISTS(), DTWF_WRITEFILE()
- Die Möglichkeit, Java-Funktionen ohne die Direktverbindung unter AIX, Sun, Windows NT und Linux direkt aufzurufen

- Unterstützung für Linux S/390
- Unterstützung für gespeicherte Oracle-Prozeduren
- Unterstützung für Net.Data-Trace

Informationen zu diesem Handbuch

In diesem Handbuch werden Verwaltungs- und Programmierungskonzepte für Net.Data sowie das Konfigurieren von Net.Data und seiner Komponenten, das Planen der zu verwendenden Sicherheitsmaßnahmen sowie Maßnahmen zur Steigerung der Systemleistung erläutert.

Auf Ihrer Kenntnis von Programmiersprachen und Datenbanken aufbauend lernen Sie die Verwendung der Net.Data-Makrosprache, d. h. von Java-Servlets, zum Entwickeln von Makros. Sie lernen die Verwendung der von Net.Data bereitgestellten Sprachumgebungen, die auf DB2-Datenbanken und IMS-Transaktionen zugreifen, und Sie werden in die Verwendung von Java, REXX, Perl und anderen Programmiersprachen zum Zugriff auf Ihre Daten eingeführt.

In diesem Handbuch wird möglicherweise auf angekündigte, jedoch noch nicht allgemein verfügbare Produkte und Funktionen verwiesen.

Weitere Informationen wie Net.Data-Beispielmakros, Demos und die neuste Version dieses Handbuchs können über folgende World Wide Web-Site abgerufen werden:

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

Zielgruppe

Dieses Handbuch richtet sich an Planer und Programmierer von Net.Data-Anwendungen. Als Grundlage zum Verständnis der in diesem Handbuch erläuterten Konzepte müssen Sie damit vertraut sein, wie ein Webserver funktioniert, einfache SQL-Anweisungen verstehen und HTML-Befehle, einschließlich HTML-Formularbefehle, kennen.

Die Makrosprache, Variablen und integrierten Funktionen in Net.Data sowie die Unterschiede zwischen den einzelnen Betriebssystemen werden im Handbuch *Net.Data Reference* beschrieben.

Informationen zu Beispielen in diesem Handbuch

Die in diesem Handbuch verwendeten Beispiele sind möglichst einfach gehalten, um bestimmte Konzepte darzustellen. Sie zeigen nicht jede Möglichkeit für den Einsatz von Net.Data-Konstrukten. Einige Beispiele zeigen nur Ausschnitte, bei denen für eine Ausführung zusätzlicher Code erforderlich ist.

Senden von Kommentaren

Wir sind auf Ihre Rückmeldung angewiesen, um möglichst genaue und hochwertige Informationen bereitstellen zu können. Bitte setzen Sie sich deshalb mit uns in Verbindung, wenn Sie Anregungen oder Kommentare zu diesem Handbuch oder zu einem anderen DB2-Dokument haben. Nutzen Sie dazu eine der folgenden Möglichkeiten:

- Senden Sie Ihre Kommentare per E-Mail und unter Angabe des Produkts, der Versionsnummer des Produkts und der Teilenummer der Veröffentlichung an db2pubs@vnet.ibm.com. Wenn Sie sich auf einen bestimmten Textabschnitt beziehen, geben Sie die Textposition an (z. B. Kapitel- und Abschnittüberschrift, Seitennummer oder Titel eines Hilfethemas). Kommentare an obige Adresse müssen auf Englisch formuliert werden.
- Senden Sie Ihren Kommentar über das Web. Unsere Website finden Sie unter folgender Internet-Adresse:

  http://www.ibm.com/software/db2os390

  Die Website verfügt über eine spezielle Seite für Rückmeldungen, über die Sie Ihren Kommentar senden können. (Kommentare müssen auf Englisch formuliert werden.)
- Füllen Sie das Antwortschreiben am Ende dieses Handbuchs aus, und senden Sie es per Post oder Fax an IBM, oder nehmen Sie Kontakt mit Ihrem IBM Ansprechpartner auf.
- E-Mail — Drucken und verwenden Sie die Antwortkarte am Ende dieses Handbuchs. Wählen Sie zum Drucken des Formulars **Drucken** oder **Kopieren** aus dem Pulldown-Menü **Services** aus. Geben Sie dabei *COMMENTS* als Thema an. Senden Sie das ausgefüllte Formular an:

  IBM Corporation, Department W92/H3
  P.O. Box 49023
  San Jose, CA 95161-9023
  U.S.A.

  Sie können das Formular auch Ihrer örtlichen IBM Zweigstelle oder Ihrem IBM Ansprechpartner übergeben, damit diese es dann in die USA schicken.
- Fax — Drucken Sie die Antwortkarte hinten im Handbuch aus, und faxen Sie sie an folgende Nummer: +1-408 463-4393. Befolgen Sie zum Drucken der Antwortkarte die Anweisungen unter "E-Mail".

Kapitel 1. Einführung

Net.Data ist eine Script-basierte Sprache, die auf dem Server ausgeführt wird und Webserver erweitert, indem die dynamische Generierung von Webseiten unter Verwendung der Daten aus einer Vielzahl an Datenquellen aktiviert wird. Die Datenquellen können relationale und nichtrelationale Datenbankverwaltungssysteme wie z. B. DB2, DRDA-fähige Datenbanken und unstrukturierte Dateien enthalten. Sie können Anwendungen sehr schnell erstellen, indem Sie die einfache und doch leistungsstarke Script-Sprache von Net.Data verwenden. Net.Data ermöglicht die Wiederverwendung von vorhandener Geschäftslogik, indem Aufrufe an Anwendungen unterstützt werden, die in einer Vielzahl an Programmiersprachen geschrieben sind, einschließlich Java, C/C++, REXX und so weiter.

In diesem Kapitel wird Net.Data beschrieben, und die Gründe zur Verwendung für Ihre Webanwendung werden genannt.

Net.Data - Produktbeschreibung

Mit Hilfe von Net.Data-Makros können Sie Programmierungslogik ausführen, auf Variablen zugreifen und sie bearbeiten, Funktionen aufrufen und Tools zum Generieren von Berichten verwenden. Ein Makro ist eine Textdatei mit Net.Data-Sprachkonstrukten, die verwendet werden, um eine Anwendung zu erstellen, die HTML-, XML-, JavaScript-Anweisungen und Anweisungen einer Sprachumgebung wie z. B. SQL und Perl enthalten kann. Net.Data verarbeitet das Makro, um eine Ausgabe, die von einem Web-Browser angezeigt werden kann, zu erzeugen. Makros kombinieren die Einfachheit von HTML mit der dynamischen Funktionalität von Webserver-Programmen, wodurch das Hinzufügen von dynamischen Daten zu statischen Webseiten erheblich vereinfacht wird. Die dynamischen Daten können aus lokalen bzw. fernen Datenbanken und aus unstrukturierten Textdateien extrahiert oder durch Anwendungen und Systemservices generiert werden.

Abb. 1 auf Seite 2 illustriert die Beziehung zwischen Net.Data, dem Webserver und unterstützten Daten sowie Sprachumgebungen für Programmiersprachen.

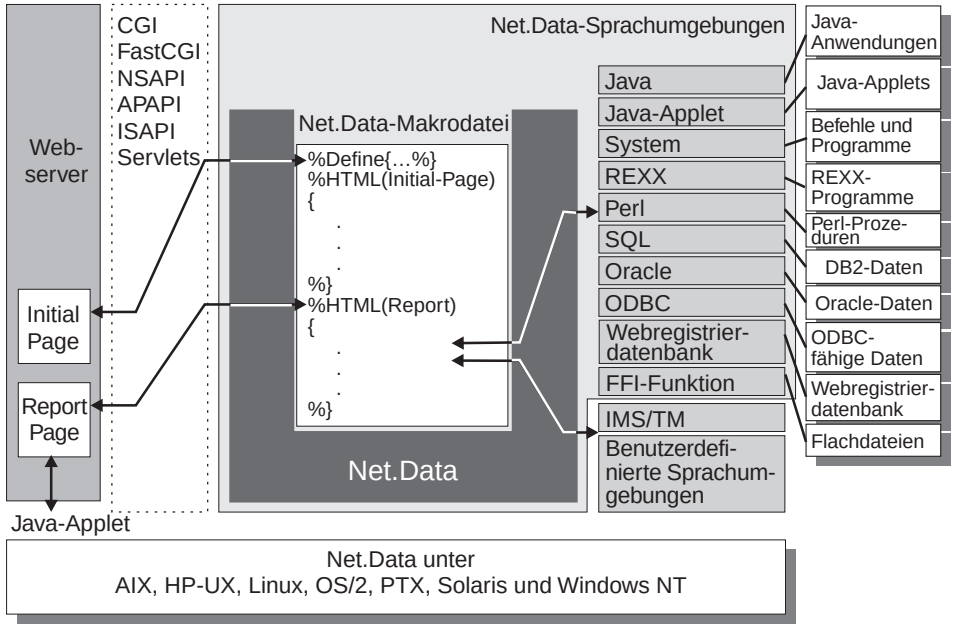


Abbildung 1. Die Beziehung zwischen Net.Data, dem Webserver und unterstützten Daten sowie Programmquellen

Der Webserver ruft Net.Data beim Empfang einer URL-Adresse, die Net.Data-Services anfordert, als eine CGI-, FastCGI- oder Webserver-API auf, indem Net.Data als DLL oder gemeinsam benutzte Bibliothek gestartet wird. Die URL-Adresse enthält Net.Data-spezifische Informationen, und zwar entweder das zu verarbeitende Makro oder die SQL-Anweisung bzw. das direkt aufzurufende Programm. Wenn Net.Data die Verarbeitung der Anforderung beendet hat, wird die entstandene Webseite an den Webserver gesendet. Der Server übergibt diese an den Web-Client, auf dem sie mit dem Browser angezeigt wird.

Gründe zur Verwendung von Net.Data

Net.Data ist eine gute Wahl für das Erstellen dynamischer Webseiten, weil die Verwendung der Makrosprache einfacher ist als das Schreiben eigener Webserver-Anwendungen und weil Net.Data den Einsatz von Sprachen ermöglicht, die Ihnen bereits bekannt sind, wie HTML, SQL, Perl, REXX und JavaScript. Net.Data stellt zudem Sprachumgebungen (Language Environments) bereit, die auf DB2-Datenbanken zugreifen, IMS-Transaktionen über IMS-Web ausführen oder REXX, Perl und andere Sprachen für Ihre Anwendungen verwenden. Außerdem werden Änderungen an einem Makro sofort in einem Browser angezeigt.

Net.Data erweitert die Datenverwaltungsfunktionen, die bereits für Ihr Betriebssystem entwickelt wurden, indem sowohl Daten als auch die zugehörige Geschäftslogik für das Web aktiviert werden. Im einzelnen gilt für Net-.Data Folgendes:

- Es wird eine einfache, jedoch leistungsfähige Makrosprache bereitgestellt, die die rasche Entwicklung von Internet- und Intranet-Anwendungen ermöglicht.
- In Ihren Webanwendungen kann die Datengenerierungslogik von der Darstellungslogik getrennt werden. Bei Net.Data gibt es keine Einschränkungen hinsichtlich der Darstellungsmethode für die Daten (z. B. HTML oder JavaScript). Durch diese Trennung können die Benutzer die Darstellung der Daten leicht entsprechend den neuesten Darstellungstechniken ändern.
- Vorhandene Erfahrung und Geschäftslogik können zum Generieren von Webseiten verwendet werden, indem Programme, die in C, C++, REXX, Java oder anderen Sprachen geschrieben wurden, eingebunden werden können.
- Komplexe Internet-Anwendungen können mit Hilfe einer einfachen Makrosprache schnell entwickelt werden.
- Es wird ein leistungsfähiger Zugriff auf Daten ermöglicht, die in DB2 und in jeder beliebigen fernen DRDA-fähigen Datenbank gespeichert sind.
- Makros können mühelos zwischen allen von der Net.Data-Produktfamilie unterstützten Betriebssystemen migriert werden.

Interpreter-Makrosprache

Die Net.Data-Makrosprache ist eine Interpreter-Sprache. Wenn Net.Data zur Verarbeitung eines Makros aufgerufen wird, interpretiert Net-.Data direkt jede Sprachanweisung und zwar sequentiell oben in der Datei beginnend. Bei dieser Vorgehensweise können Sie am Makro vorgenommene Änderungen bei der nächsten Angabe der URL-Adresse, die das Makro ausführt, sofort sehen. Eine Neukompilierung ist nicht erforderlich.

Direktanforderungen

Für einfache Anforderungen, die die Ausführung einer einzelnen SQL-Anweisung, einer gespeicherten DB2-Prozedur, eines REXX-Programms, eines C- bzw. C++-Programms oder eines Perl-Scripts erfordern, braucht kein Makro erstellt zu werden. Diese Anforderungen können direkt in der URL-Adresse angegeben werden, die vom Browser an den Webserver übergeben wird.

Freies Format

Die Net.Data-Makrosprache weist nur einige wenige Regeln zum Programmformat auf. Diese Unkompliziertheit ermöglicht Program-

mierern ein hohes Maß an Freiheit und Flexibilität. Eine einzelne Anweisung kann sich über mehrere Zeilen erstrecken, oder mehrere Anweisungen können auf einer einzelnen Zeile eingegeben werden. Die Anweisungen können in einer beliebigen Spalte beginnen. Hierbei können Leerzeichen und sogar ganze Zeilen übersprungen werden. Kommentare können an einer beliebigen Stelle eingefügt werden.

Variablen ohne Typ

Net.Data interpretiert alle Daten als Zeichenfolgen. Net.Data führt mit integrierten Funktionen arithmetische Operationen für eine Zeichenfolge aus, die eine gültige Zahl darstellt, einschließlich jener in Exponentialschreibweise. Die Variablen der Makrosprache werden in „Net.Data-Makrovariablen" auf Seite 106 näher beschrieben.

Integrierte Funktionen

Net.Data verfügt über integrierte Funktionen, die verschiedene Verarbeitungsprozesse, Such- und Vergleichsoperationen sowohl für Text als auch für Zahlen ausführen. Andere integrierte Funktionen bieten Unterstützung bei der Formatierung und bei arithmetischen Berechnungen.

Fehlerbehandlung

Wenn Net.Data einen Fehler feststellt, werden entsprechende Nachrichten mit Erklärungen an den Client zurückgegeben. Sie können die Fehlernachrichten anpassen, bevor sie über einen Browser an einen Benutzer zurückgegeben werden. Weitere Informationen finden Sie in „Konfigurationsvariablenanweisungen" auf Seite 14 und im Handbuch *Net.Data Reference*.

Kapitel 2. Konfigurieren von Net.Data

Sie können Net.Data für Ihr Betriebssystem installieren, indem Sie die Anweisungen in der mit dem Produkt gelieferten Informationsdatei (README) befolgen. Die meisten Konfigurationsschritte werden während der Installation abgeschlossen. Im einzelnen hängt dies jedoch vom Betriebssystem ab.

Nach der Installation von Net.Data für Ihr Betriebssystem müssen Sie den Webserver und die Net.Data-Konfigurationen ändern. Zu den Konfigurations-Tasks gehören:

- Anpassen der Net.Data-Initialisierungsdatei (INI)
- Konfigurieren von Net.Data für CGI, FastCGI, einer der unterstützten Webserver-APIs (optional) oder Net.Data-Servlets
- Anpassen der Konfigurations- und Umgebungsvariablendateien für den Webserver
- Konfigurieren des Cache-Managers (optional)
- Konfigurieren der Direktverbindung (optional)
- Definieren der Net.Data-Sprachumgebungen
- Angeben von Zugriffsrechten

Sie können Net.Data mit den folgenden Tools konfigurieren:

- Texteditor

  Bearbeiten Sie die Initialisierungsdatei und die Konfigurationsdateien für Direktverbindungen und für den Cache-Manager auf allen Betriebssystemen mit einem Texteditor. Mit einem Texteditor können Sie ferner die Konfigurationsdateien von Webservern aktualisieren. Es empfiehlt sich, die Dateien vor der Durchführung von Änderungen zu sichern.
- Net.Data Administration-Tool

  Administration-Tool bietet eine Grafikschnittstelle zum Anpassen der Initialisierungsdatei und der Konfigurationsdatei für Direktverbindungen. Mit Administration-Tool können Sie Net.Data auf den Betriebssystemen OS/2, Windows NT und AIX konfigurieren.

Die verwendete Methode hängt davon ab, welche Komponenten konfiguriert werden müssen und auf welchem Betriebssystem Net.Data ausgeführt wird, wie in Tabelle 1 auf Seite 6 beschrieben. Wenn Sie eine Konfigurationsaufgabe mit einer bestimmten Methode anfangen, sollten Sie die Verwendung dieser Methode beibehalten, um optimale Ergebnisse zu erzielen.

Tabelle 1. Gegenüberstellung der Konfigurationsmethoden mit Aufgaben und Betriebssystemen. **A** *- Kann mit Administration-Tool oder manuell konfiguriert werden.* **M** *- Kann nur manuell konfiguriert werden.*

Aufgabe	**Betriebssysteme:**			
	AIX	**NT**	**OS/2**	**HP | SUN | Linux**
Konfigurieren der Net.Data-INI-Datei	A	A	A	M
Definieren der Cliette-Anschlüsse	A	A	A	M
Definieren von Cliettes	A	A	A	M
Aktivieren der Cliette-Kennwortverschlüsselung	A	A	N/V	M
Aktivieren der Fehlerprotokollierung	A	A	A	M
Konfigurieren des Webservers für FastCGI, CGI und APIs*	M	M	M	M
Definieren der Cache-Manager-Anschlüsse	M	M	N/V	N/V
Konfigurieren des Cache-Managers	M	M	N/V	N/V

*__Hinweis:__ Viele Webserver bieten Verwaltungs-Tools, mit denen Sie den Webserver konfigurieren können.

In diesem Kapitel wird beschrieben, wie Sie Net.Data konfigurieren und Ihre Konfiguration des Webservers zur Verwendung mit Net.Data ändern können. Außerdem wird beschrieben, wie Sie optionale Komponenten konfigurieren können.

- „Anpassen der Net.Data-Initialisierungsdatei" auf Seite 12
- „Definieren der Net.Data-Sprachumgebungen" auf Seite 32
- „Konfigurieren der Direktverbindung" auf Seite 36
- „Konfigurieren des Webservers zur Verwendung mit CGI" auf Seite 42
- „Konfigurieren von Net.Data für FastCGI" auf Seite 44
- „Konfigurieren von Net.Data zur Verwendung mit Java-Servlets" auf Seite 46
- „Konfigurieren von Net.Data zur Verwendung mit den Webserver-APIs" auf Seite 47
- „Konfigurieren von Net.Data mit Net.Data Administration-Tool" auf Seite 50
- „Erteilen von Zugriffsrechten für Dateien, auf die Net.Data zugreift" auf Seite 65

Informationen zur Net.Data-Initialisierungsdatei

Net.Data verwendet seine Initialisierungsdatei zum Festlegen der Einstellungen verschiedener Konfigurationsvariablen und zum Konfigurieren der Sprachumgebungen und Suchpfade. Die Einstellungen von Konfigurationsvariablen steuern verschiedene Aspekte der Net.Data-Operation, unter anderem:

- Die Codierung von Zeichendaten als Unicode
- Die MBCS-Fähigkeit von Zeichenfolge- und Wortfunktionen
- Der Name des DB2-Exemplars für Zugriff auf Datenbankdaten
- Die Art der Verbindung und Kommunikation mit den Sprachumgebungen, Datenbanken, der Verbindungsverwaltung und dem Caching von Net.Data
- Die Aktivierung der Fehlerprotokollierung

Die Sprachumgebungsanweisungen definieren die Net.Data-Sprachumgebungen, die verfügbar sind, und geben spezielle Eingabe- und Ausgabepara- meterwerte an, die mit den Sprachumgebungen ausgetauscht werden. Die Sprachumgebungen ermöglichen es Net.Data, auf verschiedene Datenquellen, z. B. DB2-Datenbanken und Systemservices, zuzugreifen. Die Pfadanweisungen geben die Verzeichnispfade zu Dateien an, die Net.Data verwendet, z. B. Makros, REXX-Programme und Perl-Scripts.

Die Net.Data-Initialisierungsdatei (db2www.ini) befindet sich im Dokumentverzeichnis des Webservers. Weitere Informationen finden Sie in der Readme-Datei für Ihr Betriebssystem.

Hinweis zu Berechtigungen: Stellen Sie sicher, dass die Benutzer-ID, unter der der Webserver ausgeführt wird, über eine Berechtigung zum Lesen dieser Datei verfügt. Weitere Informationen finden Sie in „Erteilen von Zugriffsrechten für Dateien, auf die Net.Data zugreift" auf Seite 65.

Informationen zu den Net.Data-Konfigurationsdateien für optionale Komponenten

In den folgenden Abschnitten werden die Konfigurationsdateien für optionale Komponenten von Net.Data erläutert.

„Die Konfigurationsdatei für Direktverbindungen" auf Seite 8

„Die Konfigurationsdatei für den Cache-Manager" auf Seite 8

„Gemeinsame Abschnitte der Initialisierungs-, Steuer- und Makrodateien von Net.Data" auf Seite 9

Die Konfigurationsdatei für Direktverbindungen

Eine Direktverbindung bietet Verbindungsverwaltung unter den Betriebssystemen Windows NT, OS/2, AIX, Linux und Sun Solaris, um die Leistung durch Beseitigen des Systemaufwands beim Start zu steigern. Die Konfigurationsdatei für die Net.Data-Direktverbindung enthält Informationen zu mindestens einer benannten Cliette. Eine Cliette ist ein Langzeitprozess, der eine Verbindung zu einer Datenbank verwaltet, oder eine Java Virtual Machine, die Net.Data-Makroaufrufe von mehreren Benutzern überdauert. Eine Cliette ist nach ihrem Start so lange vorhanden, bis die Net.Data-Direktverbindung beendet wird. Es können mehrere Cliettes mit einer einzelnen Datenbank verbunden sein.

Sie geben als Teil der Cliette-Informationen in der Konfigurationsdatei einen Cliette-Namen und die Mindest- und Höchstanzahl von Prozessen an. Bei Datenbank-Cliettes können Sie auch für jeden Cliette-Eintrag den Datenbanknamen, den Anmeldenamen und das Kennwort angeben.

Hinweis zu Berechtigungen: Stellen Sie sicher, dass die Benutzer-ID, mit der Connection Manager gestartet wird, über die Berechtigung zum Lesen dieser Datei verfügt. Weitere Informationen finden Sie in „Erteilen von Zugriffsrechten für Dateien, auf die Net.Data zugreift" auf Seite 65.

Die Konfigurationsdatei für den Cache-Manager

Die Konfigurationsdatei für den Cache-Manager enthält die Definitionen für den Cache-Manager und für jeden Cache. Net.Data-Caching wird in „Net.Data-Caching" auf Seite 205 beschrieben. Das Konfigurieren des Cache-Managers wird in „Konfigurieren des Cache-Managers und der Net.Data-Caches" auf Seite 212 beschrieben. Die Struktur der Datei besteht aus einer Reihe von Abschnitten, d. h. Zeilengruppen:

Zeilengruppe für den Cache-Manager
Diese Zeilengruppe definiert die Parameter des Cache-Managers selbst und enthält Netzwerkinformationen, den Protokollierungsstatus und den Ablaufverfolgungsstatus. Die Zeilengruppe ist erforderlich und muss als `cache-manager` bezeichnet werden.

Zeilengruppen für die Cache-Definition
Diese Zeilengruppen definieren die Parameter für jeden Cache. In der Konfigurationsdatei gibt es für jeden vom Cache-Manager verwalteten Cache eine Zeilengruppe für die Cache-Definition. Dieser Abschnitt enthält Netzwerkinformationen, Informationen zum Hauptspeicher- und Plattenspeicherbedarf, den Protokollierungsstatus und den Statistikdatenstatus. Die Zeilengruppe für die Cache-Definition ist für jeden vom Cache-Manager verwalteten Cache erforderlich.

Die Konfigurationsdatei für den Cache-Manager wird nicht durch Administration-Tool verwaltet und kann mit einem beliebigen Texteditor aktualisiert werden. Informationen zum Definieren dieser Datei finden Sie in „Net.Data-Caching" auf Seite 205.

Hinweis zu Berechtigungen: Stellen Sie sicher, dass die Benutzer-ID, mit welcher der Cache-Manager gestartet wird, Zugriffsrechte für diese Datei hat. Weitere Informationen finden Sie in „Erteilen von Zugriffsrechten für Dateien, auf die Net.Data zugreift" auf Seite 65.

Gemeinsame Abschnitte der Initialisierungs-, Steuer- und Makrodateien von Net.Data

Bestimmte Abschnitte der Initialisierungs-, Konfigurations- und Makrodateien von Net.Data müssen für eine ordnungsgemäße Zusammenarbeit aller Net.Data-Komponenten konsistent sein. Die folgende Tabelle gibt einen Überblick über die Bereiche dieser Dateien, die übereinstimmen müssen.

Tabelle 2. Konsistenzvoraussetzungen für die Net.Data-Konfigurationsdateien und das Makro

Datei	Gemeinsame Abschnitte	Anmerkungen
Net.Data-INI-Datei	Anweisung ENVIRONMENT	Die Sprachumgebungen, die eine Direktverbindung verwenden, müssen den Datenbank-Cliette-Namen in ihrer Anweisung ENVIRONMENT angeben.
	Konfigurationsvariablen für Direktverbindungen	Wenn Sie eine Net.Data-Direktverbindung verwenden, geben Sie den Direktverbindungsanschluss DTW_CM_PORT an. Dieser Variablenwert muss mit dem Wert für MAIN_PORT in der Konfigurationsdatei für Direktverbindungen übereinstimmen.
	Cache-Konfigurationsvariablen	Wenn Sie Net.Data-Caching verwenden, können Sie optional Anschlussnummer- und Einheitennamenvariablen angeben. Diese Werte müssen mit den Werten in der Konfigurationsdatei für den Cache-Manager übereinstimmen, sofern verwendet.

Tabelle 2. Konsistenzvoraussetzungen für die Net.Data-Konfigurationsdateien und das Makro (Forts.)

Datei	Gemeinsame Abschnitte	Anmerkungen
Konfigurationsdatei für Direkt-verbindungen	Cliette-Definitionen	Jede Cliette-Definition muss mit einer entsprechenden Definition in der INI-Datei übereinstimmen. Zudem muss der Wert für MAIN_PORT mit dem Variablenwert für DTW_CM_PORT in der INI-Datei übereinstimmen.
Konfigurationsdatei für den Cache-Manager	Konfigurationsvariablen für den Cache-Manager	Wenn Sie Net.Data-Caching verwenden, können Sie optional Anschlussnummer- und Einheiten-namenvariablen angeben. Diese Werte müssen mit den Werten in der INI-Datei übereinstimmen, sofern verwendet.

Die folgenden Ausschnitte veranschaulichen die Beziehung zwischen einem Makro, einer Net.Data-Initialisierungsdatei und einer Konfigurationsdatei für Direktverbindungen. Vom Makro werden zwei Cliettes verwendet (`DTW_SQL:SAMPLE, DTW_SQL:CELDIAL`), die auf zwei DB2-Datenbanken namens `SAMPLE` und `CELDIAL` zugreifen. Die Konfigurationsdatei für Direktverbindungen enthält die Namen und Definitionen der Cliettes. Die Anweisung ENVIRONMENT in der Net.Data-Initialisierungsdatei verweist auf den Cliette-Namen. Die Werte für LOGIN und PASSWORD werden in der Konfigurationsdatei für Direktverbindungen angegeben.

Abb. 2 auf Seite 11 zeigt einen Ausschnitt des Makros, das die Anweisung @DTW_ASSIGN enthält, die definiert, welche Cliette zum Zugriff auf eine Datenbank verwendet werden soll.

```
<3*****************************************>
<3** This is an HTML comment
<3** Access the SAMPLE database using
<3** 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** This is an HTML comment
<3** Process the CELDIAL database using
<3** the 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)
```

Abbildung 2. Net.Data-Makroausschnitt

Beachten Sie, dass die Konfigurationsvariable DATABASE durch die Anweisung ENVIRONMENT der Initialisierungsdatei ersetzt wird, um den Cliette-Namen zu generieren. Dies ermöglicht den Zugriff auf mehrere Datenbanken vom gleichen Makro aus.

Abb. 3 zeigt einen Ausschnitt der Net.Data-Initialisierungsdatei, die die Anweisung ENVIRONMENT und den zugehörigen Cliette-Typ enthält. In der Initialisierungsdatei gibt es eine Anweisung ENVIRONMENT für jeden Cliette-Typ. Die Anweisung ENVIRONMENT gibt für jeden Datenbank-Cliette-Typ einen Cliette-Namen an. Der Name besteht aus dem Cliette-Typ und dem Variablenverweis $(DATABASE), der während der Laufzeit aufgelöst wird. Jede Sprachumgebung, die Direktverbindungen verwendet, muss in der Anweisung ENVIRONMENT eine Cliette-Definition aufweisen.

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

Abbildung 3. Ausschnitt einer Net.Data-Initialisierungsdatei

Abb. 4 zeigt einen Ausschnitt der Konfigurationsdatei für Direktverbindungen, die die Cliette-Definitionen für DTW_SQL:CELDIAL und DTW_JAVAPPS enthält.

```
      CONNECTION_MANAGER{
      MAIN_PORT=7128
      ENCRYPTION=key
  }

      ##################################################################
      # This is a comment in a Live Connection configuration file.
      # Comments start with a pound (hash) character.
      # Comments terminate at the end of the line and do not continue to
      # the next line unless another pound (hash) character is specified.
      # You can include comments at the end of lines containing Live
      # Connection keywords except on password lines.
      # You cannot include comments anywhere on lines containing the
      # password keyword.
      # You cannot include spaces and pound (hash) characters within any
      # name, such as cliette name or in database cliette passwords.
     ##################################################################
      CLIETTE DTW_SQL:CELDIAL{       MIN_PROCESS=1
      MAX_PROCESS=5
      EXEC_NAME=./dtwcdb2
      DATABASE=CELDIAL
      LOGIN=marshall
      PASSWORD=encrypted_password
      }

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

Abbildung 4. Ausschnitt einer Konfigurationsdatei für Direktverbindungen

Anpassen der Net.Data-Initialisierungsdatei

Die in der Initialisierungsdatei enthaltenen Informationen werden mit drei Arten von Konfigurationsanweisungen angegeben, die in den folgenden Abschnitten beschrieben werden:

- „Konfigurationsvariablenanweisungen" auf Seite 14
- „Pfadkonfigurationsanweisungen" auf Seite 24
- „Umgebungskonfigurationsanweisungen" auf Seite 29

Die in Abb. 5 auf Seite 13 gezeigte Beispielinitialisierungsdatei enthält Beispiele dieser Anweisungen und ist für OS/2 und Windows NT gültig.

Der Text jeder einzelnen Konfigurationsanweisung muss vollständig in einer Zeile stehen. Stellen Sie sicher, dass die Initialisierungsdatei eine Anweisung ENVIRONMENT für jede Sprachumgebung enthält, die Sie von Ihren Makros aus aufrufen. Sie müssen keine Pfadkonfigurationsanweisungen angeben, wenn Sie bei allen Verweisen auf Dateien innerhalb des Makros den vollständig qualifizierten Pfad angeben.

```
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]
```

- Die Zeilen 1 - 8 definieren Konfigurationsvariablen.
- Die Zeilen 9 - 13 definieren Pfade zu Dateien, die zur Verarbeitung des Makros erforderlich sind.
- Die Zeilen 14 - 25 definieren die verfügbaren ENVIRONMENT-Anweisungen.

Abbildung 5. Die Net.Data-Initialisierungsdatei. Eine vollständige Beschreibung des DLL-Pfads und der Parameterliste finden Sie in der Datei db2www.ini selbst und in „Umgebungskonfigurationsanweisungen" auf Seite 29.

In den folgenden Abschnitten wird beschrieben, wie Sie die Konfigurationsanweisungen in der Initialisierungsdatei anpassen können.

- „Konfigurationsvariablenanweisungen" auf Seite 14
- „Pfadkonfigurationsanweisungen" auf Seite 24

Die folgenden Änderungen an den ENVIRONMENT-Anweisungen sind *erforderlich*:

- Entfernen Sie die Variable RETURN_CODE aus der Parameterliste jeder Anweisung ENVIRONMENT, in der sie vorkommt.
- Entfernen Sie die ENVIRONMENT-Anweisungen DTW_DEFAULT, DTW_FILE und DTW_APPLET.

Folgende Änderungen müssen berücksichtigt werden, weil sich einige Konfigurationsstandardwerte geändert haben:

- Wenn für Ihre Anwendungen die Verwendung der Variable SHOWSQL erforderlich ist, dann ändern Sie die Konfigurationsvariable DTW_SHOWSQL in YES. Informationen zur Syntax und Beispiele finden Sie in „DTW_SHOWSQL: Konfigurationsvariable zum Aktivieren bzw. Inaktivieren von SHOWSQL" auf Seite 20.
- Wenn für Ihre Anwendungen die Verwendung des Direktanforderungsaufrufs erforderlich ist, dann ändern Sie die Konfigurationsvariable DTW_DIRECT_REQUEST in YES. Informationen zur Syntax und Beispiele finden Sie in „DTW_DIRECT_REQUEST: Variable zum Aktivieren der Direktanforderung" auf Seite 18.
- Wenn Sie das Net.Data-Fehlerprotokoll oder das Net.Data-Trace nicht in /usr/lpp/netdata/logs/ speichern wollen, müssen Sie für die Konfigurationsvariable DTW_ERROR_LOG_DIR bzw. DTW_TRACE_LOG_DIR das entsprechende Verzeichnis angeben.

Konfigurationsvariablenanweisungen

Die Net.Data-Konfigurationsvariablenanweisungen legen die Werte der Konfigurationsvariablen fest. Konfigurationsvariablen werden für verschiedene Zwecke verwendet. Einige Variablen sind bei bestimmten Sprachumgebungen für die ordnungsgemäße Funktion oder den Betrieb in einem alternativen Modus erforderlich. Andere Variablen steuern die Zeichenverschlüsselung oder den Inhalt der momentan erstellten Webseite. Mit Konfigurationsvariablenanweisungen können Sie zudem anwendungsspezifische Variablen definieren.

Die verwendeten Konfigurationsvariablen hängen von den verwendeten Sprachumgebungen und Datenbanken sowie anderen, anwendungsspezifischen Faktoren ab.

Gehen Sie wie folgt vor, um die Konfigurationsvariablenanweisungen zu aktualisieren:

Passen Sie die Initialisierungsdatei mit den Konfigurationsvariablen an, die für Ihre Anwendung erforderlich sind. Eine Konfigurationsvariable hat die folgende Syntax:

```
NAME[=]value-string
```

Das Gleichheitszeichen ist optional, was durch die eckigen Klammern angegeben wird.

In den folgenden Unterabschnitten werden die Konfigurationsvariablenanweisungen beschrieben, die Sie in der Initialisierungsdatei angeben können:

- „Konfigurationsvariablen für den Cache-Manager" auf Seite 15
- „DB2INSTANCE: Variable für DB2-Exemplar" auf Seite 17

Konfigurationsvariablen für den Cache-Manager

Es werden zwei optionale Konfigurationsvariablen verwendet, wenn der Cache-Manager auf einer anderen Maschine ausgeführt wird als der, auf der das Net.Data-Makro ausgeführt wird:

- DTW_CACHE_PORT gibt die Anschlussnummer an, mit der Net.Data die Verbindung zum Cache-Manager herstellt.
- DTW_CACHE_HOST gibt den TCP/IP-Host-Namen der lokalen oder fernen Maschine an.

Wenn der Cache-Manager auf der lokalen Maschine ausgeführt wird, werden UNIX-Domänen-Sockets oder benannte Pipes zur Übertragung verwendet. In diesem Fall ist keine Konfiguration erforderlich. Der Cache-Manager kann nur auf Maschinen unter AIX oder Windows NT ausgeführt werden. Informationen zum Net.Data-Caching finden Sie in „Net.Data-Caching" auf Seite 205.

DTW_CACHE_PORT: Variable für Cache-Manager-Anschluss

Gibt den TCP/IP-Anschluss an, über den der Cache-Manager empfängt.

Diese Anschlussnummer muss mit der Anschlussnummer übereinstimmen, die in der Konfigurationsdatei für den Cache-Manager angegeben ist, damit Net.Data mit dem Cache-Manager kommunizieren kann. Wenn diese Variable nicht angegeben wird, verwendet der Cache-Manager den Standardanschluss 7175.

Syntax:

```
DTW_CACHE_PORT [=] port_number
```

Parameter:

port_number
Eine eindeutige Anschlussnummer, die dem Cache-Manager zugeordnet ist, um Cache-Anforderungen zu bedienen. Der Standardwert ist 7175.

Tabelle 3 beschreibt die Optionen zum Angeben der Maschinen-IDs und Anschlussnummern für diese Variablen.

Tabelle 3. Konfigurationsvariablen für den Cache-Manager: Konfigurationsoptionen

Standardwerte für Connection Manager	Bei Angabe der Cache-Maschine	Bei fehlender Angabe der Cache-Maschine
Bei Angabe des Cache-Anschlusses	Net.Data stellt mit dem angegebenen Anschluss die Verbindung zum Cache-Manager auf der angegebenen Maschine her.	Net.Data stellt mit dem angegebenen Anschluss die Verbindung zum Cache-Manager auf der lokalen Maschine her.
Bei fehlender Angabe des Cache-Anschlusses	Net.Data stellt mit dem Standardanschluss 7175 die Verbindung zum Cache-Manager auf der angegebenen Maschine her.	Net.Data stellt mit dem Standardanschluss 7175 die Verbindung zum Cache-Manager auf der lokalen Maschine her.

DTW_CACHE_HOST: Variable für Cache-Manager-Maschinen-ID

Gibt die Maschine an, auf der sich der Cache-Manager befindet. Wenn diese Variable nicht angegeben wird, geht Net.Data davon aus, dass die korrekte Maschine die lokale Maschine ist.

Syntax:

```
DTW_CACHE_HOST [=] host_name
```

Parameter:

host_name
Der qualifizierte TCP/IP-Host-Name der lokalen oder fernen Maschine, auf der der Cache-Manager ausgeführt wird. Der Standardwert ist der Host-Name der lokalen Maschine.

DB2INSTANCE: Variable für DB2-Exemplar

Gibt das DB2-Exemplar an, das von der SQL-Sprachumgebung verwendet wird. Dieser Variablenwert ist erforderlich, wenn Net.Data die Verbindung zu DB2 herstellt und dieses Produkt auf den Betriebssystemen Windows NT, OS/2 und UNIX ausgeführt wird.

Für DB2 unter OS/2, Windows NT und UNIX muss DB2INSTANCE als Umgebungsvariable definiert werden. Wenn Net.Data feststellt, dass DB2INSTANCE nicht als Umgebungsvariable definiert ist, setzt es die Umgebungsvariable DB2INSTANCE auf den in der INI-Datei vorhandenen Wert von DB2INSTANCE, bevor die Verbindungsherstellung zu DB2 versucht wird.

Syntax:

```
DB2INSTANCE [=] instance_name
```

DTW_CM_PORT: Variable für Anschlussnummer bei Direktverbindung

Gibt eine eindeutige Anschlussnummer an, die Net.Data für Direktverbindungen verwendet.

Syntax:

```
DTW_CM_PORT [=] port_number
```

Dabei gilt Folgendes: *port_number* gibt die für Direktverbindungen verwendete eindeutige Anschlussnummer an.

DTW_DEFAULT_ERROR_MESSAGE: Angeben generischer Fehlernachrichten

Mit der Konfigurationsvariablen DTW_DEFAULT_ERROR_MESSAGE geben Sie eine generische Fehlernachricht für Anwendungen in der Produktion an. Diese Variable stellt eine generische Nachricht für Fehlerbedingungen zur Verfügung, die in keinem MESSAGE-Block erfasst sind.

Wenn Sie dennoch die ursprünglich von Net.Data generierten Fehlernachrichten sehen wollen, müssen Sie die Nachrichten mit Hilfe der Fehlernachrichtenprotokollierung erfassen. Informationen zur Verwendung des Fehlerprotokolls finden Sie in „Kapitel 8. Net.Data-Protokollierung” auf Seite 237.

Wird die Konfigurationsvariable nicht angegeben, zeigt Net.Data eine eigene Nachricht für die Fehlerbedingung an.

Syntax:

```
DTW_DEFAULT_ERROR_MESSAGE [=] "message"
```

Beispiel: Gibt eine generische Nachricht an

```
DTW_DEFAULT_ERROR_MESSAGE "This site is temporarily unavailable."
```

DTW_DIRECT_REQUEST: Variable zum Aktivieren der Direktanforderung

Aktiviert bzw. inaktiviert den Net.Data-Direktanforderungsaufruf. Die Direktanforderung ist standardmäßig inaktiviert.

Bei der Direktanforderungsmethode zum Aufrufen von Net.Data kann ein Benutzer die Ausführung einer SQL-Anweisung bzw. eines Perl-, REXX- oder C-Programms direkt in einer URL-Adresse angeben. Ist die Direktanforderung inaktiviert, muss der Benutzer Net.Data mit der Makroanforderungsmethode aufrufen, d. h., Benutzer können nur jene SQL-Anweisungen und Funktionen ausführen, die in einem Makro definiert sind bzw. aufgerufen werden. Empfehlungen zur Sicherheit bei der Verwendung von DTW_DIRECT_REQUEST finden Sie in „Verwenden von Net.Data-Mechanismen" auf Seite 71.

Syntax:

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

Dabei gilt Folgendes:

YES Aktiviert die Net.Data-Direktanforderung.

NO Inaktiviert die Net.Data-Direktanforderung. Der Standardwert ist NO.

DTW_INST_DIR: Variable für Net.Data-Installationsverzeichnis

Lokalisiert bestimmte Dateien während der Net.Data-Ausführung. Sie setzen diese Variable während der Installation auf das Ausgangsverzeichnis (inst_dir), in dem Net.Data installiert wird. Ändern Sie diesen Wert nach der Installation nicht.

HTML_PATH

Gibt an, in welches Verzeichnis Net.Data große Objekte (LOBs) schreibt.

Während der Installation erstellt Net.Data das Verzeichnis `tmplobs` unter dem in der Pfadkonfigurationsvariablen HTML_PATH angegebenen Verzeichnis. Net.Data speichert alle LOB-Dateien in diesem Verzeichnis. Wenn Sie den Wert von HTML_PATH ändern, erstellen Sie ein neues Unterverzeichnis unter dem neuen Verzeichnis.

Syntax:

```
HTML_PATH [=] path
```

Beispiel: Das folgende Beispiel zeigt die Konfigurationsvariable HTML_PATH in der Initialisierungsdatei.

```
HTML_PATH /db2/lobs
```

Wenn eine Abfrage ein LOB zurückgibt, sichert Net.Data diese Datei in dem durch die Konfigurationsanweisung HTML_PATH angegebenen Verzeichnis.

Hinweis: Berücksichtigen Sie bei der Verwendung von LOBs die Systemeinschränkungen, denn LOBs können Ressourcen schnell aufbrauchen. Weitere Informationen finden Sie in „Verwenden großer Objekte" auf Seite 161.

DTW_LOG_DIR und DTW_LOG_LEVEL: Fehlerprotokollvariablen

DTW_LOG_DIR gibt das Verzeichnis an, in dem die Fehlerprotokolle gespeichert sind. Die Protokollierung findet nur dann statt, wenn diese und die Variable DTW_LOG_LEVEL definiert sind.

Weitere Informationen zu diesen Variablen und zum Protokollieren von Fehlernachrichten mit Net.Data finden Sie in „Protokollieren von Net.Data-Fehlernachrichten" auf Seite 237.

Syntax:

```
DTW_LOG_DIR [=] \inst_dir\path
```

Beispiel: Konfiguration der Initialisierungsdatei

```
DTW_LOG_DIR \inst_dir\mylogfiles\
```

DTW_LOG_LEVEL: Variable für Fehlerprotokollstufe

DTW_LOG_LEVEL gibt die Stufe der Fehler an, die in den Fehlerprotokollen aufgezeichnet werden soll. Die Protokollierung findet nur dann statt, wenn diese und die Variable DTW_LOG_DIR definiert sind.

Weitere Informationen zu diesen Variablen und zum Protokollieren von Fehlernachrichten mit Net.Data finden Sie in „Protokollieren von Net.Data-Fehlernachrichten" auf Seite 237.

Syntax:

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

Beispiel: Konfiguration der Initialisierungsdatei

```
DTW_LOG_LEVEL error
```

DTW_MBMODE: Variable für Unterstützung der Landessprache

Aktiviert die Unterstützung in der Landessprache für Wort- und Zeichenfolgefunktionen. Wenn der Wert für die Variable YES ist, verarbeiten alle Zeichenfolge- und Wortfunktionen MBCS-Zeichen in Zeichenfolgen ordnungsgemäß, indem sie sie als gemischte Daten behandeln (d. h. als Zeichenfolgen, die möglicherweise Zeichen aus Einzelbytezeichensätzen und Doppelbytezeichensätzen enthalten). Der Standardwert ist NO. Sie können den in der Initialisierungsdatei festgelegten Wert überschreiben, indem Sie die Variable DTW_MBMODE in einem Net.Data-Makro entsprechend einstellen.

Diese Konfigurationsvariable steht mit der Konfigurationsvariablen DTW_UNICODE in Verbindung. Wenn DTW_ UNICODE den Standardwert NO verwendet, wird der Wert von DTW_MBMODE verwendet. Wenn DTW_UNICODE auf einen anderen Wert als NO gesetzt ist, wird dieser Wert verwendet. Tabelle 4 veranschaulicht, wie die Einstellungen dieser beiden Variablen festlegen, wie integrierte Funktionen Zeichenfolgen verarbeiten:

Tabelle 4. Beziehung zwischen den Einstellungen von DTW_UNICODE und DTW_MBMODE

DTW_UNICODE-Einstellung	DTW_MBMODE=YES	DTW_MBMODE=NO
NO	Unterstützt MBCS gemischt mit SBCS	Unterstützt nur SBCS
UTF8	Unterstützt UTF-8	Unterstützt UTF-8

Syntax:

```
DTW_MBMODE [=] NO|YES
```

DTW_REMOVE_WS: Variable für das Entfernen von Leerzeichen

Wird für diese Variable YES definiert, entfernt Net.Data überflüssige Leerzeichen aus der HTML-Ausgabe. Diese Variable verringert durch Komprimierung von Leerzeichen die Menge der an den Web-Browser gesendeten Daten, was zu einer höheren Leistung führt. Der Standardwert ist NO.

Sie können diese Variable im Makro mit der Anweisung DEFINE überschreiben.

Syntax:

```
DTW_REMOVE_WS [=] YES|NO
```

DTW_SHOWSQL: Konfigurationsvariable zum Aktivieren bzw. Inaktivieren von SHOWSQL

Überschreibt die Wirksamkeit der Einstellung SHOWSQL in Ihren Net.Data-Makros.

Syntax:

```
DTW_SHOWSQL [=] YES|NO
```

Dabei gilt Folgendes:

YES Aktiviert SHOWSQL in einem Makro, das den Wert von SHOWSQL auf YES setzt.

NO Inaktiviert SHOWSQL in Ihren Makros, selbst wenn die Variable SHOWSQL auf YES gesetzt ist. Der Standardwert ist NO.

Tabelle 5 beschreibt, wie die Einstellungen in der Net.Data-Initialisierungsdatei und das Makro festlegen, ob die Variable SHOWSQL für ein bestimmtes Makro aktiviert oder inaktiviert ist.

Tabelle 5. Die Beziehung zwischen den Einstellungen in der Net.Data-Initialisierungsdatei und dem Makro für SHOWSQL

Einstellung von DTW_SHOWSQL	Einstellung SHOWSQL	SQL-Anweisung angezeigt
NO	NO	NO
NO	YES	NO
YES	NO	NO
YES	YES	YES

DTW_SMTP_SERVER: Variable für E-Mail-SMTP-Server

Gibt den SMTP-Server an, der zum Senden von E-Mail-Nachrichten über die integrierte Funktion DTW_SENDMAIL verwendet werden soll. Der Wert dieser Variablen kann ein Host-Name oder eine IP-Adresse sein. Wenn diese Variable nicht gesetzt wird, verwendet Net.Data den lokalen Host als SMTP-Server.

Syntax:

```
DTW_SMTP_SERVER [=] server_name
```

Dabei gilt Folgendes: `server_name` ist der Host-Name bzw. die IP-Adresse des SMTP-Servers, der zum Senden von E-Mail-Nachrichten verwendet werden soll.

Hinweis zur Leistung: Geben Sie eine IP-Adresse für diesen Wert an, um zu verhindern, dass Net.Data die Verbindung zu einem Domänennamensserver herstellt, wenn die IP-Adresse des angegebenen SMTP-Servers abgerufen wird.

Beispiel:

```
DTW_SMTP_SERVER us.ibm.com
```

DTW_UNICODE: Variable für Unicode

Gibt an, ob Net.Data Unicode in folgenden Elementen unterstützt:

- Makros
- Formulardaten
- Von einer DB2-Datenbank empfangene Daten
- Durch integrierte Net.Data-Funktionen verarbeitete Zeichenfolgen

Net.Data unterstützt das UTF-8-Unicode-Format in Makros, Formulardaten und integrierten Funktionen, und die Ausgabe erfolgt immer in UTF-8. Net-.Data kann auf eine Datenbank zugreifen, die UCS-2-Daten enthält, und diese in UTF-8 umsetzen.

Wird DTW_UNICODE auf UTF8 gesetzt, wird Net.Data zur Ausführung in einer Unicode-Umgebung angewiesen. Net.Data generiert dann Seiten in UTF-8 und erwartet alle Eingabedaten im UTF-8-Format (bei DB2-Datenbankdaten wird auch UCS-2 akzeptiert). Zu den Eingabedaten gehören der Inhalt der Makrodatei, die vom Browser gesendeten Formulardaten und alle anderen Daten aus externen Datenquellen.

Voraussetzung für DB2-Unicode-Datenbank: Neben der Definition der Variable DTW_UNICODE müssen Sie auch für die DB2-spezifische Umgebungsvariable, DB2CODEPAGE, 1208 in der Umgebung, in der Net.Data aktiv ist, definieren. Für den Apache-Webserver müssen Sie beispielsweise der Datei HTTPD.CONF folgende Zeile hinzufügen:

```
SetEnv DB2CODEPAGE 1208
```

Lesen Sie in Ihrer Webserverdokumentation nach, wie Umgebungsvariablen für CGI-Scripts, Webserver-APIs, FastCGI-Programme oder Servlets definiert werden. Net.Data verwendet in einer Unicode-Umgebung den englischen Nachrichtenkatalog.

Die Konfigurationsvariable DTW_UNICODE steht mit der Konfigurationsvariablen DTW_MBMODE in Verbindung. Der Wert der Konfigurationsvariablen DTW_UNICODE überschreibt die Einstellung der Variablen DTW_MBMODE bei der Verarbeitung von integrierten Wort- und Zeichenfolgefunktionen. Wird für DTW_UNICODE jedoch NO definiert oder wird diese Variable nicht definiert, wird der Wert DTW_MBMODE verwendet. Tabelle 4 auf Seite 20 veranschaulicht, wie die Einstellungen dieser beiden Variablen festlegen, wie integrierte Funktionen Zeichenfolgen verarbeiten:

Syntax:

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

Dabei gilt Folgendes:

NO Gibt die Verwendung des Werts für die Variable DTW_MBMODE an. Tabelle 4 auf Seite 20 beschreibt Net.Data-Unterstützung basierend auf dem Wert von DTW_MBMODE.

UTF8 Gibt die Unterstützung der UTF-8-Codepage und das Ignorieren des Werts für die Konfigurationsvariable DTW_MBMODE an. UTF-8 stellt Zeichen durch eine unterschiedliche Anzahl von Byte dar und kann auch für das ASCII-Format sicher verwendet werden.

DTW_UPLOAD_DIR

Gibt an, in welchem Verzeichnis Net.Data Dateien speichert, die vom Client hochgeladen werden. Wenn diese Variable nicht gesetzt wird, lehnt Net.Data die Dateien für das Hochladen ab.

Syntax:

```
DTW_UPLOAD_DIR [=] path
```

Beispiel:

```
DTW_UPLOAD_DIR /tmp/uploads
```

DTW_USE_DB2_PREPARE_CACHE

Gibt an, dass Net.Data den DB2-Vorbereitungs-Cache ohne explizite Verwendung von Parametermarken in der SQL-Anweisung der Makrodatei nutzen soll. Wenn alle Ihre Makros diese Funktion nutzen sollen, müssen Sie die Konfigurationsvariable DTW_USE_DB2_PREPAR_CACHE in der Net.Data-Initialisierungsdatei auf YES setzen. Soll diese Funktion nur für die Anweisungen in einem bestimmten Makro aktiviert werden, können Sie die Makrovariable DTW_USE_DB2_PREPARE_CACHE verwenden. Weitere Informationen finden Sie im Handbuch *Net.Data Reference*.

Syntax:

```
DTW_USE_DB2_PREPARE_CACHE [=] YES|NO
```

Dabei gilt Folgendes:

YES Gibt an, dass Net.Data alle SQL-Anweisungen so ändert, dass der Vorbereitungs-Cache genutzt wird. Diese Funktion können Sie für eine bestimmte SQL-Anweisung inaktivieren, indem Sie die Makrovariable mit Hilfe von %DEFINE oder @DTW_ASSIGN() auf NO setzen.

NO Gibt an, dass Net.Data die SQL-Anweisung unverändert lässt. Dies ist der Standardwert.

DTW_VARIABLE_SCOPE: Variable für Variablenbereich

Gibt an, wie Net.Data den lokalen Variablenbereich behandelt, d. h. ob lokale Variablen nur lokal gültig sind oder ob lokale Variablen außerhalb des Funktionsblocks, in dem sie erstellt wurden, verwendet werden können. Diese Variable wird für Abwärtskompatibilität mit vorherigen Versionen von Net-.Data zur Verfügung gestellt und ist bei den OS/390- oder OS/400-Versionen von Net.Data nicht verfügbar.

Syntax:

```
DTW_VARIABLE_SCOPE [=] LOCAL|GLOBAL
```

Dabei gilt Folgendes:

LOCAL
Gibt an, dass lokale Variablen nur lokal gültig sind. Dieses Verhalten wurde bei Net.Data Version 2.0 eingeführt und ist die Standardeinstellung.

GLOBAL
Gibt an, dass lokale Variablen außerhalb des Funktionsblocks, in dem sie erstellt wurden, verwendet werden können. Diese Variable wird für Abwärtskompatibilität mit früheren Versionen von Net.Data zur Verfügung gestellt; LOCAL wird als Einstellung empfohlen.

Pfadkonfigurationsanweisungen

Net.Data ermittelt die Speicherposition der Dateien und ausführbaren Programme, die von Net.Data-Makros verwendet werden, anhand der Einstellungen der Pfadkonfigurationsanweisungen. Es gibt folgende Pfadanweisungen:

- „DTW_ATTACHMENT_PATH" auf Seite 25
- „EXEC_PATH" auf Seite 25
- „FFI_PATH" auf Seite 26
- „INCLUDE_PATH" auf Seite 26
- „MACRO_PATH" auf Seite 28

Diese Pfadanweisungen geben mindestens ein Verzeichnis an, das Net.Data durchsucht, wenn es versucht, Makros, ausführbare Dateien, Textdateien, LOB-Dateien und Kopfdateien zu lokalisieren. Die benötigten Pfadanweisungen hängen von dem Net.Data-Leistungsspektrum ab, das Ihre Makros verwenden.

Aktualisierungsrichtlinien:

Für die Pfadanweisungen gelten einige allgemeine Richtlinien. Ausnahmen werden in den Beschreibungen der einzelnen Pfadanweisungen angemerkt.

- Trennen Sie die einzelnen Verzeichnisse in der Pfadanweisung durch ein Semikolon (;).
- Jede Pfadanweisung kann mehrere Pfade angeben, mit Ausnahme der Anweisung HTML_PATH, für die nur eine Pfadanweisung zulässig ist. Pfade werden von links nach rechts in der angegebenen Reihenfolge durchsucht. Durch die Möglichkeit zur Angabe mehrerer Pfade können Sie Ihre Dateien in mehreren Verzeichnissen verwalten. Sie können zum Beispiel jede Ihrer Webanwendungen in einem separaten Verzeichnis ablegen.
- Es wird empfohlen, absolute Pfadanweisungen zu verwenden.

In den folgenden Abschnitten werden der Zweck und die Syntax jeder Pfadanweisung beschrieben und Beispiele gültiger Pfadanweisungen gegeben. Die Beispiele können sich je nach Betriebssystem und Konfiguration von Ihrer Anwendung unterscheiden.

DTW_ATTACHMENT_PATH

Diese Pfadkonfigurationsanweisung gibt den Pfad zum Lokalisieren von Anlagen an, die mit Hilfe von DTW_SENDMAIL gesendet werden sollen.

Syntax:

```
DTW_ATTACHMENT_PATH [=] path
```

Beispiel:

```
DTW_ATTACHMENT_PATH /usr/lpp/internet/server_root/pub/upload
```

EXEC_PATH

Diese Pfadkonfigurationsanweisung gibt mindestens ein Verzeichnis an, das Net.Data nach einem externen Programm durchsucht, das über die EXEC-Anweisung oder eine ausführbare Variable aufgerufen wird. Die Reihenfolge der Verzeichnisse in der Pfadanweisung legt die Reihenfolge fest, in der Net-.Data nach den Verzeichnissen sucht. Wenn das Programm gefunden wird, wird der Name des externen Programms an die Pfadangabe angefügt. Der daraus resultierende, vollständig qualifizierte Dateiname wird zur Ausführung an die Sprachumgebung übergeben.

Syntax:

```
EXEC_PATH [=] path1;path2;...;pathn
```

Beispiel: Das folgende Beispiel zeigt die Anweisung EXEC_PATH in der Initialisierungsdatei und die Anweisung EXEC im Makro, das das externe Programm aufruft.

Net.Data-Initialisierungsdatei:

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

Net.Data-Makro:

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

Wenn die Datei `MyFunction.cmd` im Verzeichnis `/usr/lpp/netdata/prgms` gefunden wird, ist der qualifizierte Name des Programms `/usr/lpp/netdata/prgms/myFunction.cmd`.

Wenn die Datei in den in der Anweisung EXEC_PATH angegebenen Verzeichnissen nicht gefunden wird, gilt Folgendes:

- Wenn der angegebene Pfad absolut ist, sucht Net.Data die Datei im angegebenen Pfad. Beispiel: Die folgende URL-Adresse wird übergeben:

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

  Net.Data sucht dann die Datei im Verzeichnispfad `/u/user1/prgms/myFunction.cmd`.
- Wenn der angegebene Pfad relativ ist, sucht Net.Data die Datei im aktuellen Arbeitsverzeichnis. Beispiel: Die folgende URL-Adresse wird übergeben:

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

  Die Datei `myFunction.cmd` wurde in keinem der in EXEC_PATH angegebenen Verzeichnisse gefunden. Net.Data versucht dann, die Datei im aktuellen Arbeitsverzeichnis zu finden.

FFI_PATH

Diese Pfadkonfigurationsanweisung gibt mindestens ein Verzeichnis an, in dem Net.Data in der angegebenen Reihenfolge nach einer unstrukturierten Textdatei sucht, auf die durch die FFI-Funktion (FFI - Flat File Interface - Schnittstelle für unstrukturierte Dateien) verwiesen wird.

Syntax:

```
FFI_PATH [=] path1;path2;...;pathn
```

Beispiel: Das folgende Beispiel zeigt eine Anweisung FFI_PATH in der Initialisierungsdatei.

Net.Data-Initialisierungsdatei:

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

Wenn die FFI-Sprachumgebung aufgerufen wird, sucht Net.Data im in der Anweisung FFI_PATH angegebenen Pfad.

Da die Anweisung FFI_PATH verwendet wird, um die nicht in der Pfadanweisung aufgeführten Dateien zu sichern, gelten für nicht gefundene FFI-Dateien besondere Regeln. Weitere Informationen finden Sie im Abschnitt über in FFI integrierte Funktionen im Handbuch *Net.Data Reference*.

INCLUDE_PATH

Diese Pfadkonfigurationsanweisung gibt mindestens ein Verzeichnis an, das Net.Data durchsucht, und zwar in der angegebenen Reihenfolge, um eine in einer INCLUDE-Anweisung in einem Net.Data-Makro angegebene Datei zu finden. Wenn Net.Data die Datei findet, hängt es den Namen der Kopfdatei an die Pfadangabe an, um den qualifizierten Kopfdateinamen zu erstellen.

Syntax:

```
INCLUDE_PATH [=] path1;path2;...;pathn
```

Beispiel 1: Das folgende Beispiel zeigt sowohl die Anweisung INCLUDE-_PATH in der Initialisierungsdatei als auch die Anweisung INCLUDE, die die Kopfdatei angibt.

Net.Data-Initialisierungsdatei:

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

Net.Data-Makro:

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

Wenn die Datei `myInclude.txt` im Verzeichnis `/u/user1/includes` gefunden wird, ist der vollständig qualifizierte Name der Kopfdatei `/u/user1/includes/myInclude.txt`.

Beispiel 2: Das folgende Beispiel zeigt die Anweisung INCLUDE_PATH und eine Kopfdatei mit einem Unterverzeichnisnamen.

Net.Data-Initialisierungsdatei:

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

Net.Data-Makro:

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

Die Kopfdatei wird in den Verzeichnissen `/u/user1/includes/OE` und `/usr/lpp/netdata/includes/OE` gesucht. Wenn die Datei im Verzeichnis `/usr/lpp/netdata/includes/OE` gefunden wird, ist der vollständig qualifizierte Name der Kopfdatei `/usr/lpp/netdata/includes/OE/oeheader.inc`.

Wenn die Datei in den in der Anweisung INCLUDE_PATH angegebenen Verzeichnissen nicht gefunden wird, gilt Folgendes:

- Wenn der angegebene Pfad absolut ist, sucht Net.Data die Datei im angegebenen Pfad. Beispiel: Die folgende URL-Adresse wird übergeben:

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

  Net.Data sucht dann die Datei im Verzeichnispfad `/u/user1/includes/oeheader.inc`.
- Wenn der angegebene Pfad relativ ist, sucht Net.Data die Datei im aktuellen Arbeitsverzeichnis. Beispiel: Die folgende URL-Adresse wird übergeben:

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

Die Datei myFunction.cmd wurde in keinem der in INCLUDE_PATH angegebenen Verzeichnisse gefunden. Net.Data versucht dann, die Datei im aktuellen Arbeitsverzeichnis zu finden.

MACRO_PATH

Diese Pfadkonfigurationsanweisung gibt die Verzeichnisse an, die Net.Data nach Net.Data-Makros durchsucht. Zum Beispiel wird durch Angabe der folgenden URL-Adresse das Net.Data-Makro mit dem Pfad und Dateinamen /macro/sqlm.dtw angefordert:

```
http://server/cgi-bin/db2www/macro/sqlm.dtw/report
```

Syntax:

```
MACRO_PATH [=] path1;path2;...;pathn
```

Das Gleichheitszeichen (=) ist optional, wie durch eckige Klammern angegeben.

Net.Data fügt den Pfad /macro/sqlm.d2w/report an die Pfade in der Konfigurationsanweisung MACRO_PATH von links nach rechts an, bis das Net.Data-Makro gefunden wird. Wird das Makro nicht gefunden, führt Net-.Data das Makro, das für die Konfigurationsvariable DTW_DEFAULT_MACRO definiert ist, aus oder gibt einen Fehler aus. Informationen zum Aufrufen von Net.Data-Makros finden Sie in „Kapitel 4. Aufrufen von Net.Data" auf Seite 79.

Beispiel: Das folgende Beispiel zeigt die Anweisung MACRO_PATH in der Initialisierungsdatei und die zugehörige Verbindung (Link), die Net.Data aufruft.

Net.Data-Initialisierungsdatei:

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

HTML-Verbindung (Link):

```
<a href="http://server/cgi-bin/db2www/query.dtw/input">Submit another query.</a>
```

Wenn die Datei *query.dtw* im Verzeichnis /u/user1/macros gefunden wird, ist der vollständig qualifizierte Pfad /u/user1/macros/query.dtw.

Wenn die Datei in den in der Anweisung MACRO_PATH angegebenen Verzeichnissen nicht gefunden wird, gilt Folgendes:

- Wenn der angegebene Pfad absolut ist, sucht Net.Data die Datei im angegebenen Pfad. Beispiel: Die folgende URL-Adresse wird übergeben:

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

Net.Data sucht dann die Datei im Verzeichnispfad `/u/user1/macros/myfile.txt`.

- Wenn der angegebene Pfad relativ ist, sucht Net.Data die Datei in allen Verzeichnissen, beginnend mit dem Stammverzeichnis (`/`). Beispiel: Die folgende URL-Adresse wird übergeben:

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

  Die Datei `myfile.txt` wurde in keinem der in MACRO_PATH angegebenen Verzeichnisse gefunden. Net.Data versucht dann, die Datei im Stammverzeichnis (`/`) zu finden: `/myfile.txt`

Umgebungskonfigurationsanweisungen

Eine Anweisung ENVIRONMENT konfiguriert eine Sprachumgebung. Eine Sprachumgebung ist eine Net.Data-Komponente, mit der Net.Data auf eine Datenquelle, wie zum Beispiel eine DB2-Datenbank, zugreift oder ein in einer Sprache wie REXX geschriebenes Programm ausführt. Net.Data stellt eine Reihe von Sprachumgebungen bereit sowie eine Schnittstelle, mit der Sie Ihre eigenen Sprachumgebungen erstellen können. Diese Sprachumgebungen werden in „Kapitel 6. Verwenden der Sprachumgebungen" auf Seite 151 beschrieben, und die Schnittstelle für Sprachumgebungen wird im Handbuch *Net.Data Language Environment Interface Reference* erörtert.

Net.Data erfordert, dass eine Anweisung ENVIRONMENT für eine bestimmte Sprachumgebung vorhanden ist, bevor Sie diese Sprachumgebung aufrufen können.

Sie können Variablen einer Sprachumgebung zuordnen, indem Sie die Variablen als Parameter in der Anweisung ENVIRONMENT angeben. Net.Data übergibt die in einer Anweisung ENVIRONMENT angegebenen Parameter implizit als Makrovariablen an die Sprachumgebung. Sie können im Makro den Wert eines Parameters ändern, der in einer Anweisung ENVIRONMENT angegeben ist, indem Sie entweder der Variablen mit der Funktion DTW_ASSIGN() einen Wert zuordnen oder die Variable in einem DEFINE-Abschnitt definieren.

Wichtig: Wird eine Variable in einem Makro definiert, aber nicht in der Anweisung ENVIRONMENT angegeben, wird die Makrovariable nicht an die Sprachumgebung übergeben.

Zum Beispiel kann ein Makro eine Variable DATABASE definieren, um den Namen einer Datenbank anzugeben, in der eine SQL-Anweisung innerhalb der Funktion DTW_SQL ausgeführt werden soll. Der Wert von DATABASE muss an die SQL-Sprachumgebung (DTW_SQL) übergeben werden, damit die SQL-Sprachumgebung die Verbindung zur angegebenen Datenbank herstellen

kann. Zum Übergeben der Variablen an die Sprachumgebung müssen Sie die Variable DATABASE der Parameterliste in der Umgebungsanweisung für DTW_SQL hinzufügen.

Die Net.Data-Beispielinitialisierungsdatei geht beim Anpassen der Einstellungen der Anweisungen für die Net.Data-Umgebungskonfiguration von bestimmten Annahmen aus. Diese Annahmen sind für Ihre Umgebung eventuell nicht zutreffend. Ändern Sie die Anweisungen Ihrer Umgebung entsprechend.

Gehen Sie wie folgt vor, um eine Anweisung ENVIRONMENT hinzuzufügen oder zu aktualisieren:

Anweisungen ENVIRONMENT haben die folgende Syntax:

```
ENVIRONMENT(type) library_name (parameter_list, ...) [CLIETTE "cliette_name"]
```

Parameter:

- *type*

  Der Name, durch den Net.Data diese Sprachumgebung einem FUNCTION-Block zuordnet, der in einem Net.Data-Makro definiert ist. Sie müssen die Art der Sprachumgebung in einer FUNCTION-Blockdefinition angeben, um die Sprachumgebung zu definieren, in der Net.Data die Funktion ausführen soll.

- *library_name*

  Der Name der DLL-Datei oder gemeinsam benutzten Bibliothek mit den Schnittstellen für Sprachumgebungen, die Net.Data aufruft.

  – Unter AIX wird der Name der gemeinsam benutzten Bibliothek mit der Erweiterung *.o* angegeben.
  – Unter HP-UX wird der Name der gemeinsam benutzten Bibliothek mit der Erweiterung *.sl* angegeben.
  – Unter SUN und LINUX wird der Name der gemeinsam benutzten Bibliothek mit der Erweiterung *.so* angegeben.
  – Unter OS/2 und Windows NT wird der Name der gemeinsam benutzten Bibliothek mit der Erweiterung *.dll* angegeben.

- *parameter_list*

  Die Liste der Parameter, die bei jedem Funktionsaufruf neben den Parametern, die in der FUNCTION-Blockdefinition angegeben sind, an die Sprachumgebung übergeben werden.

  Definieren Sie die Variable im Makro, um die Variablen in der Parameterliste festzulegen und zu übergeben.

  Sie müssen diese Parameter als Konfigurationsvariablen oder als Variablen in Ihrem Makro definieren, bevor Sie eine Funktion ausführen können, die

von der Sprachumgebung verarbeitet wird. Das folgende Beispiel gibt die Variablen in der Anweisung ENVIRONMENT an:

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

Wenn eine Funktion ihre Ausgabeparameter ändert, behalten die Parameter ihre geänderten Werte nach der Beendigung der Funktion bei.

- *cliette_name*

  Der Name der Cliette. *cliette_name* kann sich auf die Cliette für die Sprachumgebung der Java-Anwendung beziehen oder eine Datenbank-Cliette angeben. Der Parameter *cliette_name* wird zusammen mit dem Schlüsselwort CLIETTE verwendet, und beide sind nur bei Direktverbindungen gültig. `CLIETTE` und *`cliette_name`* sind optional und können nur für Sprachumgebungen von Datenbanken und Java-Anwendungen verwendet werden.

  Java-Anwendungs-Cliette

  Dieser Cliette-Name gibt die Sprachumgebung der Java-Anwendung an.

  Syntax:

  ```
  CLIETTE "DTW_JAVAPPS"
  ```

  Datenbank-Cliette

  Dieser Cliette-Name gibt eine Cliette an, die einer Datenbank zugeordnet ist.

  Syntax:

  ```
  CLIETTE "type:db_name"
  ```

  Parameter:

  type Die der Cliette zugeordnete Sprachumgebung der Datenbank. Eine Liste der gültigen Typen finden Sie auf Seite 63.

  db_name

  Der Datenbank-Cliette-Name. Dieser Name entspricht häufig dem Namen der Datenbank, der die Cliette zugeordnet ist, wie zum Beispiel MYDBASE, kann aber auch ein anderer Name sein. *db_name* ist optional, wenn Sie in der Oracle-Sprachumgebung arbeiten.

Wenn Net.Data die Initialisierungsdatei verarbeitet, werden die DLL-Dateien bzw. die gemeinsam benutzten Bibliotheken der Sprachumgebung nicht geladen. Net.Data lädt eine DLL-Datei bzw. eine gemeinsam benutzte Bibliothek, wenn es das erste Mal eine Funktion ausführt, die die betreffende Sprachumgebung angibt. Die DLL-Datei bzw. gemeinsam benutzte Bibliothek bleibt dann so lange geladen, wie Net.Data geladen ist.

Beispiel: Anweisungen ENVIRONMENT für von Net.Data bereitgestellte Sprachumgebungen

Beim Anpassen der Anweisungen ENVIRONMENT für Ihre Anwendung müssen Sie den Anweisungen ENVIRONMENT die Variablen hinzufügen, die von Ihrer Initialisierungsdatei an die Sprachumgebung übergeben werden müssen, oder die Net.Data-Makroprogrammierer zum Festlegen oder Überschreiben in ihren Makros benötigen.

```
ENVIRONMENT (DTW_SQL)  /net.data/lib/dtwsql.so  ( IN DATABASE, LOGIN, PASSWORD,
  TRANSACTION_SCOPE, ALIGN, START_ROW_NUM, DTW_SET_TOTAL_ROWS)
  CLIETTE "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 )
CLIETTE "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 )
```

Wichtiger Hinweis: Jede Anweisung ENVIRONMENT muss in einer separaten Zeile stehen.

Definieren der Net.Data-Sprachumgebungen

Nach einer Änderung von Konfigurationsvariablen und Konfigurationsanweisungen ENVIRONMENT für die Net.Data-Sprachumgebungen sind einige zusätzliche Konfigurationsschritte erforderlich, damit die folgenden Sprachumgebungen ordnungsgemäß funktionieren. In den folgenden Abschnitten werden die zum Definieren der Sprachumgebungen erforderlichen Schritte beschrieben:

- „Definieren der Java-Sprachumgebung mit Cliette"
- „Definieren der Oracle-Sprachumgebung" auf Seite 33

Definieren der Java-Sprachumgebung mit Cliette

Werden Verbindungen zur virtuellen Java-Maschine mit Hilfe von Direktverbindungen verwaltet, sind für die Java-Sprachumgebung einige zusätzliche Konfigurationsschritte erforderlich, bevor Sie Funktionen von einem Makro aus aufrufen können:

1. Erstellen Sie eine Stapeldatei zum Starten der Java-Anwendung. Net.Data verwendet diese Datei zum Starten der virtuellen Java-Maschine, die Ihre Java-Funktion ausführt. Die Stapeldatei muss die Java-Klassenpfadanweisung enthalten, um sicherzustellen, dass die erforderlichen Java-

Pakete (die Standardpakete und die anwendungsspezifischen Pakete) gefunden werden. Zum Beispiel enthält die Stapeldatei `launchjv.bat` folgende Java-Klassenpfadanweisung:

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

2. Definieren Sie für die Java-Sprachumgebung eine Cliette in der Konfigurationsdatei für Direktverbindungen `dtwcm.cnf`. Geben Sie mit der Konfigu-
rationsvariablen EXEC_NAME einen Stapeldateinamen für die Cliette an. Im folgenden Beispiel wird der Java-Cliette-Name als DTW_JAVAPPS definiert, und die Konfigurationsvariable EXEC_NAME wird auf den Namen der Stapeldatei gesetzt, d. h. `launchjv.bat`:

```
CLIETTE DTW_JAVAPPS{
MIN_PROCESS=1

MAX_PROCESS=1

EXEC_NAME=launchjv.bat

}
```

   Wenn Sie Net.Data Connection Manager starten, startet Net.Data die in der Konfigurationsdatei angegebene Java-Cliette. Die Cliette wird zum Verarbeiten der Java-Sprachumgebungsanforderungen Ihrer Net.Data-Makroanwendungen zur Verfügung gestellt.

3. Aktualisieren Sie die ENVIRONMENT-Anweisung DTW_JAVAPPS in der Net.Data-Initialisierungsdatei `db2www.ini`, indem Sie der Anweisung den Cliette-Namen hinzufügen. Beispiel:

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

Definieren der Oracle-Sprachumgebung

Führen Sie die folgenden Schritte aus, um von einem Net.Data-Makro aus auf Oracle-Datenbanken zuzugreifen:

1. Stellen Sie wie folgt sicher, dass die erforderlichen Oracle-Komponenten installiert sind und ordnungsgemäß funktionieren:
   a. Installieren Sie SQL*Net auf der Maschine, auf der Net.Data installiert ist, sofern dieses Produkt nicht bereits installiert ist. Weitere Informationen hierzu finden Sie an folgender URL-Adresse:

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

   b. Prüfen Sie, ob die Oracle-Funktion *tnsping* mit derselben Sicherheitsberechtigung verwendet werden kann, die auch für Ihren Webserver eingesetzt wird. Melden Sie sich dazu mit der Benutzer-ID Ihres Webservers an, und geben Sie Folgendes ein:

      ```
      tnsping oracle-instance-name
      ```

Dabei gilt Folgendes: *oracle-instance-name* ist der Name des Oracle-Systems, auf das Ihre Net.Data-Makros zugreifen.

Sie sind eventuell nicht in der Lage, die Funktion *tnsping* unter Windows NT zu prüfen, wenn Ihr Webserver als Windows NT-Service ausgeführt wird. Überspringen Sie in diesem Fall diesen Schritt.

c. Prüfen Sie, ob auf die Oracle-Tabellen mit der auch vom Webserver verwendeten Sicherheitsberechtigung zugegriffen werden kann. Geben Sie dazu mit Hilfe des Zeilenbefehl-Tools SQL*Plus eine SQL-Anweisung SELECT mit der Berechtigung Ihres Webservers ein, um auf eine Oracle-Tabelle zuzugreifen. Beispiel:

```
SELECT * FROM tablename
```

Sie sind eventuell nicht in der Lage, den Tabellenzugriff unter Windows NT zu prüfen, wenn Ihr Webserver als Windows NT-Service ausgeführt wird. Überspringen Sie in diesem Fall diesen Schritt.

Fehlerbehebung: Setzen Sie den Vorgang nicht fort, wenn die obigen Schritte fehlschlagen. Wenn einer der Schritte fehlschlägt, überprüfen Sie Ihre Oracle-Konfiguration.

2. Stellen Sie sicher, dass die Oracle-Umgebungsvariablen in Ihrem Webserverprozess ordnungsgemäß eingestellt sind.
   - Nehmen Sie unter AIX die folgenden Zeilen in die Datei `/etc/environment` oder in die Datei `.profile` für die Benutzer-ID, unter der der Webserver ausgeführt wird, auf:

     ```
     ORACLE_SID=oracle-instance-name
     ORACLE_HOME=oracle-runtime-library-directory
     ```

   - Fügen Sie unter Windows NT mit der Systemsteuerung für Systemeigenschaften die folgenden Umgebungsvariablen hinzu:

     ```
     ORACLE_SID=oracle-instance-name
     ORACLE_HOME=oracle-runtime-library-directory
     ```

   Hinweis: Sie benötigen je nach den zu verwendenden Oracle-Funktionen, wie Unterstützung in der Landessprache und zweiphasige Festschreibung, eventuell zusätzliche Zeilen für andere Oracle-Umgebungsvariablen. Weitere Informationen zu diesen Umgebungsvariablen finden Sie in der Oracle-Verwaltungsdokumentation.

3. Testen Sie die Verbindung von Net.Data zu Oracle. Geben Sie in Ihrem Net.Data-Makro die entsprechenden Werte für die Variablen LOGIN und PASSWORD an. Definieren Sie die Net.Data-Variable DATABASE beim Zugriff auf Oracle-Datenbanken *nicht*. Im folgenden Beispiel werden Verbindungsanweisungen in einem Makro illustriert:

   ```
   %DEFINE LOGIN=user_ID@remote-oracle-instance-name
   %DEFINE PASSWORD=password
   ```

Lokale Oracle-Exemplare:

Wenn Sie nur auf das lokale Oracle-Exemplar zugreifen, geben Sie *remote-oracle-instance-name* nicht als Teil der Anmeldebenutzer-ID an. Siehe folgendes Beispiel:

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

Direktverbindung:

Wenn Sie die Direktverbindung verwenden, können Sie LOGIN und PASSWORD in der Konfigurationsdatei für Direktverbindungen angeben. Dies wird aus Sicherheitsgründen jedoch nicht empfohlen. Beispiel:

```
CLIETTE DTW_ORA:{
MIN_PROCESS=1
MAX_PROCESS=3
EXEC_NAME=./dtwora8
DATABASE=not_used
LOGIN=userid@remote_oracle_instance_name
PASSWORD=password}
```

Hinweis: Geben Sie die Variable DATABASE für Oracle nicht an.

4. Testen Sie Ihre Konfiguration, indem Sie ein CGI-Shell-Script ausführen, um sicherzustellen, dass von Ihrem Webserver auf das Oracle-Exemplar zugegriffen werden kann. Siehe folgendes Beispiel:

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

Sie können alternativ *tnsping* direkt von einem Net.Data-Makro aus durchführen. Siehe folgendes Beispiel:

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

Fehlerbehebung:

Wenn der Prüfungsschritt fehlschlägt, überprüfen Sie, ob alle vorherigen Schritte erfolgreich waren, indem Sie die folgenden Punkte prüfen:

- Überprüfen Sie Ihre Oracle-Konfiguration.
- Prüfen Sie, ob die Oracle-Umgebungsvariablensyntax korrekt ist und ob keine Variablen fehlen.
- Überprüfen Sie die Oracle-Verbindung, indem Sie sicherstellen, dass Sie die richtige Benutzer-ID und das richtige Kennwort eingegeben haben.

Wenn die Prüfung weiterhin fehlschlägt, wenden Sie sich an den IBM Kundendienst.

Beispiel:

Nach dem Beenden der Zugriffsprüfungsschritte können Sie Aufrufe in der Oracle-Sprachumgebung mit Funktionen im Makro tätigen. Siehe folgendes Beispiel:

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

Konfigurieren der Direktverbindung

Die Direktverbindung verwaltet Datenbank- und Java-Anwendungsverbindungen zur Leistungssteigerung von Net.Data unter Windows NT, OS/2, AIX und Sun Solaris. Durch die Verwendung von Connection Manager und Cliettes, d. h. Prozessen, die offene Verbindungen verwalten, beseitigt die Direktverbindung den Systemaufwand für die Verbindungsherstellung zu einer Datenbank bzw. für das Starten einer virtuellen Java-Maschine.

Die Direktverbindung verwendet eine Konfigurationsdatei (`dtwcm.cnf`), um festzustellen, welche Cliettes gestartet werden müssen. Sie enthält Verwaltungsinformationen und Definitionen für jede der mit der Direktverbindung verwendeten Cliettes. Informationen zur Direktverbindung finden Sie in „Verwalten von Verbindungen” auf Seite 200.

Die in Abb. 6 auf Seite 37 gezeigte Beispielkonfigurationsdatei enthält die folgenden Arten von Informationen:

- Anschlussinformationen für Connection Manager
- SQL-Cliette-Informationen für eine DB2-Verbindung
- Cliette-Informationen zu Java-Anwendungen

```
 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=stlpwd
12  }
13
14  CLIETTE DTW_JAVAPPS{
15  MIN_PROCESS=1
16  MAX_PROCESS=5
17  EXEC_NAME=./javaapp
18  }
```

- Die Zeilen 1 - 3 sind für die Konfigurationsdatei erforderlich und definieren eindeutige, mit der Direktverbindung zu verwendende Anschlussnummern.
- Die Zeilen 5 - 12 definieren alle Datenbank-Cliettes, wobei der Cliette-Name, die Anzahl der auszuführenden Prozesse, der Datenbankname und die Cliette-Exec-Datei angegeben werden. Sie können zusätzliche Informationen, wie eine Benutzer-ID und ein Kennwort zur Verbindungsherstellung zu einer DB2-Datenbank, angeben.
- Die Zeilen 14 - 18 definieren alle Cliettes für Java-Anwendungen, wobei der Cliette-Name, die Anzahl der auszuführenden Prozesse, die eindeutigen Anschlussnummern und die Cliette-Exec-Datei angegeben werden.

Abbildung 6. Die Konfigurationsdatei für Direktverbindungen

Vorbereitung: Lesen Sie den Abschnitt mit Hinweisen und Tips nach den folgenden Schritten, bevor Sie die Konfigurationsdatei für Direktverbindungen anpassen.

Konfigurieren der Anschlüsse für Direktverbindungen:

Der Wert, den Sie für MAIN_PORT auswählen, ist die Anschlussnummer, die zuerst verwendet wird. Die von der Direktverbindung verwendbaren Anschlussnummern können mit Hilfe der Einstellung von MAIN_PORT und des Werts für MAX_PROCESSES jeder Cliette berechnet werden. Nach dem Laden ordnet die Direktverbindung Anschlüsse zu, wobei mit der in MAIN-_PORT angegebenen Nummer begonnen wird. Die Zuordnung endet bei dem kumulativen Wert für MIN_PROCESSES. Dann werden Anschlüsse nach Bedarf geladen, bis der Wert für MAX_PROCESSES erreicht ist. Die maximale Anzahl der verwendeten Anschlüsse ist die Summe der Einstellungen von MAX_PROCESSES.

In der Konfiguration in Abb. 6 lauten die zugeordneten Anschlussnummern beispielsweise 7100, 7101 und 7102 und dann nach Bedarf bis zu 7110.

Wichtig:

- Wenden Sie sich an Ihren Systemadministrator, um sicherzustellen, dass die zu verwendenden Anschlussnummern verfügbar sind.
- Stellen Sie sicher, dass der Wert von MAIN_PORT mit dem Wert von DTW-_CM_PORT in der Net.Data-Initialisierungsdatei übereinstimmt.

Konfigurieren von Datenbank-Cliettes:

1. Geben Sie die Cliette-Umgebungsanweisung ein.

   ```
   CLIETTE type:db_name
   ```

   Parameter:

   type Der Name, der eine Sprachumgebung einer Cliette zuordnet. Eine Liste der gültigen Typen finden Sie auf Seite 63.

   db_name
   Der Datenbank-Cliette-Name, der häufig dem Namen der Datenbank entspricht, der die Cliette zugeordnet ist, wie zum Beispiel MYDBASE; *db_name* kann aber auch ein anderer Name sein. *db_name* ist optional, wenn Sie in der Oracle-Sprachumgebung arbeiten.

2. Legen Sie Werte für MIN_PROCESS und MAX_PROCESS fest. MIN_PROCESS gibt die Anzahl der zu startenden Prozesse beim Start von Connection Manager an. Wenn danach zusätzliche gleichzeitige Anforderungen ankommen, startet Connection Manager weitere Cliettes. Bei Bedarf wird jeweils eine Cliette hinzugefügt, bis der für MAX_PROCESS angegebene Wert erreicht ist.

   Geben Sie die Anweisungen MIN_PROCESS und MAX_PROCESS wie folgt ein:

   ```
   MIN_PROCESS=min_num
   MAX_PROCESS=max_num
   ```

   Parameter:

   min_num
   Die Anzahl der zu startenden Cliette-Prozesse beim Start von Connection Manager. Für diese Anzahl von Cliettes müssen Sie über genügend eindeutige Anschlussnummern verfügen.

   max_num
   Die maximale Anzahl von Cliettes, die gleichzeitig ausgeführt werden können. Für diese Anzahl von Cliettes müssen Sie über genügend eindeutige Anschlussnummern verfügen.

3. Geben Sie den Namen der ausführbaren Cliette-Datei wie folgt an:

   ```
   EXEC_NAME=./dtwcdbtypeid
   ```

Dabei gilt Folgendes: *dbtypeid* ist die Kennung der Datenbankart. Die Namen der gültigen ausführbaren Dateien finden Sie in Tabelle 6:

Tabelle 6. Namen ausführbarer Cliette-Dateien

Cliette-Beschreibung	Cliette-Typ	Namen		Plattformverfügbarkeit					
		UNIX	Windows NT oder OS/2	AIX	NT	OS/2	HP	SUN	PTX
DB2-Prozess-Cliette	DTW_SQL	dtwcdb2	dtwcdb2.exe	J	J	J	J	J	N
ODBC-Prozess-Cliette	DTW_ODBC	dtwcodbc	dtwcodbc.exe	J	J	N	N	N	N
Oracle-Prozess-Cliette	DTW_ORA	dtwcora	dtwcora.exe	J	J	N	N	N	N

4. Geben Sie den Namen der Datenbank an, der die Cliette zugeordnet ist:
   ```
   DATABASE=db_name
   ```
   Dabei gilt Folgendes: *db_name* ist der Name der Datenbank, der die Cliette zugeordnet ist, zum Beispiel MYDBASE.
5. Optional: Ersetzen Sie die Standardwerte für die Variablen LOGIN und PASSWORD durch *USE_DEFAULT, so dass Net.Data die gleiche Benutzer-ID verwendet, über die Connection Manager gestartet wurde, um die Verbindung zur DB2-Datenbank herzustellen. Durch Angabe dieser Standardwerte brauchen Sie diese Informationen nicht in die Konfigurationsdatei zu stellen. Ersetzen Sie zum Beispiel die Zeilen 14 und 15 in der Beispielkonfigurationsdatei in Abb. 6 auf Seite 37 durch folgende Zeilen:
   ```
   LOGIN=*USE_DEFAULT
   PASSWORD=*USE_DEFAULT
   ```
   Hinweis: Wenn Sie in der Konfigurationsdatei mehrere Cliette-Einträge definieren, können Sie verschiedene Anmeldenamen und Kennwörter für eine bestimmte Datenbank angeben.

Konfigurieren von Java-Anwendungs-Cliettes:

1. Geben Sie die Cliette-Umgebungsanweisung ein:
   ```
   CLIETTE DTW_JAVAPPS
   ```
2. Legen Sie Werte für MIN_PROCESS und MAX_PROCESS fest. MIN_PROCESS gibt die Anzahl der zu startenden Prozesse beim Start von Connection Manager an. Wenn danach gleichzeitige Anforderungen ankommen,

startet Connection Manager weitere Cliettes. Bei Bedarf wird jeweils eine Cliette hinzugefügt, bis der für MAX_PROCESS angegebene Wert erreicht ist.

Geben Sie die Anweisungen MIN_PROCESS und MAX_PROCESS wie folgt ein.

```
MIN_PROCESS=min_num
MAX_PROCESS=max_num
```

Parameter:

min_num
: Die Anzahl der gestarteten Cliette-Prozesse beim Start von Connection Manager. Für diese Anzahl von Cliettes müssen Sie über genügend eindeutige Anschlussnummern verfügen.

max_num
: Die maximale Anzahl von zusätzlichen Cliettes, die gleichzeitig ausgeführt werden können. Für diese Anzahl von Cliettes müssen Sie über genügend eindeutige Anschlussnummern verfügen.

Hinweise und Tips zum Konfigurieren von Direktverbindungen:

- Connection Manager verwendet Cliette-Namen, um eine Gruppe von Cliettes eindeutig zu identifizieren.
- Bei Datenbank-Cliettes benötigen Sie eine benannte Gruppe von Cliettes für jede Datenbank, die Sie verwenden wollen. Für Datenbanken, auf die nur selten zugegriffen wird, können Sie die minimale und maximale (MIN und MAX) Anzahl von Cliettes auf den Wert 1 setzen. Alternativ dazu können Sie für MIN den Wert 0 angeben, d. h., Prozesse werden erst bei einer Net.Data-Anforderung gestartet.
- Der NAME der Cliette muss mit dem Cliette-Namen übereinstimmen, auf den in der Anweisung ENVIRONMENT für den Cliette-Typ in der Initialisierungsdatei verwiesen wird. Der Cliette-Name kann Variablen enthalten und muss im Fall von Datenbank-Cliettes den Variablenverweis $(DATABASE) enthalten. Der Standardwert für den Cliette-Namen in der Anweisung ENVIRONMENT ist DTW_SQL:$(DATABASE). Sie können einen Variablenverweis in der Initialisierungsdatei, aber nicht in der Konfigurationsdatei für Direktverbindungen verwenden.

  Die Variable DATABASE wird im Net.Data-Makro definiert. Wenn im Makro eine SQL-Anweisung festgestellt wird, wird der Variablenverweis $(DATABASE) in der Net.Data-Initialisierungsdatei durch den aktuellen Wert von DATABASE ersetzt.

  Mit dieser Methode können Sie auf mehrere Datenbanken zugreifen. Wenn Sie in Ihrem Net.Data-Makro auf drei Datenbanken (z. B. D1, D2 und D3) zugreifen wollen und die Initialisierungsdatei die Standardzeile CLIETTE

"DTW_SQL:$(DATABASE)" enthält, muss Ihre Konfigurationsdatei für die Direktverbindung drei Abschnitte enthalten. Beispiel:

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

- Prozesse werden gestartet, aber nicht gestoppt. Wenn Sie die maximale Anzahl von Prozessen auf den Wert M setzen und zu einem beliebigen Zeitpunkt M Prozesse gleichzeitig verwendet werden, bleiben diese Prozesse so lange aktiv, bis Sie Connection Manager schließen. Aus diesem Grund sollten Sie den Wert für MAX_PROCESS nicht zu hoch einstellen, da andernfalls Ihre Systemressourcen durch das Starten selten verwendeter Prozesse aufgebraucht werden.

  Empfehlung: Probieren Sie verschiedene Werte für MIN_PROCESS und MAX_PROCESS aus, bis Sie die Werte gefunden haben, die sich für Ihr System optimal eignen. Wenn Connection Manager mehr Anforderungen als die angegebene maximale Anzahl empfängt, wird die letzte Anforderung in eine Warteschlange gestellt, bis eine Cliette die Verarbeitung beendet. Wenn eine Cliette zur Verfügung steht, wird die in die Warteschlange gestellte Anforderung verarbeitet. Dieses Stellen von Anforderungen in eine Warteschlange ist für den Anwendungsbenutzer transparent.
- Sie können denselben Cliette-Typ für verschiedene benannte Abschnitte verwenden. Beispielsweise verwenden alle für DB2-Datenbanken relevanten Abschnitte in der Konfigurationsdatei denselben Cliette-Typ. Es ist nicht möglich, zwei Abschnitte mit demselben Namen zu verwenden.

Wenn Sie CGI verwenden und nur einige Datenbanken die Direktverbindung verwenden sollen, listen Sie die gewünschten Datenbanken einfach in der Konfigurationsdatei auf. Wenn Net.Data bei der Verarbeitung eines Net.Data-Makros eine SQL-Funktion findet, fordert das Programm eine bestimmte Cliette von Connection Manager an. Steht der gewünschte Cliette-Typ nicht zur Verfügung, gibt Connection Manager die Nachricht aus, dass keine Cliette verfügbar ist (NO_CLIETTE_AVAIL). Net.Data verarbeitet die Anforderung stattdessen mit einer DLL-Version.

Konfigurieren des Connection Manager-Services für den automatischen Start:

Unter Windows NT können Sie angeben, dass Connection Manager nicht von der Befehlszeile, sondern als ein Windows NT-Service gestartet werden soll. Die Ausführung von Connection Manager als ein Windows NT-Service ermöglicht den automatischen Start von Connection Manager bei jedem Start der Maschine.

Wichtig: Starten Sie Connection Manager von der Befehlszeile aus, bevor Sie seinen automatischen Start definieren, um sicherzustellen, dass die Konfigurationsdatei für Direktverbindungen korrekt ist.

- Wählen Sie in der NT-Task-Leiste **Start -> Einstellungen -> Systemsteuerung -> Dienste** aus.
- Wählen Sie **Net.Data Connection Manager** aus, und klicken Sie anschließend den Knopf **Starten** an.
- Wählen Sie **Startart** aus, und klicken Sie anschließend **OK** an.

Konfigurieren des Webservers zur Verwendung mit CGI

Common Gateway Interface (CGI) ist eine Standardschnittstelle, über die ein Webserver ein Anwendungsprogramm wie Net.Data aufrufen kann. Dadurch, dass Net.Data CGI unterstützt, können Sie es mit Ihrem bevorzugten Webserver verwenden.

Konfigurieren Sie Net.Data so, dass nur jeweils eine Schnittstelle verwendet wird. Wenn Sie beispielsweise den Webserver so konfigurieren, dass Net.Data mit CGI ausgeführt wird, dürfen Sie den Webserver nicht zusätzlich so konfigurieren, dass Net.Data mit einer anderen Schnittstelle ausgeführt wird. Wenn Sie Net.Data später mit einer anderen Schnittstelle ausführen wollen, z. B. FastCGI, müssen Sie den Webserver zur ausschließlichen Verwendung der neuen Schnittstelle rekonfigurieren.

Konfigurieren Sie den Webserver so, dass Net.Data aufgerufen wird. Nehmen Sie dazu Map-, Exec- und Pass-Anweisungen in die HTTP-Konfigurationsdatei auf, die bewirken, dass Net.Data aufgerufen wird.

Empfehlung: Verwalten Sie die Anweisungen innerhalb der HTTP-Konfigurationsdatei in der folgenden Reihenfolge, um zu verhindern, dass Anweisungen ignoriert werden: Map, Exec, Pass. Wenn z. B. die folgende Pass-Anweisung einer Map- oder Exec-Anweisung vorangeht, werden die Map- und Exec-Anweisungen ignoriert:

```
Pass /*
```

Map-Anweisungen

Mit den Map-Anweisungen werden Einträge mit dem Format `/cgi-bin/db2www/*` der Bibliothek zugeordnet, in der sich das Net.Data-Programm auf Ihrem System befindet. (Der Stern (*) am Ende der Zeichenfolge bezieht sich auf alles, was auf die Zeichenfolge folgt.) Es werden sowohl Map-Anweisungen in Kleinbuchstaben als auch solche in Großbuchstaben aufgeführt, da bei den Anweisungen zwischen Groß-/Kleinschreibung unterschieden wird.

Exec-Anweisungen

Mit der Exec-Anweisung können vom Webserver beliebige CGI-Programme in der CGI-Bibliothek ausgeführt werden. Geben Sie die Bibliothek, in der sich das Programm befindet (nicht das Programm selbst), in der Anweisung an.

Einstellungen allgemeiner Webserverparameter

Konfigurieren Sie den Webserver so, dass die von Net.Data benötigten Umgebungsvariablen definiert und/oder übergeben werden. Die Bearbeitung von Umgebungsvariablen wird in Sprachumgebungen und Betriebssystemen unterschiedlich ausgeführt. Die exakte Syntax für Ihre Umgebung und Ihre Plattform finden Sie in Ihrer Webserverdokumentation.

libpath Die Umgebungsvariable LIBPATH sollte den Pfad zu den Verzeichnissen enthalten, in denen sich die gemeinsamen Bibliotheken von Net-.Data oder die DLLs befinden, die in den Anweisungen ENVIRONMENT in der Net.Data-Initialisierungsdatei erscheinen. Wird auf DB2 zugegriffen, sollte die Variable LIBPATH den Pfad zum DB2-Bibliotheksverzeichnis enthalten.

Für Apache und IBM HTTP Web Server:

```
SetEnv LIBPATH /u/mydir/myserver/lib:/u/mydir/myserver:
  /usr/lpp/db2_07_01/lib:/usr/lib
```

oracle_home

Bei Verwendung von Oracle erforderlich. Der Pfad und das Verzeichnis der ausführbaren Oracle-Datenbankdateien.

Für Apache und IBM HTTP Web Server:

```
SetEnv ORACLE_HOME /home.native/oracle/product/8.1.5
```

oracle_sid

Bei Verwendung von Oracle erforderlich. Das Exemplar der Oracle-Datenbank. Sie müssen für Oracle die Direktverbindung verwenden.

Für Apache und IBM HTTP Web Server:

```
SetEnv ORACLE_SID mvpdb2
```

db2instance

Bei Verwendung von DB2 erforderlich. Das Exemplar der DB2-Datenbank.

Für Apache und IBM HTTP Web Server:

```
SetEnv DB2INSTANCE wwwinst
```

REXX_owner_pid

Bei Verwendung von REXX unter AIX erforderlich. Die Leistungsvariable wird mit FastCGI und REXX auf dem Betriebssystem AIX verwendet. Der Standardwert ist 0. Deklarieren Sie diese Variable bei anderen Produkten und Betriebssystemen im Net.Data-Makro. Weitere Informationen zu dieser Variablen finden Sie in „Anhang B. Net.Data für AIX" auf Seite 249.

Für Apache und IBM HTTP Web Server:

```
SetEnv RXQUEUE_OWNER_PID 0
```

lang Die UNIX-Variable für länderspezifische Angaben. Verwenden Sie De_DE für Deutsch.

Für Apache und IBM HTTP Web Server:

```
SetEnv LANG De_DE
```

NLSPATH

Gibt die Verzeichnisposition des Nachrichtenkatalogs an.

Für Apache und IBM HTTP Web Server:

```
SetEnv NLSPATH /usr/lib/nls/msg/%L/%N
```

Konfigurieren von Net.Data für FastCGI

Die Schnittstelle FastCGI ist eine Schnittstelle mit Industriestandard, mit der eine Anwendung ähnlich wie CGI-Anwendungen ausgeführt werden kann, wobei die Prozesse von Anforderung zu Anforderung aktiv bleiben. Sie verbindet die Leistungsstärke anderer Web-API-Programme mit der Anwendungsisolation von CGI. Net.Data kann als ein FastCGI-Prozess auf Apache Webserver und IBM HTTP Server ausgeführt werden. FastCGI wird unter den Betriebssystemen AIX und Sun Solaris unterstützt.

Konfigurieren Sie Net.Data so, dass nur jeweils eine Schnittstelle verwendet wird. Wenn Sie beispielsweise den Webserver so konfigurieren, dass Net.Data mit FastCGI ausgeführt wird, dürfen Sie den Webserver nicht zusätzlich so konfigurieren, dass Net.Data mit einer anderen Schnittstelle ausgeführt wird. Wenn Sie Net.Data später mit einer anderen Schnittstelle ausführen wollen, müssen Sie den Webserver zur ausschließlichen Verwendung der neuen Schnittstelle rekonfigurieren.

Vor der Verwendung von FastCGI müssen Sie Folgendes installieren:

- Apache Web Server 1.2.0
- IBM HTTP Server 1.3.12.0 oder höher
- HTTP FastCGI-Modul

Gehen Sie wie folgt vor, um Net.Data für FastCGI zu konfigurieren:

1. Konfigurieren Sie die Webserver- und die FastCGI-Konfigurationsdatei für Ihr Betriebssystem:

   Für Apache Web Server:

   Aktualisieren Sie die Datei `httpd.conf`.

   - Deklarieren Sie die neue Anwendung:

     ```
     AppClass inst_dir
     -processes proc_num
     -initial-env LIBPATH=libpath
     -initial-env ORACLE_HOME=oracle_path
     -initial-env ORACLE_SID=oracle_instance
     ```

```
-initial-env DB2INSTANCE=db2_instance
-initial-env RXQUEUE_OWNER_PID=REXX_perf_var
-initial-env LANG=locale
```

- Deklarieren Sie das FastCGI-Modul:

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

Für IBM HTTP Web Server:

Aktivieren Sie das FastCGI-Modul und das Net.Data-FastCGI-Modul:

- Verwenden Sie in der Datei `httpd.conf` die folgenden Einträge:
  -
    ```
    #Load FCGI Module
    LoadModule fastcgi_module   libex/mod_fastcgi.so
    ```
  -
    ```
    # Add FCGI Module
    AddModule mod_fastgi.c
    ```
  -
    ```
    FastCgiServer  /usr/HTTPServer/db2www/fcgi-bin/fcgi-bin/db2www
    -appConnTimeout 0 -idle-timeout 30 -init-start-delay 1
    -listen-queue-depth 100 -processes 3
    -restart-delay 5 -port 7125

    <Location /fcgi-bin>
    SetHandler fastcgi-script

    <Location>
    ```

Parameter:

inst_dir

Der Pfad und Verzeichnisname für die ausführbaren Dateien von Net.Data.

Für Apache Web Server:

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

Für IBM HTTP Web Server:

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

proc_num

Die Anzahl Anforderungen, die gleichzeitig bearbeitet werden können. Der Standardwert ist 1, sollte jedoch gemäß Ihren Anwendungsanforderungen für eine Leistungsverbesserung erhöht werden. Informationen zur Optimierung der Leistung finden Sie in „Verwenden von FastCGI" auf Seite 200.

Für Apache Web Server:

```
-processes 7
```

IBM HTTP Web Server:

```
NumProcesses 7
```

MAXREQUEST

Gibt die Anzahl Anforderungen an, die der Net.Data FastCGI-Prozess bearbeitet, bevor der Webserver einen neuen Prozess startet.

Für Apache Web Server:

```
SetEnv MAXREQUEST 5000
```

Für IBM HTTP Web Server:

```
SetEnv MAXREQUEST 5000
```

2. **Für Apache:** Fügen Sie das Verzeichnis `fcgi-bin` als einen neuen Script-Aliasnamen in der Datei „srm.conf" hinzu: `ScriptAlias /fcgi-bin/ /u/mydir/apache/fcgi-bin`
3. Stellen Sie alle Hyperlinks in statisch oder dynamisch generierten Webseiten von CGI-BIN auf FCGI-BIN um. Beispiel:

   ```
   <a href="http://server/fcgi-bin/db2www/filename.ext/block
   [?name=val&...]">any text</a>
   ```

4. Ändern Sie die Endbenutzerdokumentation für Aufrufen von URL-Adressen durch Net.Data mit FCGI-BIN anstelle von CGI-BIN. Beispiel:

   ```
   http://server/fcgi-bin/db2www/filename.ext/block[?name=val&...]
   ```

Konfigurieren von Net.Data zur Verwendung mit Java-Servlets

Anweisungen zum Registrieren und Verwenden von Servlets finden Sie in der Dokumentation zu Ihrem Webserver. Die Net.Data-Servlets befinden sich in der Datei NetDataServlets.jar. Ihr Webserver erfordert, dass der Anweisung CLASSPATH *inst_dir*`/servlet-lib/NetDataServlets.jar` und *inst_dir*`/servlet-lib` hinzugefügt werden.

Anmerkung: Sie müssen die Datei NetDataServlets.jar dekomprimieren. Bei einigen Webservern ist es erforderlich, dass alle jar-Dateien dekomprimiert werden, bevor sie verwendet werden können.

Weitere Informationen zur Installation des Webservers und zu den Anweisungen in der Konfigurationsdatei des Webservers finden Sie in Ihrer Webserverdokumentation.

Konfigurieren von Net.Data zur Verwendung mit den Webserver-APIs

Die Verwendung einer Webserveranwendungsprogrammierschnittstelle (API) anstelle von CGI kann die Leistung von Net.Data wesentlich steigern. Net.Data unterstützt die folgenden Server-APIs:

- Apache API (APAPI)
- Microsoft Internet Server API (ISAPI)
- Netscape API (NSAPI)

Weitere Informationen zu den einzelnen APIs finden Sie in „Verwenden von FastCGI" auf Seite 200 und in der Informationsdatei (README) für Ihre Version von Net.Data.

Voraussetzung: Zur Ausführung von Net.Data im ISAPI- oder NSAPI-Modus müssen Sie Ihren Webserver rekonfigurieren, so dass er DLL-Dateien oder gemeinsam benutzte Bibliotheken von Net.Data als Serviceanweisungen verwendet. Nach dem Rekonfigurieren müssen Sie Ihren Webserver erneut starten, so dass die von Ihnen an der Net.Data-Initialisierungsdatei vorgenommenen Änderungen wirksam werden. Net.Data wird standardmäßig im CGI-Modus ausgeführt.

Konfigurieren Sie Net.Data so, dass nur jeweils eine Schnittstelle verwendet wird. Wenn Sie beispielsweise den Webserver so konfigurieren, dass Net.Data mit FastCGI ausgeführt wird, dürfen Sie den Webserver nicht zusätzlich so konfigurieren, dass Net.Data mit ISAPI oder einer anderen Schnittstelle ausgeführt wird. Wenn Sie Net.Data später mit einer anderen Schnittstelle ausführen wollen, z. B. NSAPI, müssen Sie den Webserver zur ausschließlichen Verwendung der neuen Schnittstelle rekonfigurieren.

In den folgenden Abschnitten wird beschrieben, wie Sie Net.Data und den Webserver zur Ausführung im API-Modus konfigurieren können. Die folgenden Schritte und Beispiele sind allgemein gehalten und weichen eventuell von Ihrem Betriebssystem ab. Spezifische Anweisungen finden Sie in der Net.Data-Informationsdatei (README) für Ihr Betriebssystem.

Gehen Sie wie folgt vor, um die Apache API zu konfigurieren:

1. Stoppen Sie den Webserver.
2. Stellen Sie sicher, dass `libmod_db2www.so` sich im Verzeichnis `/opt/netdata/lib` befindet.
3. Nur für Linux: Kopieren Sie `libmod_db2www.so` (Linux) oder mod_db2www.dll (WinNT) in das Verzeichnis `/apache/modules`.
4. Fügen Sie der Konfigurationsdatei des Webservers (`httpd.conf`) eine Serviceanweisung hinzu, um die API aufzurufen.

Beispiel:

```
LoadModule db2www_module /pot/netdata/lib/libmod_db2www.so AddHandler
     db2 www_handler .db2www
```

Spezifische Datei- und Verzeichnisnamen finden Sie in der Net.Data-Informationsdatei (README) für Ihr Betriebssystem.

5. Starten Sie den Webserver erneut.

APAPI wird nur unter Windows NT, Linux und Linux s/390 unterstützt.

Gehen Sie wie folgt vor, um ISAPI zu konfigurieren:

1. Stoppen Sie den Webserver.
2. Kopieren Sie die mit Net.Data gelieferte DLL-Datei für ISAPI in das Unterverzeichnis des Servers. Beispiel:

   `/inetsrv/scripts/dtwisapi.`*`filetype`*

   Dabei gilt Folgendes: *filetype* ist für Windows NT und OS/2 `.dll` und für UNIX `.o`.

   Spezifische Datei- und Verzeichnisnamen finden Sie in der Net.Data-Informationsdatei (README) für Ihr Betriebssystem.
3. Da ISAPI die CGI-Verarbeitung umgeht, können Sie den Teil `cgi-bin/db2www/` der URL-Adresse in Formularen und Verbindungen (Links) auslassen. Verwenden Sie stattdessen `dtwisapi.`*`filetype`*. Beispiel: Die folgende URL-Adresse ruft Net.Data als CGI-Programm auf:

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

   In diesem Fall müssen Sie Net.Data als ISAPI-Plug-In mit der folgenden URL-Adresse aufrufen:

   `http://server1.stl.ibm.com/scripts/dtwisapi.dll/test1.dtw/report`
4. Wenn Sie Ihr Makro `test1.dtw` im Unterverzeichnis `/order/` unter einem der in der Anweisung MACRO_PATH angegebenen Verzeichnisse oder im aktuellen Verzeichnis des Webservers gespeichert haben, rufen Sie Net.Data mit der folgenden URL-Adresse im CGI-Modus auf:

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

   Die entsprechende URL-Adresse zum Aufrufen von Net.Data im ISAPI-Modus lautet dann wie folgt:

   `http://server1.stl.ibm.com/scripts/dtwisapi.dll/orders/test1.dtw/report`
5. Starten Sie den Webserver erneut.

Gehen Sie wie folgt vor, um NSAPI zu konfigurieren:

1. Stoppen Sie den Webserver.
2. Kopieren Sie die mit Net.Data gelieferte DLL-Datei für NSAPI in das Serververzeichnis. Beispiel:

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

   Dabei gilt Folgendes: *filetype* ist für Windows NT und OS/2 `.dll` und für UNIX `.o`. Spezifische Datei- und Verzeichnisnamen finden Sie in der Net.Data-Informationsdatei (README) für Ihr Betriebssystem.
3. Ändern Sie Ihre Serverkonfigurationsdatei wie unten angegeben. Informationen zu den Unterschieden zwischen den Betriebssystemen finden Sie in der Net.Data-Informationsdatei (README) bzw. im Programmverzeichnis für Ihr Betriebssystem.

obj.conf	Fügen Sie am Anfang der Datei folgende Angaben hinzu: `Init fn="load-modules" shlib="<path>dtwnsapi.dll" funcs=dtw_nsapi`
obj.conf	Fügen Sie der Serviceanweisung folgende Angaben hinzu: `Service fn="dtw_nsapi" method=(GET|HEAD|POST) type="magnus-internal/d2w"`
mime.types	Fügen Sie diesen Typ hinzu, wobei *d2w* die Standarderweiterung des Makros ist. Sie können eine beliebige Kombination aus drei Zeichen angeben. `type=magnus-internal/d2w exts=d2w`

4. Versetzen Sie die Net.Data-Makros aus dem Verzeichnis `netdata/macro` in das Dokumentstammverzeichnis des Servers:

   ```
   /netscape/server/docs/
   ```
5. Fügen Sie der Anweisung MACRO_PATH in der Initialisierungsdatei das Dokumentstammverzeichnis des Servers hinzu. Durch diese Änderung wird Net.Data mitgeteilt, an welcher Position nach den Makros gesucht werden soll.
6. Da NSAPI die CGI-Verarbeitung umgeht, können Sie den Teil `cgi-bin/db2www/` der URL-Adresse in Formularen und Verbindungen (Links) auslassen. Der Server erkennt Dateien mit dem Dateityp `d2w` als Net.Data-Makros, weil Sie dies beim Ändern der Netscape-Konfigurationsdateien entsprechend definiert haben. Zum Beispiel ruft die folgende URL-Adresse Net.Data als CGI-Programm auf:

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

   Die folgende URL-Adresse hingegen ruft Net.Data als NSAPI-Plug-In auf:

   ```
   http://server1.stl.ibm.com/test1.dtw/report
   ```

7. Starten Sie den Webserver erneut.

Wenn Sie Ihre Net.Data-Makros in verschiedenen Verzeichnissen speichern, ändern sich die letzten drei Schritte:

1. Versetzen Sie die Verzeichnisse mit den darin enthaltenen Net.Data-Makros in das Dokumentstammverzeichnis des Servers.
2. Aktualisieren Sie die Variable MACRO_PATH in der Initialisierungsdatei so, dass sie alle Verzeichnisse und Unterverzeichnisse mit Makros enthält.
3. Ändern Sie die Verbindungen (Links) und Formulare, die auf diese Net.Data-Makros verweisen, und behalten Sie die jeweiligen Verzeichnisnamen bei. Beispielsweise ruft die folgende URL-Adresse bei Ausführung in CGI ein Net.Data-Makro auf, das im Verzeichnis `/orders/` gespeichert ist:

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

   Die aktualisierte URL-Adresse zum Aufrufen von Net.Data im NSAPI-Modus ist kürzer, behält jedoch den Verzeichnisnamen bei:

   ```
   http://server1.stl.ibm.com/orders/test1.dtw/report
   ```

Konfigurieren von Net.Data mit Net.Data Administration-Tool

Net.Data Administration-Tool hilft Ihnen beim Konfigurieren und Verwalten der Net.Data-Initialisierungsdatei (DB2WWW.INI) und der Konfigurationsdatei für Direktverbindungen (dtwcm.cnf) auf den Betriebssystemen Windows NT, AIX und OS/2. Mit diesem Tool können Sie die folgenden Funktionen ausführen:

- „Starten von Administration-Tool" auf Seite 51
- „Konfigurieren von Pfadanweisungen" auf Seite 52
- „Konfigurieren von Anschlüssen" auf Seite 54
- „Konfigurieren von Cliettes" auf Seite 55
- „Konfigurieren von Sprachumgebungen" auf Seite 60
- „Definieren von Konfigurationsvariablen" auf Seite 64

Informationen zum Einrichten von Administration-Tool und zum Sicherstellen, dass die Softwarevoraussetzungen erfüllt werden, finden Sie in „Vorbereitung" auf Seite 51.

Vorbereitung

1. Planen Sie die Konfiguration der Sprachumgebungen (Language Environments), Datenbanken, Cliettes, Anschlüsse und Konfigurationsvariablen von Net.Data.
2. Installieren Sie Net.Data von der CD-ROM.
3. Installieren Sie die Java-Laufzeitbibliotheken (JDK 1.1 und nachfolgende Versionen für jedes Betriebssystem). Weitere Informationen finden Sie in der Net.Data-Informationsdatei (README) für Ihr Betriebssystem.

   Stellen Sie sicher, dass `classes.zip` nach der Installation von JDK in Ihrer Anweisung CLASSPATH angegeben ist.
4. Wenn Sie den IBM JDBC-Treiber installiert haben, der zum Lieferumfang von DB2 Universal Database gehört, fügen Sie das Treiberverzeichnis Ihrer Java-Anweisung CLASSPATH hinzu, um die DB2-Testanmeldung zu aktivieren.
5. Wechseln Sie in das Verzeichnis, in dem Net.Data Administration-Tool gespeichert ist:

   Für OS/2 und Windows NT:

   inst_dir`\connect\`*admin_directory*, wobei *inst_dir* das für Net.Data während der Installation angegebene Verzeichnis und *admin_directory* das Verzeichnis ist, in dem sich die Dateien von Administration-Tool befinden.

   Für AIX:

   `/usr/lpp/internet/db2www/db2.v2/`*admin_directory*, wobei *admin_directory* das Verzeichnis ist, in dem sich die Dateien von Administration-Tool befinden.

Starten von Administration-Tool

Das von Ihnen verwendete Betriebssystem legt fest, wie Sie Administration-Tool starten.

Für OS/2 und Windows NT:

Wählen Sie im Ordner für IBM Net.Data das Symbol **Net.Data Administration** aus.

Für AIX:

Wechseln Sie in das Net.Data-Installationsverzeichnis (inst_dir). Geben Sie in der Befehlszeile `ndadmin` ein, um das Tool zu starten.

Administration-Tool wird gestartet, und das Notizbuch **Net.Data Administration** wird angezeigt.

Konfigurieren von Pfadanweisungen

Auf der Seite **Pfad** können Sie die Pfadanweisungen zum Lokalisieren der Dateien, die Net.Data zum Verarbeiten von Net.Data-Makros benötigt, hinzufügen, ändern oder löschen. Diese Anweisungen werden in „Pfadkonfigurationsanweisungen" auf Seite 24 beschrieben.
Abb. 7 zeigt die Seite **Pfad**.

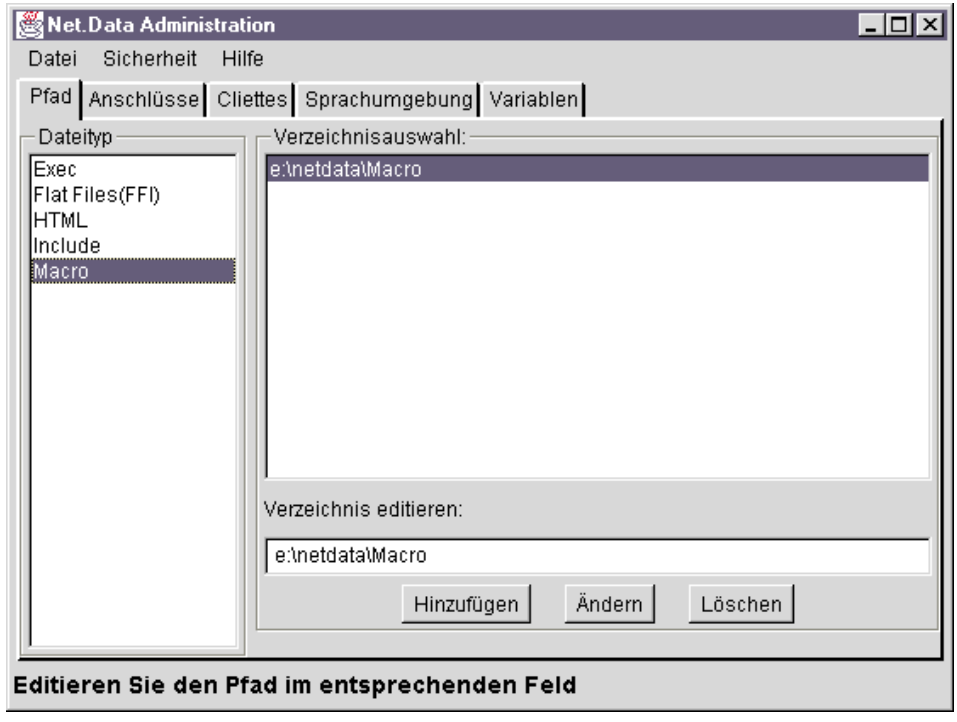


Abbildung 7. Die Seite "Pfad" des Net.Data Administration-Tools. Auf dieser Seite können Sie Pfadanweisungen hinzufügen, ändern oder löschen.

Konfigurationshinweis: Der Dateityp HTML kann nur einen einzigen Pfad aufweisen.

Gehen Sie wie folgt vor, um eine Pfadanweisung hinzuzufügen:

1. Starten Sie Administration-Tool.
2. Wählen Sie auf der Seite **Pfad** in der Liste **Dateityp** einen Dateityp aus, zum Beispiel `Exec`.
3. Geben Sie im Feld **Verzeichnis editieren** den neuen Pfad ein, und klicken Sie den Knopf **Hinzufügen** an.

   Wenn der angegebene Pfad nicht vorhanden ist, wird ein Fenster mit einer Warnung geöffnet. Wenn kein Verzeichnis ausgewählt ist, wird das neue Verzeichnis als letzter Eintrag in der Liste hinzugefügt.
4. Schließen Sie Administration-Tool, oder klicken Sie eine andere Indexzunge an, um zusätzliche Konfigurationsaufgaben auszuführen.

Gehen Sie wie folgt vor, um eine Pfadanweisung zu ändern:

1. Starten Sie Administration-Tool.
2. Wählen Sie auf der Seite **Pfad** in der Liste **Dateityp** den zu ändernden Dateityp aus.
3. Wählen Sie den zu ändernden Pfad in der Liste **Verzeichnisauswahl** aus. Der ausgewählte Pfad wird im Feld **Verzeichnis editieren** angezeigt.
4. Geben Sie im Feld **Verzeichnis editieren** einen anderen Pfad an, und klicken Sie den Knopf **Ändern** an. Wenn der eingegebene Pfad nicht vorhanden ist, wird ein Fenster mit einer Warnung geöffnet.
5. Schließen Sie Administration-Tool, oder klicken Sie eine andere Indexzunge an, um zusätzliche Konfigurationsaufgaben auszuführen.

Gehen Sie wie folgt vor, um eine Pfadanweisung zu löschen:

1. Starten Sie Administration-Tool.
2. Wählen Sie auf der Seite **Pfad** in der Liste **Dateityp** den zu löschenden Dateityp aus.
3. Wählen Sie den zu löschenden Pfad im Feld **Verzeichnisauswahl** aus. Der ausgewählte Pfad wird im Feld **Verzeichnis editieren** angezeigt.
4. Klicken Sie den Knopf **Löschen** an.
5. Schließen Sie Administration-Tool, oder klicken Sie eine andere Indexzunge an, um zusätzliche Konfigurationsaufgaben auszuführen.

Konfigurieren von Anschlüssen

Auf der Seite **Anschlüsse** können Sie die von Net.Data verwendeten TCP/IP-Anschlussnummern angeben. Abb. 8 zeigt die Seite **Anschlüsse**.

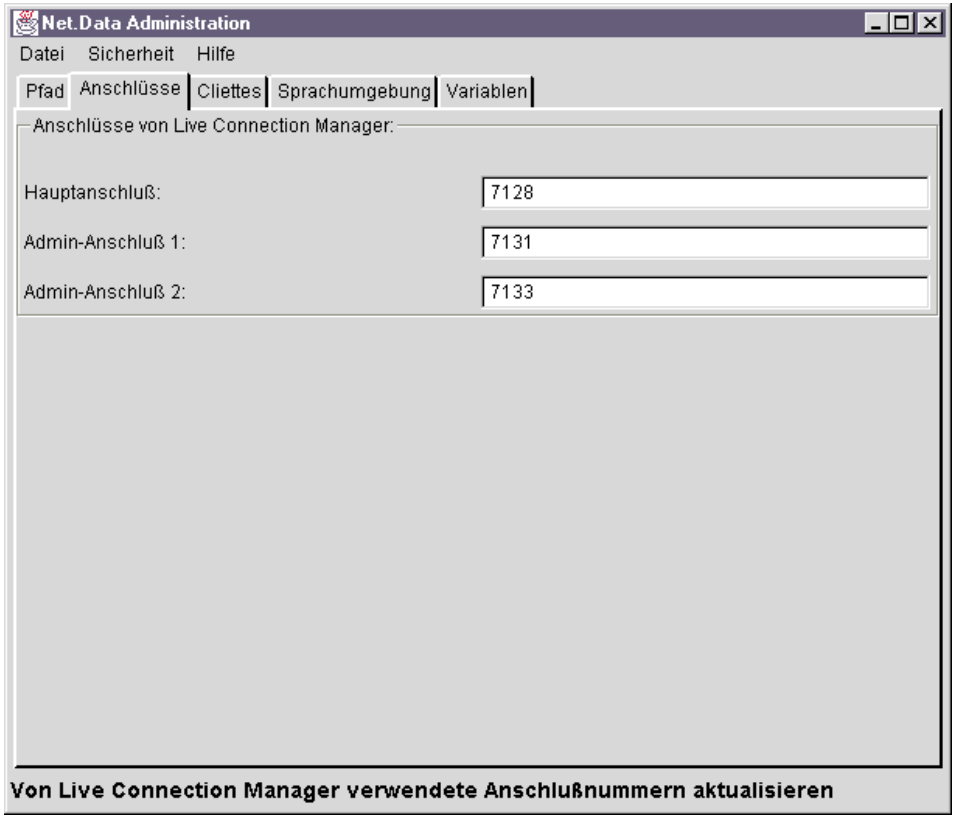


Abbildung 8. Die Seite "Anschlüsse" des Net.Data Administration-Tools. Auf dieser Seite können Sie Anschlüsse angeben.

Gehen Sie wie folgt vor, um TCP/IP-Anschlussnummern anzugeben:

1. Starten Sie Administration-Tool.
2. Geben Sie auf der Seite **Anschlüsse** in den einzelnen Anschlussfeldern jeweils eine eindeutige Anschlussnummer ein. Administration-Tool prüft die in den einzelnen Feldern eingegebenen Anschlussnummern jeweils beim Drücken der Tabulatortaste, durch die das nächste Feld angesteuert wird.
3. Schließen Sie Administration-Tool, oder klicken Sie eine andere Indexzunge an, um zusätzliche Konfigurationsaufgaben auszuführen.

Konfigurieren von Cliettes

Auf der Seite **Cliettes** können Sie Datenbank-Cliettes für Direktverbindungen hinzufügen, ändern oder löschen, und Sie können ferner Benutzer-IDs und Kennwörter von Datenbanken und Administratoren für Datenbank-Cliettes verwalten. Weitere Informationen zu Cliettes finden Sie in „Verwalten von Verbindungen" auf Seite 200. Abb. 9 zeigt die Seite **Cliettes**.
Gehen Sie wie folgt vor, um eine Cliette hinzuzufügen:

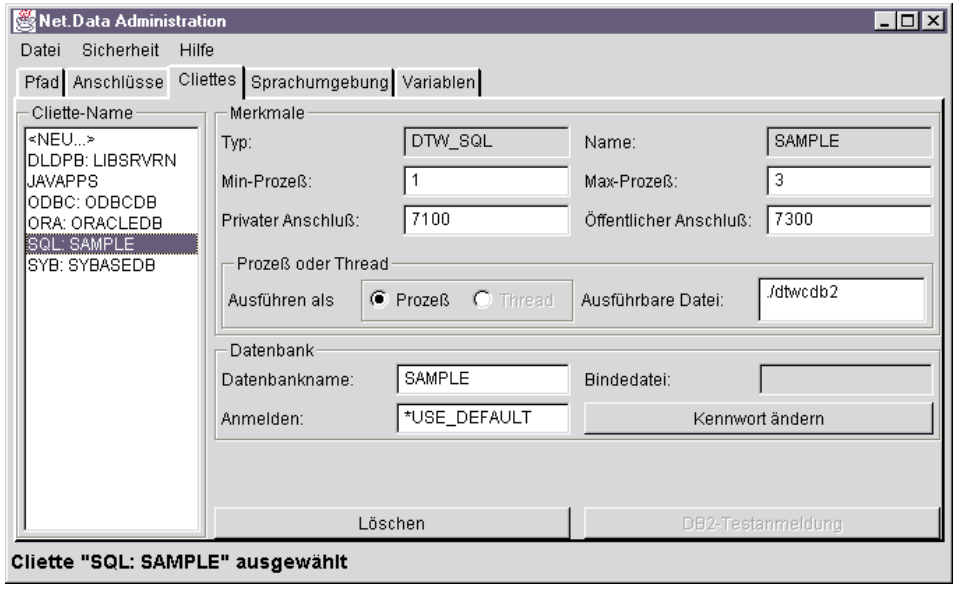


Abbildung 9. Die Seite "Cliettes" des Net.Data Administration-Tools. Auf dieser Seite können Sie Cliettes hinzufügen, ändern und löschen.

1. Starten Sie Administration-Tool.
2. Wählen Sie auf der Seite **Cliettes** in der Liste **Cliette-Name** die Option **<neu...>** aus. Das Fenster **Cliette hinzufügen** wird geöffnet.

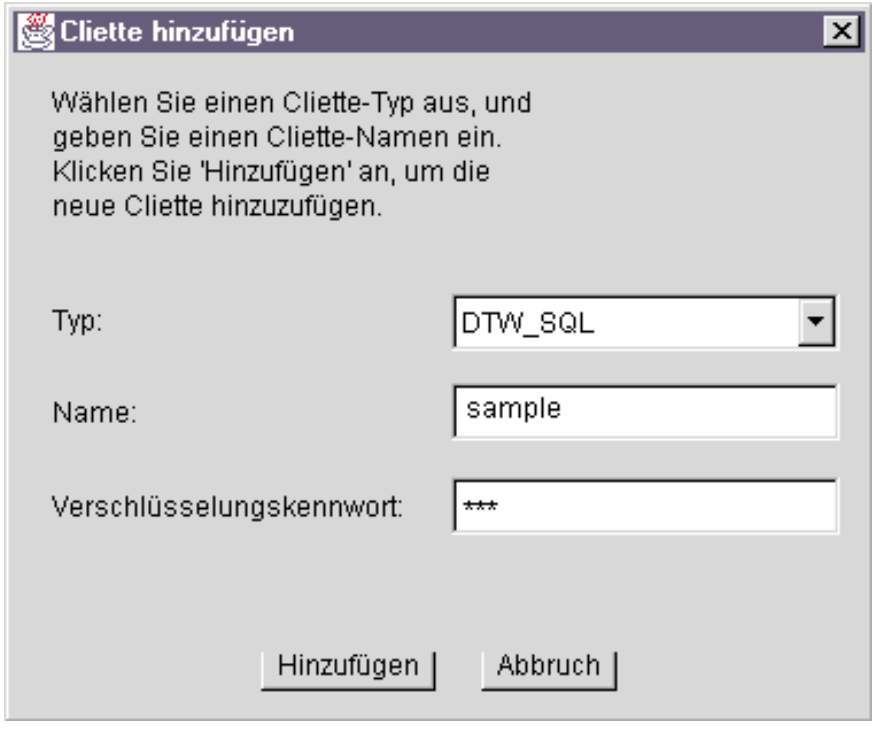


Abbildung 10. Das Fenster "Cliette hinzufügen" des Net.Data Administration-Tools. Auf dieser Seite können Sie Cliettes hinzufügen.

Wenn Sie die Verschlüsselung aktiviert haben, werden Sie beim erstmaligen Erstellen oder Ändern einer Cliette zur Eingabe des Verschlüsselungskennworts aufgefordert. Dieses Kennwort wird gesichert, und Sie brauchen es nie mehr einzugeben.

3. Wählen Sie in der Liste **Typ** einen Cliette-Typ aus.
4. Geben Sie im Feld **Name** einen Namen für die neue Cliette ein. Der Name kann der Name der Datenbank oder ein anderer eindeutiger Cliette-Name sein, zum Beispiel: MYCLIETTE.
5. Geben Sie das Verschlüsselungskennwort ein, wenn das Feld **Verschlüsselungskennwort** aktiviert ist. Sie brauchen das Kennwort nicht erneut einzugeben, weil Administration-Tool das Kennwort für Sie sichert.
6. Klicken Sie den Knopf **Hinzufügen** an. Die neue Cliette wird erstellt und an das Ende der Cliette-Liste hinzugefügt. Außerdem wird der neue Name hervorgehoben, und die Standardmerkmale für die Cliette werden in der Auswahlgruppe **Merkmale** angezeigt. Sie können diese Werte Ihrer Konfiguration anpassen.
7. Schließen Sie Administration-Tool, oder klicken Sie eine andere Indexzunge an, um zusätzliche Konfigurationsaufgaben auszuführen.

Gehen Sie wie folgt vor, um eine Cliette zu ändern:

1. Starten Sie Administration-Tool.
2. Wählen Sie auf der Seite **Cliettes** in der Liste **Cliette-Name** den Namen der zu ändernden Cliette aus. Die Merkmale der Cliette werden in der Auswahlgruppe **Merkmale** angezeigt.
3. Ändern Sie die Merkmale wie erforderlich in der Auswahlgruppe **Merkmale**.
   a. Im Feld **Typ** wird der Cliette-Typ angezeigt, der definiert wird und der dem Namen einer Sprachumgebungsart entspricht. Net.Data gibt die erforderlichen Angaben in dieses Feld ein, wenn Sie eine neue Cliette hinzufügen. Die Auswahlmöglichkeiten werden in der Liste **Typ** im Fenster **Cliette hinzufügen** definiert.
   b. Das Feld **Name** zeigt den Namen der Cliette an, der in der Regel der Name der Datenbank ist. Net.Data gibt die erforderlichen Angaben in dieses Feld ein, wenn Sie eine neue Cliette hinzufügen.
   c. Geben Sie im Feld **Min-Prozess** die Anzahl von Cliette-Prozessen ein, die beim Start von Connection Manager gestartet werden können. Für jeden Prozess ist eine eindeutige Anschlussadresse erforderlich. Weitere Informationen zu den Werten für **Min-Prozess** finden Sie in „Konfigurieren der Direktverbindung" auf Seite 36.
   d. Geben Sie im Feld **Max-Prozess** die Anzahl von Cliette-Prozessen ein, die zusätzlich zu den beim Start von Connection Manager gestarteten Prozessen gleichzeitig ausgeführt werden können. Für jeden Prozess ist eine eindeutige Anschlussadresse erforderlich. Weitere Informationen zu den Werten für **Max-Prozess** finden Sie in „Konfigurieren der Direktverbindung" auf Seite 36.
   e. Geben Sie im Feld **Privater Anschluss** eine eindeutige Anschlussnummer ein, um die Anfangsanschlussnummer für die Cliette-Prozesse anzugeben, die mit Connection Manager gestartet werden. Für jeden durch den Wert von **Min-Prozess** angegebenen Prozess wird eine zusätzliche Anschlussnummer verwendet. Wenn Sie zum Beispiel die Anschlussnummer 7012 für **Privater Anschluss** und den Wert 5 für **Min-Prozess** angeben, werden die Anschlussnummern 7012 bis 7016 verwendet, die nicht mit anderen Anschlusszuordnungen im System kollidieren dürfen.
   f. Geben Sie im Feld **Öffentlicher Anschluss** eine eindeutige Anschlussnummer ein, um die Anfangsanschlussnummer für die Cliette-Prozesse anzugeben, die beim Start zusätzlicher Prozesse gestartet werden. Die Endanschlussnummer wird durch die im Feld **Max-Prozess** angegebene Zahl definiert. Für jeden der Prozesse wird eine zusätzliche Anschlussnummer verwendet. Wenn Sie zum Beispiel die Anschlussnummer 7020 für **Öffentlicher Anschluss** und den Wert 5 für **Max-Prozess** angeben, werden die Anschlussnummern 7020 bis 7024 verwendet, die nicht mit anderen Anschlusszuordnungen im System kollidieren dürfen.

g. Im Feld **Ausführbare Datei** wird der Name der ausführbaren Cliette-Datei angezeigt.

4. Wenn die Cliette mit einer Datenbank verwendet wird, ändern Sie die Werte für die Auswahlgruppe **Datenbank** wie erforderlich:
   a. Geben Sie im Feld **Datenbankname** den Namen der Datenbank an, der die Cliette zugeordnet ist, zum Beispiel MYDBASE.
   b. Das Feld **Bindedatei** enthält den Namen und Pfad der Bindedatei für den verwendeten Cliette-Typ.
   c. Das Feld **Anmelden** gibt die Anmeldebenutzer-ID an, mit der die Verbindung zur Datenbank hergestellt wird.
   d. Durch den Druckknopf **Kennwort ändern** wird das Fenster **Datenbankkennwort ändern** geöffnet. Geben Sie das Verschlüsselungskennwort und das neue Kennwort zweimal ein. Sie können das Datenbankkennwort mit Hilfe der im Menü **Sicherheit** angegebenen Verschlüsselungsfunktionen verschlüsseln.
5. Wählen Sie **Datei** und anschließend **Sichern** aus, um Ihre Änderungen zu sichern.
6. Schließen Sie Administration-Tool, oder klicken Sie eine andere Indexzunge an, um zusätzliche Konfigurationsaufgaben auszuführen.

Gehen Sie wie folgt vor, um die DB2-Datenbankanmeldung und -verbindung zu testen:

1. Klicken Sie auf der Seite **Cliettes** von Administration-Tool den Druckknopf **DB2-Testanmeldung** an. Wenn der Test abgeschlossen ist, wird ein Bestätigungsfenster geöffnet, in dem der Status des Verbindungstests angezeigt wird.
2. Schließen Sie das Fenster, um den Konfigurationsvorgang fortzusetzen, oder schließen Sie Administration-Tool.

Gehen Sie wie folgt vor, um eine Cliette zu löschen:

1. Starten Sie Administration-Tool.
2. Wählen Sie auf der Seite **Cliettes** in der Liste **Cliette-Name** den Namen der zu löschenden Cliette aus.
3. Klicken Sie den Knopf **Löschen** an.
4. Schließen Sie Administration-Tool, oder klicken Sie eine andere Indexzunge an, um zusätzliche Konfigurationsaufgaben auszuführen.

Gehen Sie wie folgt vor, um die Verschlüsselung von Benutzer-IDs und Kennwörtern für Cliettes zu aktivieren:

Die Verschlüsselung bietet Sicherheit für Datenbankverbindungen mit Cliettes. Wenn die Verschlüsselung aktiviert ist, werden alle Datenbankkennwörter in der Konfigurationsdatei für Direktverbindungen verschlüsselt und erfordern ein Verschlüsselungskennwort für Zugriff und Entschlüsselung.

Voraussetzung: Sie müssen mit einer Konfigurationsdatei für Direktverbindungen von Net.Data Version 2 arbeiten, um die Verschlüsselung verwenden zu können.

1. **Wichtig:** Erstellen Sie eine Sicherungskopie Ihrer Konfigurationsdatei für Direktverbindungen namens `<path>dtwcm.cnf`. Diese Datei ist erforderlich, falls Sie das Verschlüsselungskennwort verlieren oder Datenbankkennwörter entschlüsseln wollen und die Kennwörter wiederherstellen müssen.
2. Wählen Sie auf der Seite **Cliettes** von Administration-Tool die Menüoption **Sicherheit -> Verschlüsselung ein** aus. Das Fenster **Bestätigung für das Aktivieren der Verschlüsselung** wird geöffnet.
3. Klicken Sie **Ja** an, um den Vorgang fortzusetzen. Das Fenster **Verschlüsselungskennwort** wird geöffnet.
4. Geben Sie das Kennwort für die Berechtigung zum Arbeiten mit Cliettes, für die es verschlüsselte Kennwörter gibt, zweimal ein.
5. Klicken Sie **OK** an, um das neue Kennwort zu definieren und alle Datenbankkennwörter für Ihre Cliettes zu verschlüsseln.

Gehen Sie wie folgt vor, um die Verschlüsselung von Benutzer-IDs und Kennwörtern für Cliettes zu inaktivieren:

1. Wählen Sie auf der Seite **Cliettes** von Administration-Tool die Menüoption **Sicherheit -> Verschlüsselung aus** aus. Das Fenster **Bestätigung für das Inaktivieren der Verschlüsselung** wird geöffnet.
2. Klicken Sie **Ja** an, um den Vorgang fortzusetzen. Alle Kennwörter werden aus Sicherheitsgründen auf *USE_DEFAULT gesetzt. Sie können Ihre Kennwörter von der Sicherungskopie der Direktverbindungsdatei `<path>dtwcm.cnf` wiederherstellen.

Gehen Sie wie folgt vor, um das Verschlüsselungskennwort zu ändern:

1. Wählen Sie auf der Seite **Cliettes** von Administration-Tool die Menüoption **Sicherheit -> Verschlüsselungskennwort ändern** aus. Das Fenster **Änderung des Verschlüsselungskennworts bestätigen** wird geöffnet.
2. Klicken Sie **Ja** an, um den Vorgang fortzusetzen. Das Fenster **Verschlüsselungskennwort ändern** wird geöffnet.

3. Geben Sie das alte Verschlüsselungskennwort einmal und das neue Kennwort zweimal ein.
4. Klicken Sie **OK** an, um das Verschlüsselungskennwort zu ändern.

Gehen Sie wie folgt, um das Datenbankkennwort zu ändern:

1. Klicken Sie auf der Seite **Cliettes** von Administration-Tool den Druckknopf **Kennwort ändern** an. Das Fenster **Datenbankkennwort ändern** wird geöffnet.
2. Geben Sie das Verschlüsselungskennwort einmal und das neue Datenbankkennwort zweimal ein.
3. Klicken Sie **OK** an, um das Kennwort zu ändern und das Fenster zu schließen. Das geänderte Datenbankkennwort wird verschlüsselt, wenn Sie Verschlüsselung aktiviert haben.

Konfigurieren von Sprachumgebungen

Auf der Seite **Sprachumgebung** können Sie Net.Data-Sprachumgebungen hinzufügen, ändern oder löschen. Sprachumgebungen werden in „Umgebungskonfigurationsanweisungen" auf Seite 29 ausführlich behandelt. Abb. 11 zeigt die Seite **Sprachumgebung**.

Gehen Sie wie folgt vor, um eine Sprachumgebung hinzuzufügen:

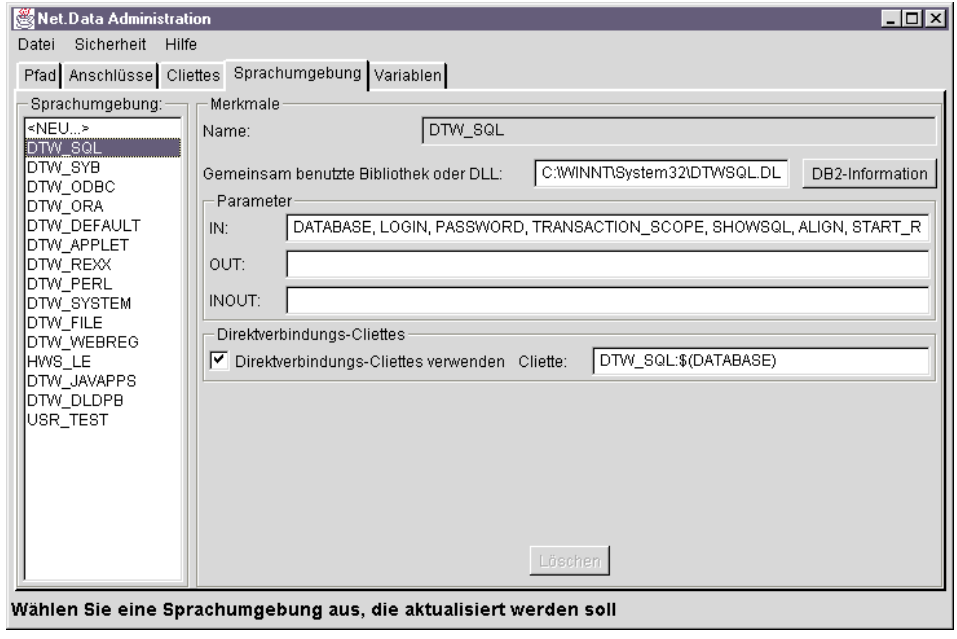


Abbildung 11. Die Seite "Sprachumgebung" des Net.Data Administration-Tools. Auf dieser Seite können Sie Sprachumgebungen angeben.

1. Starten Sie Administration-Tool.
2. Wählen Sie auf der Seite **Sprachumgebung** in der Liste **Sprachumgebung** die Option **<neu...>** aus. Das Fenster **Sprachumgebung hinzufügen** wird geöffnet.
3. Geben Sie den Namen der Sprachumgebung in das Feld ein, und klicken Sie den Knopf **Hinzufügen** an. Das Fenster **Sprachumgebung hinzufügen** wird geschlossen.

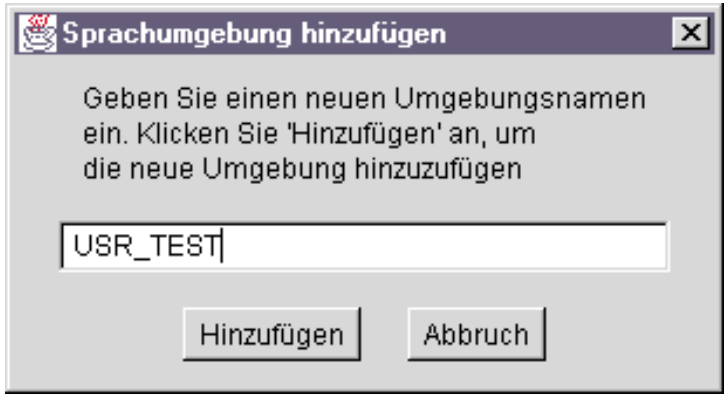


Abbildung 12. Das Fenster "Sprachumgebung hinzufügen" des Net.Data Administration-Tools. Auf dieser Seite können Sie eine neue Sprachumgebung angeben.

   Die neue Sprachumgebung wird erstellt, und ihr Name wird am Ende der Sprachumgebungsliste hinzugefügt. Außerdem wird der neue Name hervorgehoben, und die Standardmerkmale für die Sprachumgebung werden in der Auswahlgruppe **Merkmale** angezeigt. Sie können diese Werte Ihrer Konfiguration anpassen.
4. Schließen Sie Administration-Tool, oder klicken Sie eine andere Indexzunge an, um zusätzliche Konfigurationsaufgaben auszuführen.

Gehen Sie wie folgt vor, um eine Sprachumgebung zu ändern:

1. Starten Sie Administration-Tool.
2. Wählen Sie auf der Seite **Sprachumgebung** in der Liste **Sprachumgebung** den Namen der zu ändernden Sprachumgebung aus. Die Merkmale der Cliette werden in der Auswahlgruppe **Merkmale** angezeigt.
3. Ändern Sie die Merkmale in der Auswahlgruppe **Merkmale** wie in Abb. 12 gezeigt wie erforderlich:
   a. Geben Sie im Feld **Name** den Namen der Sprachumgebung an. Dieser Name entspricht der zum Definieren einer Cliette verwendeten Sprachumgebungsart. Sie können diesen Wert ändern, indem Sie in der Liste **Sprachumgebung** einen anderen Namen doppelt anklicken. Weitere

Informationen zu Sprachumgebungsarten finden Sie in „Umgebungskonfigurationsanweisungen” auf Seite 29.

b. Geben Sie im Feld **Gemeinsam benutzte Bibliothek oder DLL** die gemeinsam benutzte Bibliothek oder den DLL-Programmnamen und -Pfad für die Sprachumgebung an.

c. Wählen Sie den Druckknopf **DB2-Information** aus, um das Fenster **DB2-Information** aufzurufen, wie in Abb. 13 gezeigt.

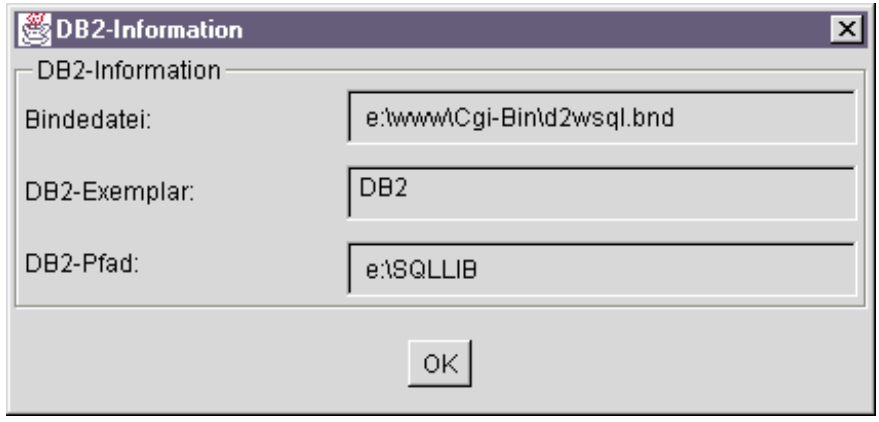


Abbildung 13. Das Fenster "DB2-Information" des Net.Data Administration-Tools. Auf dieser Seite können Sie spezifische DB2-Datenbankinformationen angeben.

Geben Sie die Werte für die DB2-Umgebungsvariablen an:

1) Geben Sie in das Feld **Bindedatei** den Pfad und Dateinamen der Bindedatei ein.
2) Geben Sie im Feld **DB2-Exemplar** den Wert von DB2INSTANCE für die zugehörige Datenbank an, wenn Sie die SQL-Sprachumgebung verwenden.
3) Geben Sie im Feld **DB2-Pfad** den Pfadverzeichnisnamen für die ausführbaren DB2-Produktdateien an (in der Regel \SQLLIB).
4) Klicken Sie **OK** an, um Ihre Änderungen zu sichern und das Fenster zu schließen.

d. Geben Sie in der Auswahlgruppe **Parameter** die Eingabe- und Ausgabeparameter an, die bei jedem Aufruf einer Sprachumgebung an die bzw. von der Sprachumgebung übergeben werden.

Hinweis: Aktualisieren Sie diese Felder nur, wenn Sie Ihre eigene Sprachumgebung definieren.

e. Geben Sie in der Auswahlgruppe **Direktverbindungs-Cliettes** an, ob Cliettes verwendet werden sollen und welche Cliette der Sprachumgebung zugeordnet werden soll.

1) Geben Sie an, ob die Cliette für die Sprachumgebung aktiv ist, indem Sie das Markierungsfeld **Direktverbindungs-Cliettes verwenden** aktivieren. Wählen Sie dieses Markierungsfeld aus, wenn Sie die im Feld **Cliette** angegebene Cliette beim Aufruf der Sprachumgebung verwenden wollen.
2) Geben Sie im Feld **Cliette** den Namen der Cliette an, die mit der soeben definierten Sprachumgebung ausgeführt werden soll. Die Syntax des Namens hängt davon ab, ob Sie die Sprachumgebung für eine Datenbank oder für die Java-Anwendung konfigurieren. Der Standardwert ist `DTW_SQL:$(DATABASE)`.

   Syntax für Datenbanken:

   type`:`*name*

   Dabei gilt Folgendes:

   type Die Sprachumgebungsart für die Cliette. Dies kann einer der folgenden Werte sein:

   Für Windows NT:
   DTW_ODBC, DTW_ORA, DTW_SQL, DTW_JAVAPPS

   Für OS/2:
   DTW_SQL, DTW_JAVAPPS

   Für AIX:
   DTW_ODBC, DTW_ORA, DTW_SQL, DTW_JAVAPPS

   name Der Name der auf der Seite **Cliette** definierten Cliette. Der Standardwert ist $(DATABASE).

   Syntax für Java-Anwendungen:

   `DTW_JAVAPPS`

4. Wählen Sie **Datei** und anschließend **Sichern** aus, um Ihre Änderungen zu sichern.
5. Schließen Sie Administration-Tool, oder klicken Sie eine andere Indexzunge an, um zusätzliche Konfigurationsaufgaben auszuführen.

Gehen Sie wie folgt vor, um eine Sprachumgebung zu löschen:

Einschränkung: Sie können nur die benutzerdefinierten Sprachumgebungen löschen, jedoch nicht die mit Net.Data gelieferten Sprachumgebungen.

1. Starten Sie Administration-Tool.

2. Wählen Sie auf der Seite **Sprachumgebung** in der Liste **Sprachumgebung** den Namen der zu löschenden Sprachumgebung aus.
3. Klicken Sie den Knopf **Löschen** an.
4. Schließen Sie Administration-Tool, oder klicken Sie eine andere Indexzunge an, um zusätzliche Konfigurationsaufgaben auszuführen.

Definieren von Konfigurationsvariablen

Auf der Seite **Variablen** können Sie das Benutzerverzeichnis für Net.Data angeben und die Protokollstufe für Fehlernachrichten auswählen. Abb. 14 zeigt die Seite **Variablen**.

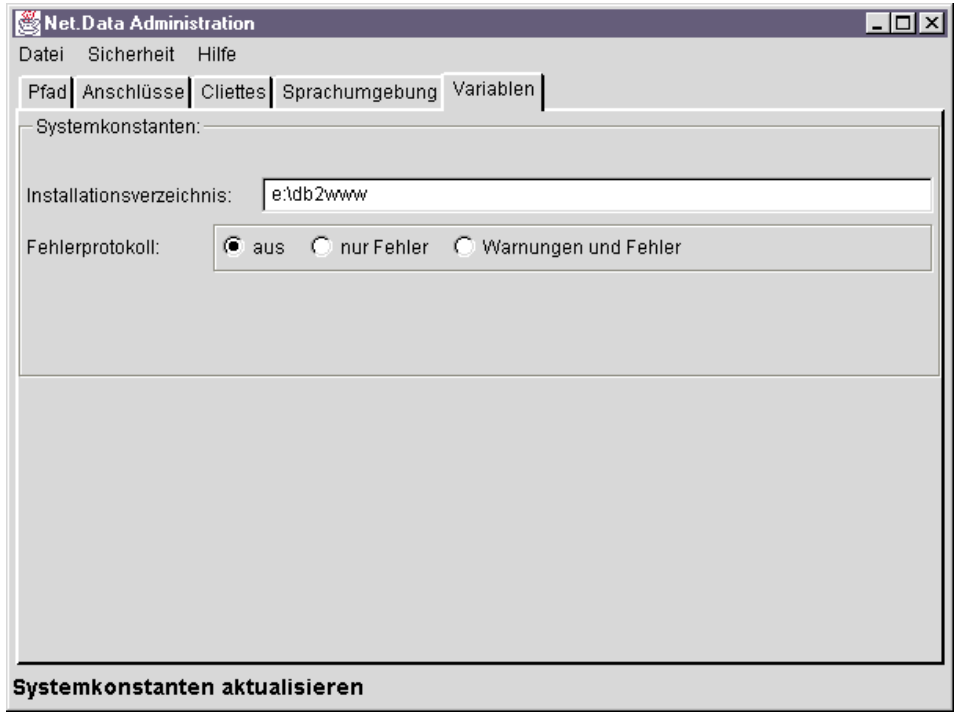


Abbildung 14. Die Seite "Variablen" des Net.Data Administration-Tools. Auf dieser Seite können Sie Initialisierungsvariablen angeben.

Gehen Sie wie folgt vor, um das Benutzerverzeichnis für Net.Data anzugeben:

Diese Variable ist auch als die Installationsverzeichnisvariable bekannt.

1. Starten Sie Administration-Tool.
2. Geben Sie auf der Seite **Variablen** im Feld **Installationsverzeichnis** den Pfad für das Verzeichnis ein, in dem die Protokolldatei gespeichert werden soll. Der Standardwert ist `\inst_dir\logs\` , zum Beispiel: `e:\db2www`.

3. Schließen Sie Administration-Tool, oder klicken Sie eine andere Indexzunge an, um zusätzliche Konfigurationsaufgaben auszuführen.

Gehen Sie wie folgt vor, um die Protokollstufe für Fehlernachrichten für Net.Data anzugeben:

1. Starten Sie Administration-Tool.
2. Wählen Sie auf der Seite **Variablen** in der Auswahlgruppe **Fehlerprotokoll** eine Stufe der Fehlerprotokollierung aus:
   - **aus**
   - **nur Fehler**
   - **Warnungen und Fehler**
3. Schließen Sie Administration-Tool, oder klicken Sie eine andere Indexzunge an, um zusätzliche Konfigurationsaufgaben auszuführen.

Erteilen von Zugriffsrechten für Dateien, auf die Net.Data zugreift

Vor der Verwendung von Net.Data müssen Sie sicherstellen, dass die Benutzer-IDs, unter denen Net.Data ausgeführt wird, über die notwendigen Zugriffsrechte für die Dateien verfügen, auf die in einem Net.Data-Makro verwiesen wird. Diese Rechte werden auch für das Makro benötigt, auf das in einer URL-Adresse verwiesen wird. Dies bedeutet, dass sich diese Dateien in Verzeichnissen bzw. Bibliotheken befinden müssen, zu denen der Webserver eine Verbindung herstellen kann oder für die diese Benutzer-IDs explizite Zugriffsrechte haben.

Stellen Sie vor allem sicher, dass die Benutzer-IDs, unter denen Net.Data ausgeführt wird, über die folgenden Berechtigungen verfügen:

- Lesen der Net.Data-Initialisierungsdatei `db2www.ini`
- Ausführen der ausführbaren Net.Data-Dateien und DLL-Dateien und Durchsuchen der Verzeichnisse in den Pfaden zu den ausführbaren Dateien und DLL-Dateien
- Lesen der entsprechenden Net.Data-Makros und Durchsuchen der entsprechenden Verzeichnisse, die durch die Pfadkonfigurationsanweisung MACRO_PATH angegeben werden
- Ausführen der entsprechenden Dateien und Durchsuchen der entsprechenden Verzeichnisse, die durch die Pfadkonfigurationsanweisung EXEC_PATH angegeben werden

- Lesen der entsprechenden Dateien und Durchsuchen der entsprechenden Verzeichnisse, die durch die Pfadkonfigurationsanweisung INCLUDE_PATH angegeben werden
- Lesen und Schreiben der entsprechenden Dateien und Durchsuchen der entsprechenden Verzeichnisse, die durch die Pfadkonfigurationsanweisung FFI-_PATH angegeben werden
- Lesen der Konfigurationsdatei für Direktverbindungen namens dtwcm.cnf
- Lesen der Konfigurationsdatei für den Cache-Manager namens CACHEM-GR.CNF
- Lesen der externen ausführbaren Perl- und REXX-Dateien, auf die durch die Sprachumgebungen verwiesen wird

Die Methoden zum Erteilen des Zugriffs auf diese Dateien hängen von dem Betriebssystem ab, unter dem Net.Data ausgeführt wird.

Kapitel 3. Sichern der Datenbestände

Internet-Sicherheit wird durch eine Kombination aus Firewall-Technologie, Betriebssystemfunktionen, Webserver-Funktionen, Net.Data-Mechanismen und Zugriffssteuerungsmechanismen, die Teil Ihrer Datenquellen sind, zur Verfügung gestellt.

Sie müssen entscheiden, welche Sicherheitsstufe für Ihre Datenbestände angebracht ist. In diesem Kapitel werden Methoden zur Sicherung Ihrer Datenbestände beschrieben und Verweise auf zusätzliche Quellen gegeben, mit denen Sie die Sicherheit Ihrer Website planen können.

In den folgenden Abschnitten werden Richtlinien für den Schutz Ihrer Datenbestände erläutert. Folgende Sicherheitsmechanismen werden beschrieben:

- „Verwenden von Firewalls"
- „Verschlüsseln Ihrer Daten im Netzwerk" auf Seite 70
- „Verwenden der Authentifizierung" auf Seite 70
- „Verwenden der Berechtigung" auf Seite 71
- „Verwenden von Net.Data-Mechanismen" auf Seite 71

Verwenden von Firewalls

Firewalls sind Gruppen von Hardware, Software und Maßnahmen, mit denen der Zugriff auf Ressourcen in einer Netzwerkumgebung eingeschränkt werden soll.

Firewalls haben folgende Funktionen:

- Schützen des internen Netzwerks vor unbefugtem Zugriff oder Störung
- Schützen des internen Netzwerks vor Daten und Programmen, die durch interne Benutzer eingeführt werden
- Begrenzen des Zugriffs interner Benutzer auf externe Daten
- Beschränken des möglichen Schadens bei einem möglichen Durchbrechen der Firewall

Net.Data kann mit Firewall-Produkten verwendet werden, die in Ihrer Umgebung ausgeführt werden.

Die folgenden Konfigurationsmöglichkeiten sind Empfehlungen für die Verwaltung der Sicherheit Ihrer Net.Data-Anwendung. Diese Konfigurationsmöglichkeiten enthalten wertvolle Informationen, wobei davon ausgegangen wird, dass Sie bereits eine Firewall konfiguriert haben, mit der Sie Ihr sicheres Intranet vom öffentlich zugänglichen Internet abgrenzen. Prüfen Sie sorgfältig, welche der folgenden Konfigurationen am besten für die in Ihrem Unternehmen eingesetzten Sicherheitsmaßnahmen geeignet ist:

- **Konfiguration mit hoher Sicherheit**

  Mit dieser Konfiguration wird ein Teilnetzwerk erstellt, mit dem Net.Data und der Webserver sowohl vom sicheren Intranet als auch vom öffentlichen Internet abgegrenzt werden. Die Firewall-Software wird für die Erstellung einer ersten Firewall zwischen Webserver und öffentlichem Internet sowie einer zweiten Firewall zwischen Webserver und gesichertem Intranet, in dem sich der DB2-Server befindet, verwendet. Abb. 15 zeigt diese Konfiguration.

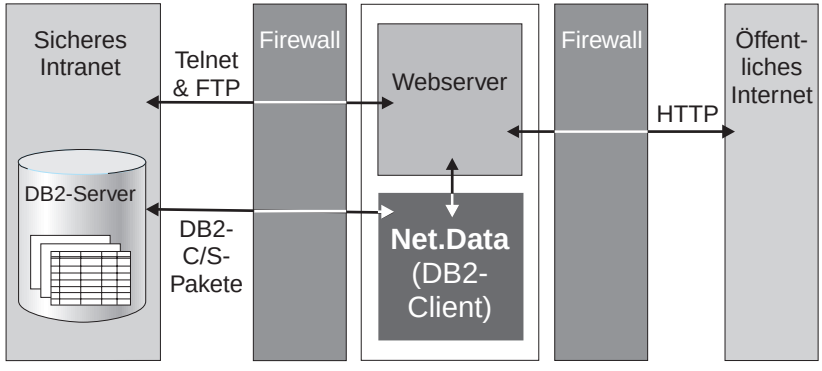


Abbildung 15. Konfiguration mit hoher Sicherheit

Gehen Sie wie folgt vor, um diese Konfiguration zu definieren:

– Installieren Sie Net.Data auf der Webserver-Maschine, und stellen Sie sicher, dass Net.Data innerhalb des Intranets auf den DB2-Server zugreifen kann.
  - Installieren Sie Client Application Enabler (CAE) auf der Webserver-Maschine.
  - Konfigurieren Sie die Firewall so, dass ein DB2-Datenverkehr durch die Firewall möglich ist. Eine Möglichkeit besteht darin, Regeln zur Paketfilterung hinzuzufügen, die DB2-Client-Anforderungen von Net.Data und Bestätigungspakete vom DB2-Server an Net.Data durchlassen.

– Erlauben Sie FTP- und Telnet-Zugriffe zwischen dem Webserver und dem sicheren Intranet. Eine Möglichkeit besteht darin, einen Socks-Server auf der Webserver-Maschine zu installieren.
– Geben Sie in der Konfigurationsdatei für die Paketfilterung der Firewall-Software an, dass eingehende TCP-Pakete vom HTTP-Standardanschluss auf den Webserver zugreifen dürfen. Geben Sie weiterhin an, dass abgehende TCP-Bestätigungspakete vom Webserver an jeden beliebigen Host des öffentlichen Internets gesendet werden dürfen.

- **Konfiguration mit mittlerer Sicherheit**

  Bei dieser Konfiguration trennt die Firewall-Software das gesicherte Intranet mit dem DB2-Server vom öffentlichen Internet. Net.Data und der Webserver befinden sich außerhalb der Firewall auf einer Workstation-Plattform. Diese Konfiguration ist einfacher als die oben beschriebene, bietet jedoch trotzdem einen Datenbankschutz. Abb. 16 zeigt diese Konfiguration.

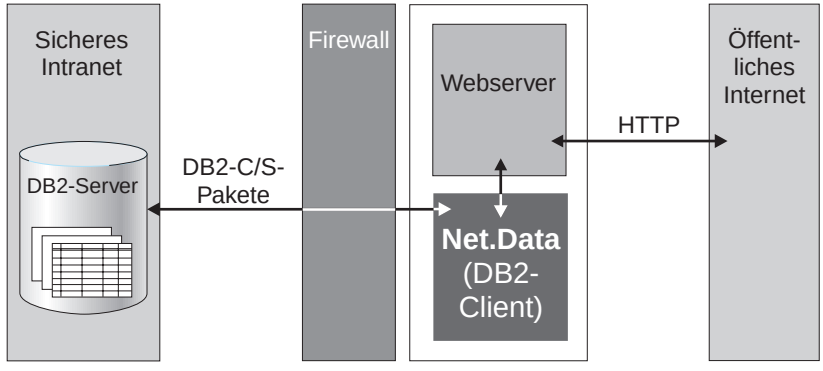


Abbildung 16. Konfiguration mit mittlerer Sicherheit

  Hierzu müssen Sie CAE auf dem Webserver installieren, damit Net.Data Übertragungen zum DB2-Server durchführen kann. Die Firewall muss so konfiguriert sein, dass DB2-Client-Anforderungen von Net.Data an DB2 weitergeleitet werden können und Bestätigungspakete von DB2 an Net.Data durchgelassen werden.

- **Konfiguration mit niedriger Sicherheit**

  Bei dieser Konfiguration werden der DB2-Server und Net.Data außerhalb der Firewall und des gesicherten Intranets installiert. Dadurch sind sie nicht gegen externe Manipulation geschützt. Für diese Konfiguration benötigt die Firewall keine Regeln zur Paketfilterung. Abb. 17 auf Seite 70 zeigt diese Konfiguration.

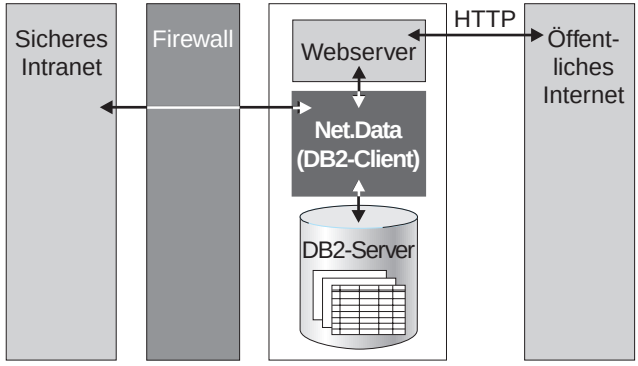


Abbildung 17. Konfiguration mit niedriger Sicherheit

Verschlüsseln Ihrer Daten im Netzwerk

Wenn Sie einen Webserver verwenden, der SSL (Secured Sockets Layer) unterstützt, können Sie alle Daten, die zwischen einem Client-System und Ihrem Webserver gesendet werden, verschlüsseln. Diese Sicherheitsmaßnahme unterstützt die Verschlüsselung von Anmelde-IDs, Kennwörtern und aller Daten, die über HTML-Formulare vom Client-System an den Webserver übertragen werden, sowie aller Daten, die vom Webserver an das Client-System gesendet werden. Die meisten Webserver unterstützen SSL.

Verwenden der Authentifizierung

Authentifizierung wird verwendet, um sicherzustellen, dass eine Benutzer-ID, die eine Net.Data-Anforderung absetzt, berechtigt ist, auf Daten in der Anwendung zuzugreifen und sie zu aktualisieren. Bei der Authentifizierung wird die Benutzer-ID mit einem Kennwort abgeglichen, um zu überprüfen, ob die Anforderung von einer gültigen Benutzer-ID stammt. Der Webserver ordnet jeder Net.Data-Anforderung, die er verarbeitet, eine Benutzer-ID zu. Der Prozess bzw. Thread, der die Anforderung bearbeitet, kann dann auf eine beliebige Ressource zugreifen, für die diese Benutzer-ID über eine entsprechende Berechtigung verfügt.

Sie können zwei Arten von Authentifizierung verwenden: eine schützt bestimmte Verzeichnisse auf Ihrem Server, die andere schützt Ihre Datenbank.

- Bei den meisten Webservern können Sie die Verzeichnisse auf dem Server angeben, die geschützt werden sollen. Ferner können Sie festlegen, dass eine Benutzer-ID und ein Kennwort angegeben werden müssen, damit der Zugriff auf Dateien in bestimmten Verzeichnissen erlaubt wird. Näheres zu den Möglichkeiten Ihres Systems finden Sie im Administratorhandbuch für Ihren Webserver.

- DB2 verfügt über ein System zur Authentifizierung für den Datenbankzugriff, mit dem der Zugriff auf Tabellen und Spalten auf bestimmte Benutzer beschränkt werden kann. Sie können Sondervariablen von Net.Data, wie LOGIN und PASSWORD, verwenden, um eine Verbindung (Link) zur Authentifizierungsroutine von DB2 herzustellen.

Hinweis: Gehen Sie wie folgt vor, um Net.Data-Makros zu schützen:

1. Fügen Sie Zugriffsschutzanweisungen für das Net.Data-Programmdateiobjekt in die Webserver-Konfigurationsdatei ein.
2. Stellen Sie sicher, dass die Benutzer-ID, unter der Net.Data ausgeführt wird, Zugriffsrechte für die Makros besitzt. Weitere Informationen zur Erteilung von Zugriffsrechten finden Sie in „Erteilen von Zugriffsrechten für Dateien, auf die Net.Data zugreift" auf Seite 65.

Verwenden der Berechtigung

Eine *Berechtigung* erlaubt einem Benutzer den vollständigen oder beschränkten Zugriff auf ein Objekt, eine Ressource oder Funktion. Datenquellen wie DB2 stellen ihre eigenen Berechtigungsmechanismen zum Schutz der von Ihnen verwalteten Daten zur Verfügung. Diese Mechanismen gehen davon aus, dass für die Benutzer-ID der Net.Data-Anforderung eine ordnungsgemäße Authentifizierung ausgeführt wurde. Eine genauere Erklärung hierzu finden Sie in „Verwenden der Authentifizierung" auf Seite 70. Die vorhandenen Zugriffssteuerungsmechanismen für diese Datenquellen erteilen bzw. verweigern dann je nach den Berechtigungen der überprüften Benutzer-ID den Zugriff.

Verwenden von Net.Data-Mechanismen

Zusätzlich zu den oben beschriebenen Methoden können Sie mit Net.Data-Konfigurationsvariablen oder Makro-Entwicklungsverfahren die Aktivitäten von Endbenutzern begrenzen, um firmeninterne Informationen wie den Aufbau Ihrer Datenbank zu verdecken und um vom Benutzer gestellte Eingabewerte in Produktionsumgebungen zu überprüfen.

Net.Data-Konfigurationsvariablen

Net.Data stellt mehrere Konfigurationsvariablen zur Verfügung, mit denen Sie die Aktivitäten von Endbenutzern begrenzen oder den Aufbau Ihrer Datenbank verdecken können.

Steuern des Dateizugriffs mit Pfadanweisungen

Net.Data überprüft die Einstellungen der Pfadkonfigurationsanweisungen, um die Speicherposition der Dateien und ausführbaren Programme zu ermitteln, die von Net.Data-Makros verwendet werden. Diese Pfadanweisungen geben mindestens ein Verzeichnis an, das Net.Data durchsucht, wenn es versucht, Makros, ausführbare

Dateien, Kopfdateien oder andere unstrukturierte Dateien zu lokalisieren. Durch die selektive Aufnahme von Verzeichnissen in diese Pfadanweisungen können Sie explizit steuern, auf welche Dateien die Benutzer über Browser zugreifen können. Zusätzliche Informationen zu Pfadanweisungen finden Sie in „Kapitel 2. Konfigurieren von Net.Data" auf Seite 5.

Sie sollten zudem Berechtigungsprüfungen verwenden (siehe „Verwenden der Berechtigung" auf Seite 71) und sicherstellen, dass Dateinamen in INCLUDE-Anweisungen nicht geändert werden können (siehe „Makro-Entwicklungsverfahren" auf Seite 73).

Inaktivieren von SHOWSQL für Produktionssysteme

Mit der Variable SHOWSQL kann der Benutzer angeben, dass Net.Data die in Net.Data-Funktionen angegebenen SQL-Anweisungen in einem Web-Browser anzeigt. Diese Variable wird hauptsächlich zum Entwickeln und Testen von SQL in einer Anwendung verwendet und wurde nicht zur Verwendung in Produktionssystemen konzipiert.

Sie können die Anzeige von SQL-Anweisungen in Produktionsumgebungen mit einer der folgenden Methoden inaktivieren:

- Bei der Verwendung von Net.Data Version 2.0.7 oder höher können Sie mit der Konfigurationsvariablen DTW_SHOWSQL in der Net.Data-Initialisierungsdatei die Auswirkung der Einstellung SHOWSQL in Ihren Net.Data-Makros überschreiben. Informationen zur Syntax und andere nützliche Informationen finden Sie in „DTW_SHOWSQL: Konfigurationsvariable zum Aktivieren bzw. Inaktivieren von SHOWSQL" auf Seite 20.
- Benutzer von Net.Data Version 2.0.5 und früher können die Funktion DTW_ASSIGN() verwenden (siehe „Makro-Entwicklungsverfahren" auf Seite 73).

Informationen zur Syntax und Beispiele für die Net.Data-Variable SHOWSQL finden Sie unter SHOWSQL im Kapitel über Variablen des Handbuchs *Net.Data Reference*.

Erwägen der Aktivierung von Direktanforderungen für Produktionsumgebungen

Bei der Direktanforderungsmethode zum Aufrufen von Net.Data kann ein Benutzer die Ausführung einer SQL-Anweisung bzw. eines Perl-, REXX- oder C-Programms direkt in einer URL-Adresse angeben. Bei der Makroanforderungsmethode können Benutzer nur jene SQL-Anweisungen und Funktionen ausführen, die in einem Makro definiert sind bzw. aufgerufen werden.

Überlegen Sie sorgfältig, ob Sie die Verwendung von Direktanforderungen zulassen wollen, denn dadurch sind Ihre Benutzer eventuell in der Lage, sehr viele Funktionen auszuführen. Wenn Sie diese

Aufrufmethode aktivieren, stellen Sie sicher, dass der Benutzer-ID, unter der die Net.Data-Anforderung verarbeitet wird, die entsprechende Berechtigungsstufe erteilt wurde.

Direktanforderungen können mit der Konfigurationsvariablen DTW_DIRECT_REQUEST inaktiviert werden. Informationen zur Syntax und andere nützliche Informationen finden Sie in „DTW_DIRECT_REQUEST: Variable zum Aktivieren der Direktanforderung" auf Seite 18.

Kennwortverschlüsselung

Wenn Sie LOGIN und PASSWORD in der Konfigurationsdatei der Direktverbindung (dtwcm.cnf) und/oder in Net.Data-Makros angeben, sollten Sie das Kennwort schützen, indem Sie es verschlüsseln.

Gehen Sie wie folgt vor, um die Verschlüsselung zu aktivieren, damit z. B. ein Kennwort mit unverschlüsseltem Text durch ein verschlüsseltes Kennwort ersetzt wird:

- Für die Datei dtwcm.cnf:
  1. Legen Sie ENCRYPTION=<key> fest.
  2. Verwenden Sie den Befehl 'dtwcm -p', um den verschlüsselten Kennworteintrag zu generieren.
  3. Schneiden Sie diese verschlüsselte Zeichenfolge aus, und fügen Sie sie in den Eintrag PASSWORD in der Datei dtwcm.cnf ein.
- Für Makros:
  1. Legen Sie ENCRYPTION=<key> in der Datei db2www.ini fest. Dieser Schlüssel muss derselbe sein, wie der Schlüssel in der Datei dtwcm.cnf.
  2. Verwenden Sie den Befehl 'dtwcm -p', um den verschlüsselten Kennworteintrag zu generieren.
  3. Schneiden Sie dieses verschlüsselte Kennwort aus, und fügen Sie es in den Eintrag PASSWORD in dem Net.Data-Makro ein.

Gehen Sie wie folgt vor, um die Verschlüsselung zu inaktivieren:

- Entfernen Sie die Zeile mit ENCRYPTION aus der Datei dtwcm.cnf.
- Entfernen Sie bei Makros die Zeile mit ENCRYPTION aus der Datei db2www.ini.

Makro-Entwicklungsverfahren

Net.Data stellt mehrere Mechanismen bereit, mit denen Benutzer Eingabevariablen Werte zuordnen können. Sie können sicherstellen, dass Makros auf die gewünschte Art ausgeführt werden, indem Sie diese Eingabevariablen durch das Makro prüfen lassen. Ihre Datenbank und Anwendung sollten zudem so entworfen werden, dass der Zugriff von Benutzern auf Daten, zu deren Anzeige sie berechtigt sind, begrenzt ist.

Verwenden Sie beim Schreiben Ihrer Net.Data-Makros folgende Entwicklungsverfahren. Diese Verfahren tragen dazu bei, dass Ihre Anwendungen wie gewünscht ausgeführt werden und dass der Datenzugriff auf Benutzer mit entsprechender Berechtigung begrenzt ist.

Sicherstellen, dass Net.Data-Variablen in einer URL-Adresse nicht überschrieben werden können

Die Einstellung von Net.Data-Variablen durch einen Benutzer in einer URL-Adresse überschreibt die Auswirkung von DEFINE-Anweisungen, mit denen Variablen in einem Makro initialisiert werden. Dadurch wird eventuell die Art der Makroausführung geändert. Sie können dies verhindern, indem Sie Ihre Net.Data-Variablen mit der Funktion DTW_ASSIGN() initialisieren.

Beispiel: Verwenden Sie nicht den folgenden Befehl:

```
%define START_ROW_NUM = "1"
```

Verwenden Sie stattdessen diesen Befehl:

```
@DTW_ASSIGN(START_ROW_NUM, "1")
```

Das Zuordnen der Variable auf diese Weise verhindert, dass eine Zuweisung einer Abfragezeichenfolge, wie z. B. ″START_ROW-_NUM=10″, Ihre Makro-Einstellung überschreibt.

Sicherstellen, dass Ihre SQL-Anweisungen nicht so geändert werden können, dass sich das beabsichtigte Verhalten Ihrer Anwendung ändert

Durch das Hinzufügen einer Net.Data-Variablen zu einer SQL-Anweisung in einem Makro können Benutzer die SQL-Anweisung vor ihrer Ausführung dynamisch ändern. Der Makroautor ist dafür verantwortlich, die vom Benutzer gestellten Eingabewerte zu prüfen und sicherzustellen, dass eine SQL-Anweisung mit einem Variablenverweis nicht auf unerwartete Weise geändert wird. Ihre Net.Data-Anwendung sollte vom Benutzer gestellte Eingabewerte in der URL-Adresse prüfen, so dass die Net.Data-Anwendung ungültige Eingaben zurückweisen kann. Beim Entwerfen einer Gültigkeitsprüfung sollten Sie die folgenden Schritte ausführen:

1. Geben Sie die Syntax für gültige Eingaben an. Zum Beispiel muss eine Kunden-ID mit einem Buchstaben anfangen und kann nur alphanumerische Zeichen enthalten.
2. Ermitteln Sie, welcher Schaden durch versehentliche oder absichtliche falsche Eingaben sowie durch Eingaben, durch die auf interne Daten der Net.Data-Anwendung zugegriffen werden soll, verursacht werden kann.

3. Nehmen Sie in das Makro Eingabeprüfungsanweisungen auf, die die Anforderungen der Anwendung erfüllen. Eine derartige Prüfung hängt von der Syntax der Eingabe und ihrer Verwendungsweise ab. In einfacheren Fällen reicht es eventuell aus, die Eingabe auf ungültigen Inhalt zu überprüfen oder Net.Data aufzurufen, um den Typ der Eingabe zu prüfen. Wenn die Syntax der Eingabe komplexer gestaltet ist, muss der Makroentwickler eventuell die Eingabe teilweise oder vollständig syntaktisch analysieren, um zu prüfen, ob sie gültig ist.

Beispiel 1: Verwenden der Zeichenfolgefunktion DTW_POS() zum Prüfen von SQL-Anweisungen

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

Der Wert der Variable `shlogid` soll eine Käufer-ID (shopper) sein. Durch diese Variable soll die Anzahl der Zeilen, die durch die Anweisung SELECT zurückgegeben wird, auf die Zeilen begrenzt werden, die Informationen zum durch die Käufer-ID angegebenen Käufer enthalten. Wenn jedoch die Zeichenfolge „`smith' or shlogid<>'smith`" als Wert der Variable `shlogid` übergeben wird, sieht die Abfrage wie folgt aus:

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

Diese vom Benutzer geänderte Version der ursprünglichen SQL-Anweisung SELECT gibt die gesamte Käufertabelle zurück.

Mit den Net.Data-Zeichenfolgefunktionen können Sie sicherstellen, dass die SQL-Anweisung nicht auf unerwünschte Weise durch den Benutzer geändert wird. Sie können z. B. mit der folgenden Logik sicherstellen, dass keine einfachen Anführungszeichen verwendet werden, um SQL-Anweisungen zu ändern:

```
@DTW_ADDQUOTE(shlogid, shlogid)
  @query1()
```

Die Abfrage hat dann die folgende Form:

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

Beispiel 2: Verwenden von DTW_TRANSLATE()

Angenommen, Ihre Anwendung muss sicherstellen, dass der in der Eingabevariablen `num_orders` bereitgestellte Wert eine ganze Zahl ist. Hierzu können Sie die Umsetztabelle `trans_table` erstellen, die alle Tastaturzeichen mit Ausnahme der numerischen Zeichen 0-9 enthält, und die Eingabe mit den Zeichenfolgefunktionen DTW_TRANSLATE und DTW_POS prüfen:

```
@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
```

Beachten Sie, dass SQL-Anweisungen in gespeicherten Prozeduren nicht durch Benutzer über Web-Browser geändert werden können und dass vom Benutzer gestellte Eingabeparameterwerte durch die SQL-Datentypen, die den Eingabeparametern zugeordnet sind, beschränkt sind. In Situationen, in denen es unpraktisch ist, Benutzereingabewerte mit den Net.Data-Zeichenfolgefunktionen zu prüfen, können Sie gespeicherte Prozeduren verwenden.

Sicherstellen, dass ein Dateiname in einer INCLUDE-Anweisung nicht so geändert wird, dass sich das beabsichtigte Verhalten Ihrer Anwendung ändert Wenn Sie den Wert für den Dateinamen mit einer INCLUDE-Anweisung bei Verwendung einer Net.Data-Variablen angeben, dann wird die aufzunehmende Datei erst während der Ausführung der INCLUDE-Datei ermittelt. Soll der Wert dieser Variablen in Ihrem Makro festgelegt werden, ein Benutzer jedoch nicht in der Lage sein, den vom Makro bereitgestellten Wert über einen Browser zu überschreiben, dann legen Sie den Wert der Variablen mit DTW_ASSIGN anstelle von DEFINE fest. Wenn der Benutzer in der Lage sein soll, über einen Browser einen Wert für den Dateinamen bereitzustellen, dann sollte Ihr Makro den angegebenen Wert prüfen.

Beispiel: Eine Abfragezeichenfolgezuordnung wie `filename="../../x"` kann zur Aufnahme einer Datei aus einem Verzeichnis führen, das normalerweise nicht in der Konfigurationsanweisung INCLUDE-_PATH angegeben ist. Angenommen, Ihre Net.Data-Initialisierungsdatei enthält die folgende Pfadkonfigurationsanweisung:

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

Und angenommen, Ihr Net.Data-Makro enthält die folgende INCLUDE-Anweisung:

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

Die Abfragezeichenfolgezuordnung `filename="../../x"` würde die Datei `/usr/lpp/x` enthalten, was durch die Angabe der Konfigurationsanweisung INCLUDE_PATH nicht beabsichtigt war.

Mit den Net.Data-Zeichenfolgefunktionen können Sie prüfen, ob der bereitgestellte Dateiname für die Anwendung geeignet ist. Sie können z. B. mit der folgenden Logik sicherstellen, dass der Eingabewert, der der Dateinamenvariablen zugeordnet ist, nicht die Zeichenfolge "../" enthält:

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

Entwerfen Ihrer Datenbank und Abfragen, so dass Benutzeranforderungen keinen Zugriff auf sensible Daten anderer Benutzer haben

Einige Datenbankentwürfe sammeln sensible Benutzerdaten in einer einzigen Tabelle. Sofern SQL-Anforderungen SELECT nicht auf eine gewisse Art qualifiziert sind, hat diese Vorgehensweise zur Folge, dass ein beliebiger Benutzer über einen Web-Browser eventuell Zugriff auf alle sensiblen Daten hat.

Beispiel: Die folgende SQL-Anweisung gibt Auftragsinformationen zu einem durch die Variable `order_rn` angegebenen Auftrag zurück:

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

Diese Methode ermöglicht Benutzern, über einen Browser willkürliche Auftragsnummern anzugeben und möglicherweise sensible Informationen zu den Aufträgen anderer Kunden abzurufen. Sie können sich hiervor zum Beispiel schützen, indem Sie die folgenden Änderungen vornehmen:

- Fügen Sie der Auftragsinformationstabelle eine Spalte hinzu, die den Kunden angibt, der den Auftragsinformationen in einer bestimmten Zeile zugeordnet ist.
- Ändern Sie die SQL-Anweisung SELECT, um sicherzustellen, dass SELECT durch eine authentifizierte, vom Benutzer über den Browser bereitgestellte Kunden-ID qualifiziert ist.

Wenn z. B. `shlogid` die Spalte mit der Kunden-ID ist, die dem Auftrag zugeordnet ist, und wenn SESSION_ID eine Net.Data-Variable ist, welche die authentifizierte ID des Benutzers am Browser enthält, dann können Sie die vorherige SELECT-Anweisung durch die folgende Anweisung ersetzen:

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

Verwenden verdeckter Net.Data-Variablen

Mit verdeckten Net.Data-Variablen können Sie verschiedene Kenndaten Ihrer Net.Data-Makros vor Benutzern schützen, die Ihre HTML-Quelle mit ihrem Web-Browser anzeigen möchten. Sie können zum Beispiel die interne Struktur Ihrer Datenbank verdecken. Weitere Informationen zu verdeckten Variablen finden Sie in „Verdeckte Variablen” auf Seite 116.

Anfordern von Informationen zur Gültigkeitsprüfung von einem Benutzer

Sie können Ihr eigenes Schutzschema basierend auf der vom Benutzer gestellten Eingabe erstellen. Beispielsweise könnten Sie anhand eines HTML-Formulars Informationen zur Gültigkeitsprüfung von einem Benutzer anfordern und diese Informationen dann anhand von Daten, die Ihr Net.Data-Makro aus einer Datenbank abruft, oder durch Aufrufen eines externen Programms aus einer Funktion, die in Ihrem Net.Data-Makro definiert ist, überprüfen lassen.

Weitere Informationen zum Schutz Ihrer Datenbestände enthält die Internet-Liste zu häufig gestellten Sicherheitsfragen, die Sie unter folgender Webadresse finden:

`http://www.w3.org/Security/Faq`

Kapitel 4. Aufrufen von Net.Data

In diesem Kapitel wird beschrieben, wie Sie Net.Data mit den verschiedenen Webserver-Schnittstellen aufrufen können. Vor der Verwendung einer dieser Aufrufmethoden muss Net.Data zuerst für die angegebene Schnittstelle konfiguriert werden. Sie können Net.Data für die Verwendung der folgenden Webserver-Schnittstellen konfigurieren:

- Common Gateway Interface (CGI)
- FastCGI
- Apache API (APAPI)
- IBM HTTP Server API
- Netscape Server (NSAPI)
- Microsoft Internet Server (ISAPI)
- Java-Servlets

Weitere Informationen zum Konfigurieren von Net.Data für diese Schnittstellen finden Sie in „Kapitel 2. Konfigurieren von Net.Data" auf Seite 5. Der Webserver ruft Net.Data standardmäßig als CGI-Programm auf, wobei jede Net.Data-Anforderung in einem neuen und separaten Prozess ausgeführt wird. Sie legen fest, wie Net.Data aufgerufen wird, wenn Sie den Webserver konfigurieren.

In den folgenden Abschnitten werden die von Net.Data akzeptierten Anforderungsarten und die Methoden zum Aufrufen von Net.Data mit den verschiedenen APIs und Servlets beschrieben.

- „Arten von Aufrufanforderungen"
- „Aufrufen von Net.Data über die Webserver-APIs" auf Seite 92

Arten von Aufrufanforderungen

Unabhängig von der von Ihnen gewählten Methode zum Aufrufen von Net-.Data können Sie eine von zwei Anforderungsarten angeben.

Makroanforderung
Gibt an, dass Net.Data das angegebene Makro ausführt.

Direktanforderung
Gibt an, dass Net.Data eine SQL-Anweisung, gespeicherte Prozedur oder Funktion ausführt.

Webentwickler, die eine einzelne SQL-Abfrage schreiben oder eine einzelne Funktion wie eine gespeicherte DB2-Prozedur, ein REXX-Programm oder eine

Perl-Funktion aufrufen wollen, können eine Direktanforderung an die Datenbank absetzen. Eine Direktanforderung hat keine komplexe Net.Data-Anwendungslogik, die ein Net.Data-Makro erfordert. Daher umgeht sie den Net.Data-Makroumwandler. Die Parameter der Direktanforderung werden zur Verarbeitung an die entsprechende Sprachumgebung übermittelt, um die Leistung zu steigern.

Abb. 18 verdeutlicht die Unterschiede zwischen einer Makroanforderung und einer Direktanforderung. Eine Makroanforderung gibt immer ein Makro innerhalb der URL-Adresse für die Anforderung an und kann auch Formulardaten verwenden. Eine Direktanforderung gibt nie ein Makro innerhalb der URL-Adresse an, kann jedoch weiterhin Formulardaten verwenden.

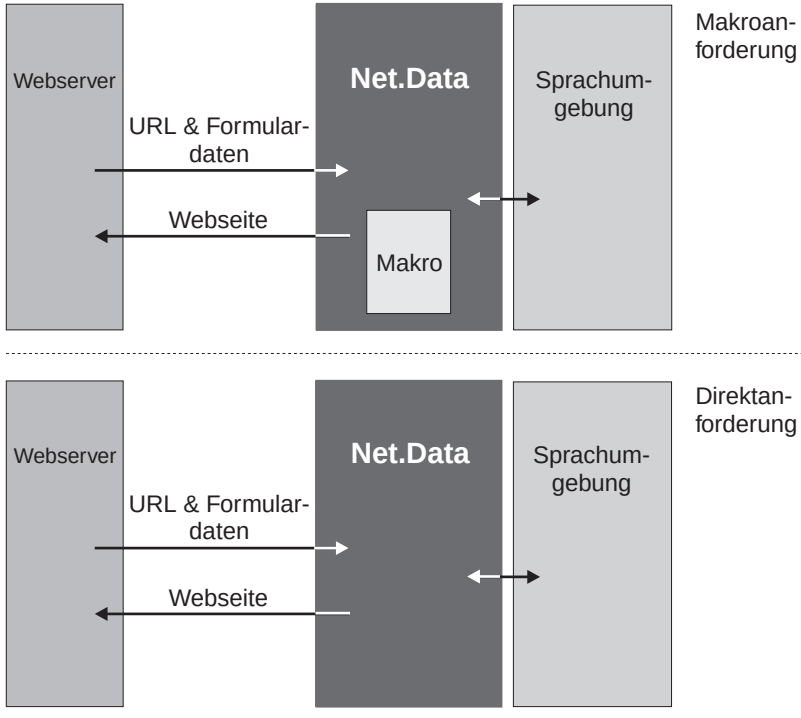


Abbildung 18. Makroanforderung im Vergleich zu Direktanforderung

Die Syntax zum Aufrufen von Net.Data hängt von der Net.Data-Konfiguration und der Art Ihrer Anforderung ab. Bei Makro- und Direktanforderungen wird Net.Data mit einer URL-Adresse aufgerufen. Die URL-Adresse kann direkt vom Benutzer eingegeben oder als HTML-Verbindung (Link) oder als HTML-Formular in die HTML-Seite codiert werden. Der Webserver ruft Net.Data über CGI, FastCGI oder eine der Webserver-APIs auf.

Geben Sie bei Makroanforderungen in der URL-Adresse den Namen des Net.Data-Makros und den Namen des im Net.Data-Makro auszuführenden HTML-Blocks an. Geben Sie bei Direktanforderungen in der URL-Adresse den Namen der Net.Data-Sprachumgebung, die SQL-Anweisung bzw. den Namen der Funktion und andere zusätzlich erforderliche Parameterwerte an. Sie geben diese Werte in einer von Net.Data definierten Syntax an. Wenn Sie von CGI auf APAPI oder FastCGI migrieren, sollten Sie einige Punkte der REXX-Sprachumgebung bedenken. Weitere Informationen finden Sie in „REXX-Sprachumgebung" auf Seite 189.

In den folgenden Abschnitten werden diese Aufrufanforderungen ausführlicher beschrieben:

- „Aufrufen von Net.Data mit einem Makro (Makroanforderung)"
- „Aufrufen von Net.Data ohne Makro (Direktanforderung)" auf Seite 86

Die Beispiele geben zwar die zu verwendende Syntax beim Aufrufen von Net.Data über CGI an, die Konzepte gelten jedoch für alle Schnittstellen, mit denen Net.Data aufgerufen werden kann. Informationen zur genauen Syntax, die für die einzelnen Schnittstellenarten erforderlich ist, finden Sie im entsprechenden Abschnitt.

- „Aufrufen von Net.Data über die Webserver-APIs" auf Seite 92

Aufrufen von Net.Data mit einem Makro (Makroanforderung)

Ein Client-Browser ruft Net.Data durch Senden einer Anforderung in Form einer URL auf. In diesem Abschnitt wird gezeigt, wie Sie Net.Data durch Angabe eines Makros in der URL-Anforderung aufrufen können.

Die an Net.Data gesendete Anforderung hat das folgende Format:

```
http://server/Net.Data_invocation_path/filename/block[?name=val&...]
```

Parameter:

server Gibt den Namen und Pfad des Webservers an. Wenn es sich bei dem Server um den lokalen Server handelt, kann der Servername übergangen und eine relative URL-Adresse verwendet werden.

Net.Data_invocation_path
Gibt Pfad und Dateinamen der ausführbaren Net.Data-Datei, Servlet-Klasse, DLL oder gemeinsam benutzten Bibliothek an. Zum Beispiel `/cgi-bin/db2www/`.

filename
Gibt den Namen der Net.Data-Makrodatei an. Net.Data sucht diesen Dateinamen und versucht, ihn mit den Pfadanweisungen abzugleichen, die in der Initialisierungspfadvariablen MACRO_PATH definiert sind. Weitere Informationen hierzu finden Sie in „MACRO_PATH" auf Seite 28.

block Gibt den Namen des HTML-Blocks im angegebenen Net.Data-Makro an.

?*name***=***val***&...**
Gibt mindestens einen optionalen Parameter an, der an Net.Data weitergegeben wird.

Sie geben diese URL direkt im Browser ein. Sie können sie auch wie folgt in einer HTML-Verbindung (Link) angeben bzw. unter Verwendung eines der folgenden Formate erstellen:

- HTML-Verbindung (Link):

```
<a href="URL">any text</a>
```

- HTML-Formular:

```
<form method="method" ACTION="URL">any text</form>
```

Parameter:

method Gibt die für das Formular verwendete HTML-Methode an.

URL Gibt die URL-Adresse an, mit der das Net.Data-Makro ausgeführt wird und deren Parameter oben beschrieben sind.

Beispiele

Die folgenden Beispiele veranschaulichen die verschiedenen Methoden zum Aufrufen von Net.Data.

Beispiel 1: Aufrufen von Net.Data mit einer HTML-Verbindung (Link)

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

Beispiel 2: Aufrufen von Net.Data mit einem Formular

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

In den folgenden Abschnitten werden HTML-Verbindungen (Links) und -Formulare und der Aufruf von Net.Data mit diesen Mitteln beschrieben:

- „HTML-Verbindungen (Links)" auf Seite 83
- „HTML-Formulare" auf Seite 84

HTML-Verbindungen (Links)

Wenn Sie eine Webseite verfassen, können Sie eine HTML-Verbindung (Link) erstellen, die zur Ausführung eines HTML-Blocks führt. Wenn ein Benutzer über einen Browser den Text bzw. das Bild anklickt, der bzw. das als eine HTML-Verbindung (Link) definiert ist, führt Net.Data den HTML-Block im Makro aus.

Verwenden Sie den HTML-Befehl <a>, um eine HTML-Verbindung (Link) zu erstellen. Entscheiden Sie, welcher Text bzw. welche Grafik als Hyperlink zum Net.Data-Makro verwendet werden soll. Stellen Sie ihm bzw. ihr dann den Befehl <a> vor und den Befehl nach. Geben Sie im Attribut HREF des Befehls <a> das Makro und den HTML-Block an.

Im folgenden Beispiel wird eine Verbindung (Link) gezeigt, die zur Ausführung einer SQL-Abfrage führt, wenn ein Benutzer den Text ″List all monitors″ auf einer Webseite auswählt.

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

Durch das Anklicken der Verbindung (Link) wird das Makro listA.dtw mit dem HTML-Block ″report″ aufgerufen. Siehe folgendes Beispiel:

```
%DEFINE DATABASE="MNS97"

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

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

Diese Abfrage gibt eine Tabelle mit der Modellnummer (MODNO), dem Preis (COST) und einer Beschreibung (DESCRIP) für jeden Monitor zurück, der in der Tabelle EQPTABLE beschrieben ist. Der Wert von `hdware` in der SQL-Anweisung stammt aus der URL-Eingabe. Eine detaillierte Beschreibung der im ROW-Block verwendeten Variablen finden Sie im Handbuch *Net.Data Reference*.

HTML-Formulare

Sie können die Ausführung Ihrer Net.Data-Makros mit HTML-Formularen dynamisch anpassen. Formulare ermöglichen Benutzern die Angabe von Eingabewerten, die die Ausführung des Makros und den Inhalt der von Net.Data erstellten Webseite beeinflussen.

Das folgende Beispiel baut auf dem Beispiel zur Monitorliste aus „HTML-Verbindungen (Links)" auf Seite 83 auf, indem es Benutzern ermöglicht, in einem Browser mit Hilfe eines einfachen HTML-Formulars den Produkttyp auszuwählen, für den Informationen angezeigt werden sollen.

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

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

Nachdem der Benutzer im Browser seine Auswahl getroffen und den Knopf für die Übergabe angeklickt hat, verarbeitet der Webserver den Parameter ACTION des Befehls FORM, wodurch Net.Data aufgerufen wird. Dann führt Net.Data das Makro listA.dtw aus, das über einen HTML-Block mit dem Namen ″report″ (siehe oben) verfügt.

```
%DEFINE DATABASE="MNS97"

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

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

Im obigen Beispiel stammt der Wert für `hdware` der SQL-Anweisung aus der HTML-Formulareingabe. Eine detaillierte Beschreibung der im ROW-Block verwendeten Variablen finden Sie im Handbuch *Net.Data Reference*.

Der Eingabetyp FILE wird von Net.Data auf besondere Weise verarbeitet. Mit diesem Eingabetyp können Benutzer eine Datei auf den Server hochladen, die dann von Net.Data oder einer anderen Anwendung auf dem Server weiter verarbeitet werden kann.

Net.Data führt keine Konvertierung an den hochgeladenen Dateien aus, sie werden als Binärdaten behandelt. Die hochgeladenen Dateien werden in dem in DTW_UPLOAD_DIR angegebenen Verzeichnis gespeichert. Dabei erhalten sie einen eindeutigen Namen, der anhand der folgenden Regeln bestimmt wird:

Syntax:

MacroFileName + '.' + *FormVarName* + '.' + *UniqueIdentifier* + '.' + *FormFileName*

MacroFileName
: Der Name des Makros, das die Anforderung verarbeitet (das im Formular genannte Makro). Nur der Dateiname wird verwendet, nicht der vollständige Pfad.

FormVarName
: Der Name der Variablen, die im Formular zur Angabe der Datei verwendet wird.

UniqueIdentifier
: Eine Zeichenfolge, die die Eindeutigkeit sicherstellt.

Beispiel:

Definieren Sie zuerst DTW_UPLOAD_DIR in der Net.Data-Initialisierungsdatei:

```
DTW_UPLOAD_DIR /tmp/uploads
```

Erstellen Sie anschließend ein Formular, das ein Makro aufruft und mindestens einen Eingabebefehl des Typs *file* verwendet.

```
<form method="post" enctype="multipart/form-data"
      action="/netdatadev/form.dtw/report">
Name:  <input type="text" name="name" /><br />
Zip code:  <input type="text" name="zipno" /><br />
Resume:  <input type="file" name="resume" /><br />
  <input type="submit" />
</form>
```

Würde ein Benutzer das Formular mit Angabe der Datei `myresume.txt` übergeben, würde die Ergebnisdatei mit einem Namen auf den Server geschrieben, der etwa wie folgt aussieht:

```
/tmp/uploads/form.dtw.resume.20010108112341275-6245-021.myresume.txt
```

Aufrufen von Net.Data ohne Makro (Direktanforderung)

In diesem Abschnitt wird gezeigt, wie Sie Net.Data mit einer *Direktanforderung* aufrufen können. Bei Verwendung einer Direktanforderung geben Sie in der URL-Adresse nicht den Namen eines Makros an. Geben Sie anstelle dessen die Net.Data-Sprachumgebung, die SQL-Anweisung bzw. ein auszuführendes Programm und andere zusätzlich erforderliche Parameterwerte unter Verwendung einer von Net.Data definierten Syntax in der URL-Adresse an. Informationen zum Aktivieren und Inaktivieren von Direktanforderungen finden Sie in „DTW_DIRECT_REQUEST: Variable zum Aktivieren der Direktanforderung” auf Seite 18.

Die SQL-Anweisung bzw. das Programm und andere angegebene Parameter werden zur Verarbeitung direkt an die bezeichnete Sprachumgebung übergeben. Direktanforderungen verbessern die Leistung, weil Net.Data kein Makro zu lesen und zu verarbeiten braucht. Die von Net.Data bereitgestellten Sprachumgebungen SQL, ODBC, Oracle, Java, SYSTEM, Perl und REXX unterstützen Direktanforderungen. Sie können Net.Data mit einer URL-Adresse, einem HTML-Formular oder einer HTML-Verbindung (Link) aufrufen.

Eine Direktanforderung ruft Net.Data durch Übergeben der Parameter in der Abfragezeichenfolge der URL-Adresse oder der Formulardaten auf. Das folgende Beispiel verdeutlicht den Kontext, in dem Sie eine Direktanforderung angeben.

```
<a href="http://server/cgi-bin/db2www/?direct_request">any text</a>
```

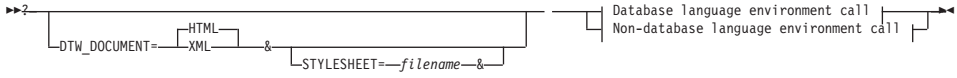
Dabei steht *direct_request* für die Direktanforderungssyntax. Die folgende HTML-Verbindung (Link) enthält zum Beispiel eine Direktanforderung:

```
<a href="http://server/cgi-bin/db2www/?LANGENV=DTW_PERL&FUNC=my_perl(hi)">
  any text</a>
```

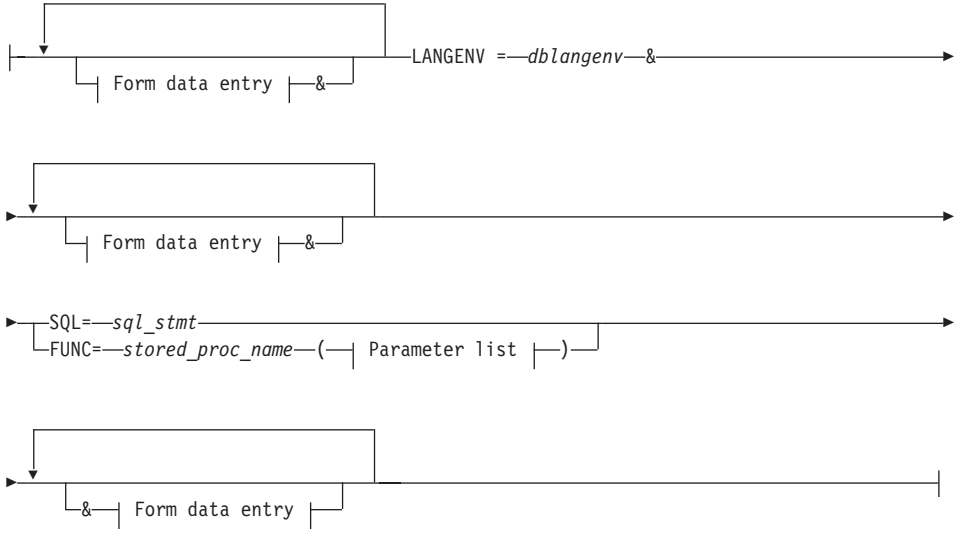
Syntax für Direktanforderungen

Die Syntax für das Aufrufen von Net.Data mit einer Direktanforderung kann den Aufruf einer Datenbank- bzw. Nicht-Datenbanksprachumgebung enthalten.

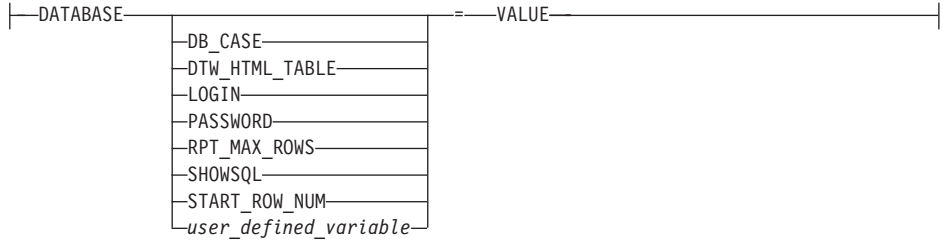
Syntax



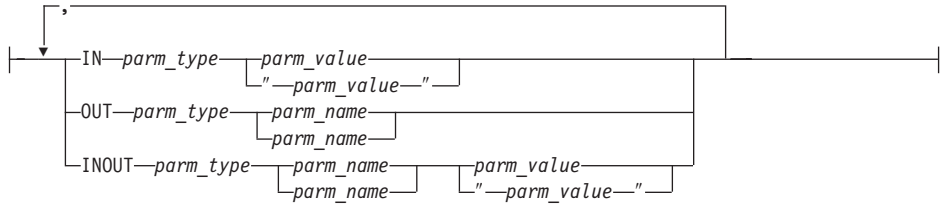
Database language environment call:



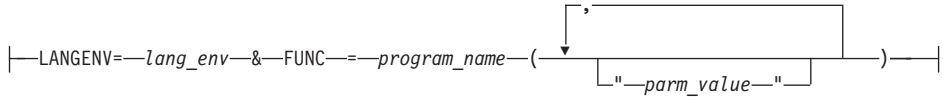
Formulardateneingabe (Form data entry):



Parameterliste (Parameter list):



Aufruf einer Nicht-Datenbanksprachumgebung (Non-database language environment call):



Parameter

DTW_DOCUMENT
Gibt die Art des Dokuments an, das Net.Data als Ausgabe zurückgeben soll. Zulässige Werte sind XML und HTML. Dieser Parameter ist optional; wird er nicht angegeben, wird HTML angenommen.

DTW_STYLESHEET
Gibt die Formatvorlage an, die Net.Data zum Anzeigen von XML verwenden soll. Dieser Parameter ist optional und nur relevant, wenn DTW_DOCUMENT=XML.

stylesheet
Gibt den Dateinamen auf dem Server für die Formatvorlage an.

Database language environment call
Gibt eine Direktanforderung an Net.Data an, die eine Datenbanksprachumgebung aufruft.

Form data entry
Parameter, mit denen Sie die Einstellungen von SQL-Variablen angeben bzw. einfache HTML-Formatierung anfordern können. Ausführlichere Informationen zu diesen Variablen finden Sie im Handbuch *Net.Data Reference* im entsprechenden Kapitel über Variablen.

DATABASE
Gibt die Datenbank an, an die Net.Data die SQL-Anforderung übergeben soll. Dieser Parameter ist erforderlich.

DB_CASE
Gibt die Groß-/Kleinschreibung für SQL-Anweisungen an.

DTW_HTML_TABLE
Gibt an, ob Net.Data eine HTML-Tabelle oder eine vorformatierte Texttabelle zurückgeben soll.

DTW_DOCUMENT
Gibt an, ob Net.Data die Ergebnisse im XML- oder HTML-Format anzeigen soll. Zulässige Werte sind XML und HTML. HTML ist der Standardwert, wenn kein Schlüsselwort angegeben wird.

LOGIN
Gibt die Benutzer-ID für die Datenbank an.

PASSWORD
Gibt das Kennwort für die Datenbank an.

RPT_MAX_ROWS
Gibt die maximale Anzahl von Zeilen an, die eine Funktion im Bericht zurückgibt.

SHOWSQL
Gibt an, ob Net.Data die auszuführende SQL-Anweisung anzeigen oder verdecken soll.

START_ROW_NUM
Gibt die Nummer der Zeile an, bei der eine Funktion den Bericht beginnen soll.

user_defined_variable
Variablen, die an Net.Data übergeben werden und erforderliche Informationen bereitstellen oder das Verhalten von Net.Data beeinflussen. Benutzerdefinierte Variablen sind Variablen, die Sie selbst für Ihre Anwendungen definieren können.

VALUE
Gibt den Wert der Net.Data-Variablen an.

LANGENV
Gibt die Zielsprachumgebung für den Aufruf der SQL-Anweisung bzw. der gespeicherten Prozedur an. Wenn es sich bei der Sprachumgebung um eine der Datenbanksprachumgebungen handelt, muss der Datenbankname ebenfalls angegeben werden.

dblangenv
Der Name der Datenbanksprachumgebung:

- DTW_SQL
- DTW_ODBC
- DTW_ORA

SQL
Gibt an, dass die Direktanforderung die Ausführung einer Inline-SQL-Anweisung angibt.

sql_stmt
Gibt eine Zeichenfolge an, die eine gültige SQL-Anweisung enthält, die mit dynamischem SQL ausgeführt werden kann.

FUNC
Gibt an, dass die Direktanforderung die Ausführung einer gespeicherten Prozedur angibt.

stored_proc_name
Gibt einen gültigen Namen für eine gespeicherte DB2-Prozedur an.

parm_type
Gibt eine gültige Parameterart für eine gespeicherte DB2-Prozedur an.

parm_name
Gibt einen gültigen Parameternamen an.

parm_value
Gibt einen gültigen Parameterwert für eine gespeicherte DB2-Prozedur an.

IN Gibt an, dass Net.Data den Parameter zum Übergeben von Eingabedaten an die gespeicherte Prozedur verwenden muss.

INOUT
Gibt an, dass Net.Data den Parameter sowohl zum Übergeben von Eingabedaten an die gespeicherte Prozedur als auch zum Zurückgeben von Ausgabedaten aus der Sprachumgebung verwenden muss.

OUT
Gibt an, dass die Sprachumgebung den Parameter zum Zurückgeben von Ausgabedaten aus der gespeicherten Prozedur verwenden muss.

Non-database language environment call
Gibt eine Direktanforderung an Net.Data an, die eine Nicht-Datenbanksprachumgebung aufruft.

LANGENV
Gibt die Zielsprachumgebung für die Ausführung der Funktion an.

lang_env
Gibt den Namen der Nicht-Datenbanksprachumgebung an:

- DTW_PERL
- DTW_REXX
- DTW_SYSTEM

FUNC
Gibt an, dass die Direktanforderung die Ausführung eines Programms angibt.

program_name
Gibt das Programm mit der auszuführenden Funktion an.

parm_value
Gibt einen gültigen Parameterwert für die Funktion an.

Beispiele für Direktanforderungen

Die folgenden Beispiele zeigen die unterschiedlichen Möglichkeiten zum Aufrufen von Net.Data bei Verwendung der Direktanforderungsmethode.

HTML-Verbindungen (Links): In den folgenden Beispielen werden Direktanforderungen zum Aufrufen von Net.Data über Verbindungen (Links) verwendet.

Beispiel 1: Eine Verbindung (Link), die die Perl-Sprachumgebung sowie ein Perl-Script aufruft, das sich in der Pfadanweisung EXEC der Net.Data-Initialisierungsdatei befindet

```
<a href="http://server/cgi-bin/db2www/?LANGENV=DTW_PERL&FUNC=my_perl(hi)">
  any text</a>
```

Beispiel 2: Eine Verbindung (Link), die die Perl-Sprachumgebung wie im vorherigen Beispiel aufruft, jedoch eine Zeichenfolge mit URL-codierten Werten für die doppelten Anführungszeichen und die Leerzeichen übergibt

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

Beispiel 3: Eine URL-Adresse, die zur Ausführung einer SQL-Abfrage mit Hilfe der SQL-Sprachumgebung führt

```
<a href="http://server/cgi-bin/db2www/?LANGENV=DTW_SQL&DATABASE=CELDIAL
 &SQL=select+*+from+customer">any text</a>
```

Beispiel 4: Eine URL-Adresse, die die REXX-Sprachumgebung und ein REXX-Programm aufruft und Parameter an das Programm übergibt

```
<a href="http://server/cgi-bin/db2www/?LANGENV=DTW_REXX&FUNC=myexec.
     cmd(parm1,parm2)">any text</a>
```

Beispiel 5: Eine URL-Adresse, die eine gespeicherte Prozedur aufruft und Parameter an die SQL-Sprachumgebung übergibt

```
<a href="http://server/cgi-bin/db2www/?LANGENV=DTW_SQL&FUNC=MY_STORED_PROC
   (IN+CHAR(30)+Salaries)&DATABASE=CELDIAL">any text</a>
```

Hinweis: Sie müssen bestimmte Zeichen, wie Leerzeichen und doppelte Anführungszeichen, innerhalb von URL-Adressen codieren. In diesem Beispiel müssen die doppelten Anführungszeichen bzw. Leerzeichen innerhalb des Parameterwerts als %22 bzw. als Pluszeichen (+) codiert werden. Wenn dieser Link von einem Makro generiert worden ist, können Sie die integrierte Funktion DTW_URLESCSEQ verwenden, um den gesamten Text zu codieren, der in einer URL-Adresse codiert sein muss. Weitere Informationen zur Funktion DTW_URLESCSEQ finden Sie im Handbuch *Net.Data Reference*.

HTML-Formulare: In den folgenden Beispielen werden Direktanforderungen zum Aufrufen von Net.Data über Formulare verwendet.

Beispiel: Ein HTML-Formular, das zur Ausführung einer SQL-Abfrage mit Hilfe der SQL-Sprachumgebung führt, die Verbindung zur Datenbank CELDIAL herstellt und eine Tabelle abfragt

```
<form method="post"
 action="http://server/cgi-bin/db2www/">
<input type=hidden name="LANGENV" value="dtw_sql" />
<input type=hidden name="database" value="celdial" />
 <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>
```

Aufrufen von Net.Data über die Webserver-APIs

Net.Data unterstützt je nach Betriebssystem die in der folgenden Liste aufgeführten Web-APIs:

APAPI-Plug-In
: Apache API-Plug-In

IBM HTTP Server-API-Plug-In
: IBM HTTP Server-API-Plug-In

ISAPI-Plug-In
: API-Plug-In für Microsoft Internet Server

NSAPI-Plug-In
: API-Plug-In für Netscape Server

Im Anhang zu den Betriebssystemen im Handbuch *Net.Data Reference* finden Sie Informationen darüber, welche Webserver-APIs für Ihr Betriebssystem unterstützt werden. Informationen zum Konfigurieren von Net.Data und des Webservers zur Verwendung mit APIs finden Sie in „Konfigurieren von Net-.Data zur Verwendung mit den Webserver-APIs” auf Seite 47.

Anforderungen:

- Wenn Net.Data im Modus APAPI, ISAPI oder NSAPI ausgeführt wird, müssen Sie Ihren Webserver erneut starten, damit er Net.Data erneut laden und als Prozess ausführen kann.
- Wenn Sie Änderungen an der Initialisierungsdatei vorgenommen haben, nachdem der Webserver Net.Data im API-Modus aufgerufen hat, müssen Sie den Webserver erneut starten. An der Net.Data-Initialisierungsdatei (`db2www.ini`) vorgenommene Änderungen werden sonst nicht wirksam. Im API-Modus liest Net.Data die Initialisierungsdatei nur einmal, um den Systemaufwand zu reduzieren.
- Im API-Modus ist für die Oracle- und ODBC-Sprachumgebungen eine Direktverbindung erforderlich.

Gehen Sie wie folgt vor, um Webserver-APIs aufzurufen:

Für APAPI:

Syntax:

`http://`*`server`*`/.db2www/`*`macro_name`*`/`*`block`*`[?`*`name`*`=val&...`

Parameter:

server
Name des Servers

macro_name
Der relative Pfad Ihres Makros unter dem Verzeichnis, das mit MACRO_PATH angegeben wird

block
Name des zu verarbeitenden HTML- oder XML-Blocks im Makro

?*name***=***val***&...**
Gibt mindestens einen optionalen Parameter an, der an Net.Data weitergegeben wird.

Beispiel:

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

Für ISAPI:

Syntax:

`http://`*`server/server_HTML_root_directory/dll_name/macro_name/block`*`[?`*`name=val`*`&...]`

Parameter:

server_name
Name des Servers

server_HTML_root_directory
Name des HTML-Stammverzeichnisses des Webservers

dll_name
Name der DLL-Datei für ISAPI von Net.Data (`dtwisapi.dll`)

macro_name
Der relative Pfad Ihres Makros unter dem Verzeichnis, das mit MACRO_PATH angegeben wird

block
Name des zu verarbeitenden HTML- oder XML-Blocks im Makro

?*name***=***val***&...**
Gibt mindestens einen optionalen Parameter an, der an Net.Data weitergegeben wird.

Beispiel:

```
http://myserver/scripts/dtwisapi.dll/mymacro.dtw/report
```

Für NSAPI:

Syntax:

`http://`*`server/macro_name/block`*`[?`*`name=val`*`&...]`

Parameter:

server
Name des Servers

macro_name
Der relative Pfad Ihres Makros unter dem Verzeichnis, das mit MACRO_PATH angegeben wird. Die Erweiterung des Makros, zum Beispiel `.dtw`, muss in der Webserver-Konfigurationsdatei definiert werden. Weitere Informationen hierzu finden Sie in „Konfigurieren von Net.Data zur Verwendung mit den Webserver-APIs” auf Seite 47.

block
Name des zu verarbeitenden HTML- oder XML-Blocks im Makro

?*name***=***val***&...**
Gibt mindestens einen optionalen Parameter an, der an Net.Data weitergegeben wird.

Beispiel:

```
http://myserver/mymacro.dtw/report
```

Kapitel 5. Entwickeln von Net.Data-Makros

Ein Net.Data-Makro ist eine Textdatei, die aus einer Reihe von Net.Data-Makrosprachkonstrukten besteht, die folgenden Zwecken dienen:

- Angeben des Layouts von Webseiten
- Definieren von Variablen und Funktionen
- Aufrufen von Funktionen, die in Net.Data integriert bzw. im Makro definiert sind
- Formatieren der Verarbeitungsausgabe in HTML und Rückgabe an den Web-Browser zur Anzeige

Das Net.Data-Makro enthält zwei Abschnitte: den Deklarationsabschnitt und den Darstellungsabschnitt, wie in Abb. 19 gezeigt wird.

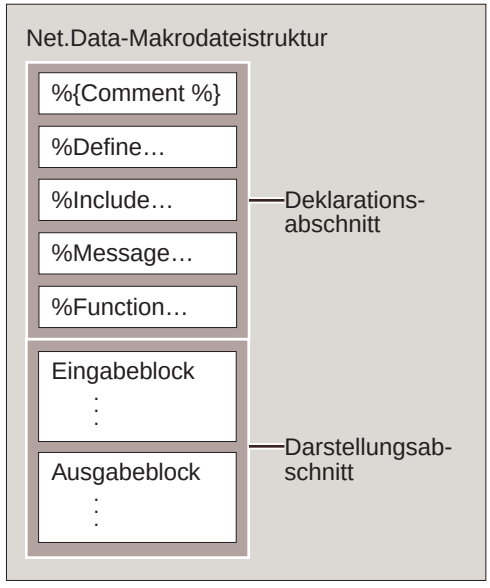


Abbildung 19. Makrostruktur

- Der *Deklarationsabschnitt* enthält die Definitionen von Variablen und Funktionen im Makro.
- Der *Darstellungsabschnitt* enthält HTML- oder XML-Blöcke, die das Layout der Webseite festlegen. Die HTML- oder XML-Blöcke bestehen aus Anweisungen zur Textdarstellung, die von Ihrem Web-Browser unterstützt werden, etwa HTML, JavaScript und gut strukturiertes XML.

Sie können diese Abschnitte auch mehrmals in beliebiger Reihenfolge verwenden. Im Handbuch *Net.Data Reference* finden Sie Informationen zur Syntax der Makroabschnitte und der Konstrukte.

Hinweis zu Berechtigungen: Stellen Sie sicher, dass die Benutzer-ID, unter der Net.Data ausgeführt wird, über eine Berechtigung zum Lesen dieser Datei verfügt. Weitere Informationen hierzu finden Sie in „Erteilen von Zugriffsrechten für Dateien, auf die Net.Data zugreift" auf Seite 65.

Dieses Kapitel behandelt die verschiedenen Blöcke, aus denen ein Net.Data-Makro besteht, und erläutert die Methoden, mit denen Sie das Makro schreiben können.

Aufbau eines Net.Data-Makros

Das Makro besteht aus zwei Abschnitten:

- Dem Deklarationsabschnitt, der die im Darstellungsabschnitt verwendeten Definitionen enthält. Im Deklarationsabschnitt werden zwei optionale Hauptblöcke verwendet:
  - DEFINE-Block
  - FUNCTION-Block

  Der Deklarationsabschnitt kann darüber hinaus noch weitere Sprachkonstrukte und Anweisungen enthalten, wie z. B. EXEC-Anweisungen, IF-Blöcke, INCLUDE-Anweisungen und MESSAGE-Blöcke. Weitere Informationen zu den Sprachkonstrukten finden Sie im Kapitel über die Sprachkonstrukte im Handbuch *Net.Data Reference*.

  Hinweis zu Berechtigungen: Stellen Sie sicher, dass die Benutzer-ID, unter der Net.Data ausgeführt wird, über eine Berechtigung zum Lesen und Ausführen von Dateien, auf die EXEC-Anweisungen verweisen, und zum Lesen von Dateien, auf die INCLUDE-Anweisungen verweisen, verfügt. Weitere Informationen hierzu finden Sie in „Erteilen von Zugriffsrechten für Dateien, auf die Net.Data zugreift" auf Seite 65.
- Der Darstellungsabschnitt definiert das Layout der Webseite, verweist auf Variablen und ruft Funktionen mit Hilfe von HTML- oder XML-Blöcken auf, die als Eingangs- und Endpunkte für das Makro verwendet werden. Wenn Sie Net.Data aufrufen, geben Sie den Namen eines Blocks als Ein-

gangspunkt zur Verarbeitung des Makros an. Die HTML- oder XML-Blöcke werden in „HTML-Blöcke” auf Seite 100 und „XML-Blöcke” auf Seite 102 beschrieben.

Im folgenden Abschnitt werden anhand eines einfachen Net.Data-Makros die Elemente der Makrosprache erläutert. Dieses Beispielmakro zeigt ein Formular, das Informationen anfordert, die an ein REXX-Programm übergeben werden. Das Makro übergibt diese Informationen an ein externes REXX-Programm namens `ompsamp.cmd`, das die vom Benutzer eingegebenen Daten zurückmeldet. Die Ergebnisse werden anschließend auf einer zweiten Webseite angezeigt.

Sehen Sie sich zunächst das gesamte Makro an. Im Folgenden werden dann die einzelnen Blöcke näher erläutert:

```
%{ **********************       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>@rexx1(input_data)
<p><hr>
<p>[<a href="/">Home page</a> |
<a href="input">Previous page</a>]
</body></html>
%}
```

Das Beispielmakro besteht aus vier Hauptblöcken: dem DEFINE-, dem FUNCTION- sowie den beiden HTML-Blöcken. Ein Net.Data-Makro kann auch mehrere DEFINE-, FUNCTION- und HTML-Blöcke enthalten.

Die beiden HTML-Blöcke enthalten Textdarstellungsanweisungen wie HTML, die das Schreiben von Webmakros sehr einfach machen. Wenn Sie mit HTML vertraut sind, besteht die Erstellung eines Makros nur aus dem Hinzufügen von Makroanweisungen, die dynamisch auf dem Server verarbeitet werden, sowie von SQL-Anweisungen, die an die Datenbank gesendet werden.

Obwohl das Makro einem HTML-Dokument ähnelt, greift der Webserver über Net.Data mit Hilfe von CGI, einer Webserver-API oder einem Java-Servlet darauf zu. Net.Data benötigt zum Aufrufen eines Makros zwei Parameter: den Namen des zu verarbeitenden Makros und den anzuzeigenden HTML-Block in diesem Makro.

Wenn das Makro aufgerufen wird, beginnt Net.Data mit der Verarbeitung am Anfang der Datei. In den folgenden Abschnitten wird die Verarbeitung der einzelnen Blöcke durch Net.Data beschrieben.

Der DEFINE-Block

Der DEFINE-Block enthält das DEFINE-Sprachkonstrukt und Variablendefinitionen, die später in den HTML-Blöcken verwendet werden. Das folgende Beispiel zeigt einen DEFINE-Block mit einer Variablendefinition:

```
%{ **********************       DEFINE Block        ************************%}
%DEFINE {
   page_title="Net.Data Macro Template"
%}
```

Die erste Zeile ist ein Kommentar. Ein Kommentar ist jeder Text, der in `%{` und `%}` eingeschlossen ist. Kommentare können an jeder beliebigen Stelle im Makro eingefügt werden. Die nächste Anweisung beginnt einen DEFINE-Block. Sie können mehrere Variablen in einem DEFINE-Block definieren. Im vorliegenden Beispiel wird lediglich eine Variable (`page_title`) definiert. Nach ihrer Definition kann auf diese Variable an jeder Stelle innerhalb des Makros

mit der Syntax $(page_title) verwiesen werden. Mit Hilfe der Variablen können Sie zu einem späteren Zeitpunkt auf einfache Art globale Änderungen an Ihrem Makro vornehmen. Die letzte Zeile dieses Blocks, %}, kennzeichnet das Ende des DEFINE-Blocks.

Der FUNCTION-Block

Der FUNCTION-Block enthält die Deklarationen für Funktionen, die von den HTML-Blöcken aufgerufen werden. Funktionen werden von Sprachumgebungen verarbeitet und können Programme, SQL-Abfragen oder gespeicherte Prozeduren ausführen.

Das folgende Beispiel zeigt zwei FUNCTION-Blöcke. Der erste Block definiert den Aufruf eines externen REXX-Programms, und der zweite Block enthält Inline-REXX-Anweisungen.

```
%{ ********************** FUNCTION Block ***********************************%}
%FUNCTION(DTW_REXX) rexx1 (IN input) returns(result) { <-- Diese Funktion
                                                          akzeptiert einen Parameter
                                                          und gibt die Variable 'result'
                                                          zurück, die vom externen
                                                          REXX-Programm zugeordnet
                                                          wird.
     %EXEC{ompsamp.cmd %}  <-- Die Funktion führt ein externes REXX-Programm
                               namens "ompsamp.cmd" aus.
%}

%FUNCTION(DTW_REXX) today () RETURNS(result) {
      result = date()  <-- Die einzelne Quellenanweisung für
                           diese Funktion befindet sich inline.
%}
```

Der erste FUNCTION-Block, rexx1, ist die Deklaration einer REXX-Funktion, die ihrerseits ein externes REXX-Programm namens ompsamp.cmd ausführt. Von dieser Funktion wird eine Eingabevariable input entgegengenommen und automatisch an den externen REXX-Befehl übergeben. Der REXX-Befehl gibt außerdem eine Variable namens result zurück. Der Inhalt der Variablen result im REXX-Befehl ersetzt den im OUTPUT-Block enthaltenen Funktionsaufruf @rexx1(). Auf die Variablen input und result kann das REXX-Programm, wie im Quellencode für ompsamp.cmd gezeigt wird, direkt zugreifen:

```
/* REXX */
result = 'The REXX program received "'input'" from the macro.'
```

Der Code in dieser Funktion gibt die an sie übergebenen Daten zurück. Sie können den Ergebnistext nach Belieben formatieren, indem Sie den anfordernden Funktionsaufruf @rexx1() zwischen normale HTML-Befehle für Hervorhebungen (wie z. B. <b> oder) setzen. Anstatt die Variable result zu verwenden, hätte das REXX-Programm auch HTML-Befehle mit Hilfe der REXX-Anweisung SAY in die Standardausgabe schreiben können.

Der zweite FUNCTION-Block, today, verweist ebenfalls auf ein REXX-Programm. In diesem Fall befindet sich jedoch das gesamte REXX-Programm selbst innerhalb der Funktionsdeklaration. Ein externes Programm ist nicht erforderlich. Inline-Programme sind für REXX- und Perl-Funktionen zulässig, da REXX und Perl Interpreter-Sprachen sind, die syntaktisch analysiert und dynamisch ausgeführt werden können. Inline-Programme bieten den Vorteil der Einfachheit, da sie keine separat zu verwaltende Programmdatei erfordern. Die erste REXX-Funktion hätte ebenfalls inline angelegt werden können.

HTML-Blöcke

HTML-Blöcke definieren das Layout einer Webseite, verweisen auf Variablen und rufen Funktionen auf. HTML-Blöcke werden als Eingangs- und Endpunkte für das Makro verwendet. In der Net.Data-Aufrufanforderung wird immer ein HTML-Block angegeben, und jedes Makro muss mindestens einen HTML-Block enthalten.

Der erste HTML-Block im Beispielmakro heißt INPUT. HTML(INPUT) enthält HTML-Code für ein einfaches Formular mit einem Eingabefeld.

```
%{ **********************      HTML Block: Input    ************************%}
%HTML (INPUT) {               <--- Gibt den Namen dieses HTML-Blocks an.
<html>
<head>
<title>$(page_title)</title>  <--- Beachten Sie die Variablensubstitution.
</head><body>
<h1>Input Form</h1>
Today is @today()             <--- Diese Zeile enthält den Aufruf einer Funktion.

<form method="post" action="output">  <--- Bei Übergabe dieses Formulars wird
                                           der HTML-Block "OUTPUT" aufgerufen.
Type some data to pass to a REXX program:
<input name="input_data"      <--- "input_data" wird bei der Übergabe des Formulars
TYPE="text" SIZE="30" />           definiert. Auf diese Variable kann von anderer
                                   Stelle des Makros verwiesen werden. Sie wird
                                   mit der Benutzereingabe initialisiert.
</p>
<input type="submit" value="enter" />
<hr>
<p>
[
<a href="/">Home page</a>]</p>
 </body><html>
%}                            <--- Beendet den HTML-Block.
```

Der gesamte HTML-Block wird von der HTML-Blockkennung %HTML (INPUT) {...%} eingeschlossen. INPUT gibt den Namen dieses Blocks an. Der Name kann Unterstreichungszeichen, Punkte und beliebige alphanumerische Zeichen enthalten. Net.Data unterscheidet nicht zwischen Groß- und Kleinschreibung.

Der HTML-Befehl `<title>` enthält ein Beispiel für eine Variablensubstitution. Der Wert der Variablen `page_title` wird in den Titel des Formulars eingesetzt.

Dieser Block enthält außerdem einen Funktionsaufruf. Der Ausdruck `@today()` ist ein Aufruf an die Funktion `today`. Diese Funktion wird im Block FUNCTION definiert, der oben beschrieben ist. Net.Data fügt das Ergebnis der Funktion `today`, d. h. das aktuelle Datum, in den HTML-Text an der Stelle ein, an der sich der Ausdruck `@today()` befindet.

Der Parameter ACTION der Anweisung FORM zeigt ein Beispiel für die Navigation zwischen HTML-Blöcken bzw. zwischen Makros. Durch den Verweis auf den Namen eines anderen Blocks in einem Parameter ACTION wird bei der Übergabe des Formulars auf diesen Block zugegriffen. Alle Eingabedaten eines HTML-Formulars werden als implizite Variablen an den neuen Block übergeben. Dies gilt für das einzelne Eingabefeld, das in diesem Formular definiert ist. Bei der Übergabe des Formulars werden die in diesem Formular eingegebenen Daten in der Variablen *input_data* an den HTML-Block `HTML(OUTPUT)` übergeben.

Sie können über einen relativen Verweis auf HTML-Blöcke in anderen Makros zugreifen, wenn sich diese Makros auf demselben Webserver befinden. So greift z. B. der ACTION-Parameter `ACTION="../othermacro.dtw/main"` auf den HTML-Block `main` im Makro `othermacro.dtw` zu. Auch hier werden alle in das Formular eingegebenen Daten in der Variablen *input_data* an dieses Makro übergeben.

Beim Aufrufen von Net.Data wird die Variable als Teil der URL-Adresse weitergegeben. Beispiel:

```
<a href="/cgi-bin/db2www/othermacro.dtw/main?input_data=value">Next macro</a>
```

Sie können auf Formulardaten im Makro zugreifen und sie bearbeiten, indem Sie auf den im Formular angegebenen Variablennamen verweisen.

Der nächste HTML-Block des Beispiels ist der Block `HTML(OUTPUT)`. Er enthält HTML-Befehle und Net.Data-Makroanweisungen, die die Verarbeitung der Ausgabe von der Anforderung `HTML(INPUT)` definieren.

```
%{ **********************    HTML Block: Output     ************************%}
%HTML (OUTPUT) {
<html>
<head>
<title>$(page_title)</title>  <--- Weitere Substitution

</head><body>
<h1>Output Page</h1>
<p>@rexx1(input_data)  <--- Diese Zeile enthält einen Aufruf der Funktion rexx1,
                           die das Argument "input_data" übergibt.
<p>
<hr>
<p>
[
<a href="/">Home page</a> |
<a href="input">Previous page</a>]
%}
```

Wie der Block `HTML(INPUT)` besteht auch dieser Block aus Standard-HTML mit Net.Data-Makroanweisungen für die Substitution von Variablen und einem Funktionsaufruf. Auch hier wird in der Titelanweisung der Wert der Variablen `page_title` eingesetzt. Außerdem enthält dieser Block ebenfalls einen Funktionsaufruf. In diesem Fall wird die Funktion `rexx1` aufgerufen, und der Inhalt der Variablen `input_data`, der von dem im INPUT-Block definierten Formular empfangen wurde, an die Funktion übergeben. Sie können Variablen in beliebiger Zahl an eine Funktion bzw. von einer Funktion übergeben. Die Funktionsdefinition gibt die Anzahl und die Verwendung der übergebenen Variablen an.

XML-Blöcke

Unabhängig davon, ob Sie XML an eine andere Verarbeitungsanwendung oder an einen Client-Browser übergeben wollen, können Sie den XML-Inhalt mit Hilfe der XML-Blockstruktur übergeben.

Der XML-Block funktioniert wie der HTML-Block; es handelt sich um einen Eingangspunkt für das Makro. Innerhalb des Blocks können Sie XML-Befehle direkt eingeben, auf Variablen verweisen und Funktionsaufrufe durchführen.

Damit Sie das generierte XML-Dokument Ihren Erfordernissen entsprechend anpassen können, generiert der XML-Block die Prologbefehle nicht.

Geben Sie die Prologinformationen Ihres Unternehmens ein und fügen Sie eine gewünschte Formatvorlage ein. Net.Data verfügt über drei XSL-Formatvorlagen, die Sie verwenden können. Diese Formatvorlagen enthalten Umsetzungen für alle XML-Elemente, die Net.Data generiert. Die Formatvorlagen sind jedoch nur Beispiele, die Sie erweitern oder als Schablone für eigene Formatvorlagen verwenden können.

```
%DEFINE SHOWSQL = "yes"

%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="/netdata-xml/ndTable.xsl" ?>

<XMLBlock>
  <h1>List of New Managers</h1>
  @NewManager()
</XMLBlock>
%}
```

Abbildung 20. Ein Makro mit einem XML-Berichtsblock

Beim Aufrufen einer SQL-Funktion, die einen Standardbericht ausgibt, generiert Net.Data die Ergebnismenge mit Hilfe einer kleinen Gruppe von XML-Elementen. Siehe hierzu die folgende Beispieldokumentartbeschreibung (Document Type Description, DTD).

```
<!-------------------------------------------------------->
<!-- The root element of the document.                -->
<!-------------------------------------------------------->
<!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)>

<!-------------------------------------------------------->
<!-- Messages are presented with the Message element.   -->
<!-------------------------------------------------------->
<!ELEMENT Message  (#PCDATA)>
```

Die Elemente werden wie folgt definiert:

XMLBlock
Das Stammelement für das Dokument. Dieser Befehl muss manuell eingegeben werden.

RowSet
Enthält die Zeilen in einer Ergebnismenge. Das Attribut name von RowSet wird wie folgt bestimmt:

- Bei einer Ergebnismenge, die aus einem Aufruf einer Funktion resultiert, die eine SQL-Abfrage ausführt, wird der Name der Funktion verwendet.
- Bei einer Ergebnismenge, die aus einem Aufruf einer gespeicherten Prozedur resultiert, wird der Name der Ergebnismenge verwendet. Hat die Ergebnismenge keinen Namen, wird der Funktionsname verwendet.

Row Enthält die Spalten einer Zeile und wird für Identifikationszwecke nummeriert.

Column
Enthält den Datenwert der betreffenden Zeile und die Spalte, nach der sie benannt ist.

ShowSQL
Enthält die SQL-Anweisung für die aktuelle Abfrage.

Message
Enthält alle durch Net.Data oder DB2 generierten Fehlernachrichten.

Bei Verwendung der oben aufgeführten Elemente würde Net.Data die folgende Ausgabe aus dem in Abb. 20 auf Seite 103 aufgelisteten Makro generieren.

```
<?xml version="1.0" ?>
<?xml-stylesheet type="text/xsl" href="/netdata-xml/ndTable.xsl" ?>
<XMLBlock>
  <h1>List of New Managers</h1>
  <ShowSQL>select * from staff where job = 'Mgr' and years <= 5</ShowSQL>
  <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>
```

Abb. 21 und Abb. 22 auf Seite 106 zeigen, wie die Daten oben bei Verwendung der beiden Net.Data-Formatvorlagen (ndTable.xsl und ndRecord.xsl) in einem Browser aussehen würden.

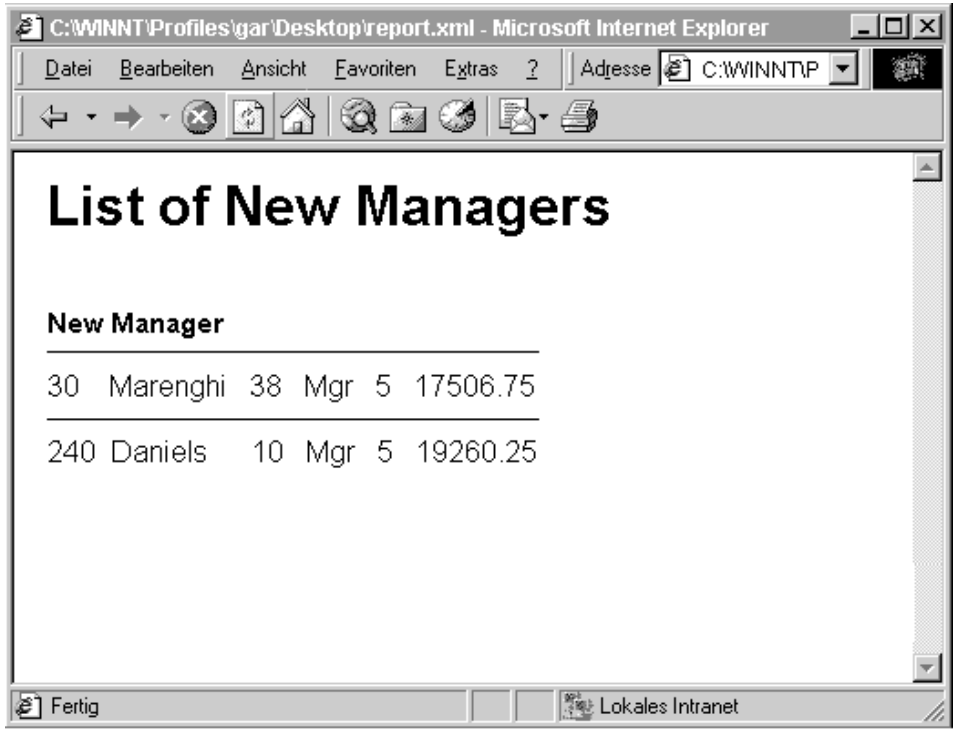


Abbildung 21. XML-Anzeige mit Formatvorlage ndTable.xsl

Abbildung 22. XML-Anzeige mit Formatvorlage ndRecord.xsl

Net.Data-Makrovariablen

Mit Net.Data können Sie Variablen in einem Net.Data-Makro definieren und auf diese Variablen verweisen. Darüber hinaus können Sie diese Variablen vom Makro aus an die Sprachumgebungen übergeben und von dort empfangen. Die übergebenen Variablennamen, Werte und Literalzeichenfolgen werden als Token bezeichnet. Net.Data setzt für die Größe der Token keine Grenze und übergibt alle Token, die Ihr Systemspeicher bearbeiten kann. Bei bestimmten Sprachumgebungen können jedoch Einschränkungen bezüglich der Token-Größe bestehen.

Bei der Definition von Net.Data-Variablen kann die Art der Variable und bei Bedarf ein vordefinierter Wert festgelegt werden.

Diese Variablen können je nach Definition in die folgenden Arten untergliedert werden:

- Explizit mit Hilfe der Anweisung DEFINE im DEFINE-Block definierte Variablen
- Vordefinierte Variablen, die von Net.Data zur Verfügung gestellt werden und auf einen bestimmten Wert gesetzt sind. Dieser Wert kann in der Regel nicht geändert werden.
- Implizit definierte Variablen, von denen es vier Arten gibt:
  - Variablen, die zwar nicht explizit definiert sind, jedoch verfügbar gemacht werden, sobald ihnen zum ersten Mal ein Wert zugeordnet wird
  - Parametervariablen, die zu einer Definition im FUNCTION-Block gehören und auf die nur innerhalb eines FUNCTION-Blocks verwiesen werden kann
  - Variablen, die von Net.Data verfügbar gemacht werden und Formulardaten bzw. Abfragezeichenfolgedaten entsprechen
  - Variablen, die einer Net.Data-Tabelle zugeordnet sind und auf die nur innerhalb eines ROW-Blocks oder eines REPORT-Blocks verwiesen werden kann

In den folgenden Abschnitten wird Folgendes beschrieben:

- „Geltungsbereich von Kennungen"
- „Definieren von Variablen" auf Seite 109
- „Verweisen auf Variablen" auf Seite 112
- „Variablenarten" auf Seite 113

Geltungsbereich von Kennungen

Wenn eine Kennung einen globalen Geltungsbereich hat, kann auf sie während einer einzelnen Anforderung an einer beliebigen Stelle in einem Makro verwiesen werden. Der Bereich, in dem eine Kennung sichtbar ist, wird als *Geltungsbereich* der Kennung bezeichnet. Es gibt die folgenden fünf Geltungsbereiche:

- Global

  Eine Kennung hat einen globalen Geltungsbereich, wenn auf sie von jeder Stelle des Makros aus verwiesen werden kann. Kennungen mit globalem Geltungsbereich sind:
  - Integrierte Net.Data-Funktionen
  - Formulardaten
  - Abfragezeichenfolgedaten
  - Variablen, die innerhalb eines HTML-Blocks verfügbar gemacht werden
- Makro

  Eine Kennung hat den Geltungsbereich einer Makrodatei, wenn ihre Deklaration außerhalb aller Blöcke erfolgt. Ein Block beginnt mit einer öffnenden geschweiften Klammer ({) und endet mit einem Prozentzeichen und einer

schließenden geschweiften Klammer (%}). (DEFINE-Blöcke sind von dieser Definition ausgenommen.) Im Gegensatz zu einer Kennung mit einem globalen Geltungsbereich kann auf eine Kennung mit einem Makrogeltungsbereich nur durch Elemente im Makro verwiesen werden, die auf die Deklaration der Kennung folgen.

- FUNCTION-Block bzw. MACRO_FUNCTION-Block

  Eine Kennung hat den Geltungsbereich FUNCTION-Block, wenn Folgendes zutrifft:
  - Die Kennung wird in der Parameterliste der Funktionsdefinition deklariert.

    Wenn eine Kennung mit dem gleichen Namen bereits außerhalb der Funktionsdefinition vorhanden ist, verwendet Net.Data die Kennung aus der Funktionsparameterliste innerhalb des Funktionsblocks.
  - Die Kennung wird im Funktionsblock angelegt und vor dem Funktionsaufruf weder deklariert noch initialisiert.

  Eine Kennung hat nicht den Geltungsbereich FUNCTION-Block, wenn sie außerhalb der Funktion deklariert oder initialisiert wurde und nicht in der Funktionsparameterliste deklariert wird. Der Wert der Kennung innerhalb des Funktionsblocks bleibt unverändert, außer wenn er durch die Funktion aktualisiert wird.

- REPORT-Block

  Eine Kennung hat den Geltungsbereich REPORT-Block, wenn auf sie nur innerhalb eines REPORT-Blocks verwiesen werden kann (zum Beispiel Tabellenspaltennamen N1, N2, ..., N*n*). Nur solche Variablen, die Net.Data implizit als Teil der Tabellenverarbeitung definiert, können den Geltungsbereich REPORT-Block haben. Alle anderen verfügbar gemachten Variablen haben den Geltungsbereich FUNCTION-Block.

- ROW-Block

  Eine Kennung hat den Geltungsbereich ROW-Block, wenn auf sie nur innerhalb eines ROW-Blocks verwiesen werden kann (zum Beispiel Tabellenwertnamen V1, V2, ..., V*n*). Nur solche Variablen, die Net.Data implizit als Teil der Tabellenverarbeitung definiert, können den Geltungsbereich ROW-Block haben. Alle anderen verfügbar gemachten Variablen haben den Geltungsbereich FUNCTION-Block.

Definieren von Variablen

Variablen können in einem Net.Data-Makro auf drei Arten definiert werden:

- DEFINE-Anweisung bzw. -Block
- HTML-Formularbefehle
- Abfragezeichenfolgedaten

Ein Variablenwert, der von einem Formular oder von Abfragezeichenfolgedaten empfangen wird, überschreibt einen mit der Anweisung DEFINE in einem Net.Data-Makro definierten Variablenwert.

- **DEFINE-Anweisung bzw. -Block**

  Die einfachste Art, eine Variable zur Verwendung in einem Net.Data-Makro zu definieren, ist der Einsatz der Anweisung DEFINE. Die Syntax sieht wie folgt aus:

```
%DEFINE variable_name="variable value"

%DEFINE variable_name={ variable value on multiple
                        lines of text  %}

%DEFINE {
    variable_name1="variable value 1"
    variable_name2="variable value 2"
%}
```

  variable_name ist der Name, den Sie der Variablen geben. Variablennamen müssen mit einem Buchstaben oder Unterstreichungszeichen beginnen und können jedes beliebige alphanumerische Zeichen, ein Unterstreichungszeichen, einen Punkt oder ein Hash-Zeichen (#) enthalten. Mit Ausnahme der Tabellenvariablen V_*columnName* muss bei allen Variablennamen die Groß-/Kleinschreibung beachtet werden.

  Beispiel:

```
%DEFINE reply="hello"
```

  Die Variable `reply` hat den Wert `hello`.

Zwei aufeinanderfolgende Anführungszeichen allein entsprechen einer leeren Zeichenfolge. Beispiel:

```
%DEFINE empty=""
```

Die Variable `empty` enthält eine leere Zeichenfolge.

Wenn Ihre Variable Sonderzeichen wie Zeilenendezeichen enthält, verwenden Sie geschweifte Blockklammern um den Wert:

```
%DEFINE introduction={
Hello,
My name is John.
%}
```

Sollen Anführungszeichen in einer Zeichenfolge verwendet werden, setzen Sie jeweils zwei Anführungszeichen hintereinander.

```
%DEFINE HI="say ""hello"""
```

Sie können auch geschweifte Blockklammern verwenden, um die Anführungszeichen zu umgehen:

```
%DEFINE HI={ say "hello" %}
```

Verwenden Sie einen DEFINE-Block, wenn Sie mehrere Variablen mit einer Anweisung DEFINE definieren wollen:

```
%DEFINE {
         variable1="value1"
         variable2="value2"
         variable3="value3"
         variable4="value4"
%}
```

- **HTML-Formularbefehle: SELECT, INPUT und TEXTAREA**

  Mit den HTML-Formularbefehlen SELECT, INPUT und TEXTAREA können Sie Variablen Werte zuordnen. Im folgenden Beispiel werden Standardbefehle für HTML-Formulare zum Definieren von Net.Data-Variablen verwendet:

  ```
  <input name="variable_name" TYPE=... />
  ```

  oder

  ```
  <select name="variable_name">
    <option>value one
    <option>value two
  </select>
  ```

  Mit dem Befehl TEXTAREA können Sie eine Variable zuordnen, die sich auf mehrere Zeilen erstreckt oder Sonderzeichen wie Anführungszeichen enthält:

  ```
  <textarea name="variable_name" ROWS="4">
  Please type the multi-line value
  of your variable here.
  </textarea>
  ```

  variable_name ist der Name, den Sie der Variablen geben. Der Wert der Variablen wird durch die im Formular empfangene Eingabe bestimmt. In „HTML-Formulare" auf Seite 84 finden Sie ein Beispiel dafür, wie diese Art der Variablendefinition in einem Net.Data-Makro verwendet wird.

- **Abfragezeichenfolgedaten**

  Sie können über die Abfragezeichenfolge Variablen an Net.Data übergeben. Beispiel:

  ```
  http://www.ibm.com/cgi-bin/db2www/stdqry1.dtw/input?field=custno
  ```

  In diesem Beispiel geben der Variablenname `field` und der Variablenwert `custno` zusätzliche Daten an, die Net.Data über die Abfragezeichenfolge empfängt. Net.Data empfängt und verarbeitet die Daten so wie Formulardaten.

Verweisen auf Variablen

Sie können auf zuvor definierte Variablen verweisen, um an der Stelle des Verweises den Wert der Variablen zu erhalten. Ein Verweis auf eine Variable in Net.Data-Makros besteht aus dem Variablennamen, der in $(und) eingeschlossen ist. Beispiel:

```
$(variable_name)
$(homeURL)
```

Wenn Net.Data auf einen Variablenverweis trifft, ersetzt Net.Data den Variablenverweis durch den Wert der Variablen. Variablenverweise können Zeichenfolgen, andere Variablenverweise und Funktionsaufrufe enthalten.

Sie können Variablennamen dynamisch generieren. Bei diesem Verfahren können Sie mit Schleifen Tabellen unterschiedlicher Größe oder Eingabedaten für während der Laufzeit erstellte Listen verarbeiten, wenn die Anzahl in der Liste nicht im voraus ermittelt werden kann. Sie können z. B. Listen von HTML-Formularelementen generieren, die basierend auf von einer SQL-Abfrage zurückgegebenen Sätzen generiert werden.

Wenn Sie Variablen als Teil Ihrer Textdarstellungsanweisungen verwenden wollen, verweisen Sie in den HTML-Blöcken Ihres Makros darauf.

Ungültige Variablenverweise: Ungültige Variablenverweise werden als leere Zeichenfolge aufgelöst. Wenn z. B. ein Variablenverweis ungültige Zeichen wie ein Ausrufezeichen (!) enthält, wird der Verweis als leere Zeichenfolge aufgelöst.

Gültige Variablennamen müssen mit einem alphanumerischen Zeichen oder einem Unterstreichungszeichen anfangen und können aus alphanumerischen Zeichen einschließlich Punkt, Unterstreichungszeichen und Hash-Zeichen bestehen.

Beispiel 1: Variablenverweis in einer Verbindung (Link)

Sie definieren beispielsweise die Variable *homeURL* wie folgt:

```
%DEFINE homeURL="http://www.ibm.com/"
```

Dann können Sie Verweise auf die Homepage in der Form $(*homeURL*) verwenden und eine Verbindung (Link) erstellen:

```
<a href="$(homeURL)">Home page</a>
```

Sie können in vielen Teilen des Net.Data-Makros auf Variablen verweisen. Überprüfen Sie die Sprachkonstrukte in diesem Kapitel, um zu ermitteln, in welchen Teilen des Makros Variablenverweise zulässig sind. Wenn die Variable zu dem Zeitpunkt, zu dem auf sie verwiesen wird, noch nicht definiert

ist, gibt Net.Data eine leere Zeichenfolge zurück. Ein Variablenverweis allein definiert die Variable nicht. **Beispiel 2:** Dynamisch generierte Variablenverweise

Angenommen, Sie führen eine SQL-Anweisung SELECT mit einer gewissen Anzahl von Elementen aus. Sie können ein HTML-Formular mit Eingabefeldern mit Hilfe der folgenden ROW-Blöcke erstellen:

```
...
%ROW {
<input type=text name=@dtw_rconcat("I", ROW_NUM) size=10 maxlength=10 />
%}
...
```

Da Sie Eingabefelder erstellt haben, wollen Sie höchstwahrscheinlich auf die vom Benutzer eingegebenen Werte zugreifen, wenn das Formular zur Verarbeitung an Ihr Makro übergeben wird. Sie können eine Schleife codieren, um die Werte in einer Liste variabler Länge abzurufen:

```
<pre>
...
@dtw_assign(rowIndex, "1")
%while (rowIndex <= rowCount) {
The value entered for row $(rowIndex) is: $(I$(rowIndex))
@dtw_add(rowIndex, "1", rowIndex) %}
...
</pre>
```

Net.Data generiert zuerst mit dem Verweis I$(rowIndex) den Variablennamen. Der erste Variablenname könnte z. B. I1 sein. Net.Data verwendet anschließend diesen Wert und löst damit den Variablenwert auf.

Beispiel 3: Ein Variablenverweis mit verschachtelten Variablenverweisen und einem Funktionsaufruf

```
%define my = "my"
%define u = "lower"
%define myLOWERvar = "hey"


$($(my)@dtw_ruppercase(u)var)
```

Der Variablenverweis gibt den Wert von `hey` zurück.

Variablenarten

Sie können die folgenden Arten von Variablen in Ihren Makros verwenden:

- „Bedingungsvariablen" auf Seite 114
- „Umgebungsvariablen" auf Seite 114
- „Ausführbare Variablen" auf Seite 115
- „Verdeckte Variablen" auf Seite 116

- „Listenvariablen” auf Seite 117
- „Tabellenvariablen” auf Seite 119
- „Zusätzliche Variablen” auf Seite 120
- „Variablen zur Tabellenverarbeitung” auf Seite 121
- „Berichtsvariablen” auf Seite 122
- „Sprachumgebungsvariablen” auf Seite 123

Wenn Sie Variablen, die von Net.Data in spezieller Weise definiert werden (z. B. ENVVAR, LIST oder Bedingungslistenvariablen), Zeichenfolgen zuordnen, funktioniert die Variable nicht mehr in der definierten Weise. Das heißt, die Variable wird zu einer regulären Variablen, die eine Zeichenfolge enthält.

Informationen zur Syntax und Beispiele jeder Variablenart finden Sie im Handbuch *Net.Data Reference*.

Bedingungsvariablen

Bedingungsvariablen ermöglichen Ihnen, einen bedingten Wert für eine Variable mit Hilfe einer Methode zu definieren, die einem IF-THEN-Konstrukt ähnlich ist. Beim Definieren einer Bedingungsvariablen können Sie zwei mögliche Variablenwerte angeben. Wenn die erste Variable, auf die Sie verweisen, existiert, erhält die Bedingungsvariable den ersten Wert. Ansonsten erhält die Bedingungsvariable den zweiten Wert. Die Syntax für eine Bedingungsvariable sieht wie folgt aus:

```
varA = varB ? "value_1" : "value_2"
```

Wenn `varB` definiert ist, gilt `varA="`*`value_1`*`"`, andernfalls gilt `varA="`*`value_2`*`"`. Dies entspricht der Verwendung eines IF-Blocks wie im folgenden Beispiel:

```
%IF (varB)
    varA = "value_1"
%ELSE
    varA = "value_2"
%ENDIF
```

In „Listenvariablen” auf Seite 117 finden Sie ein Beispiel für die Verwendung von Bedingungsvariablen mit Listenvariablen.

Umgebungsvariablen

Sie können auf Umgebungsvariablen verweisen, die der Webserver dem Prozess bzw. Thread, der Ihre Net.Data-Anforderung verarbeitet, zur Verfügung stellt. Wenn auf die Variable ENVVAR verwiesen wird, gibt Net.Data den aktuellen Wert der Umgebungsvariablen mit dem gleichen Namen zurück.

Die Syntax zur Definition von Umgebungsvariablen sieht folgendermaßen aus:

```
%DEFINE var=%ENVVAR
```

Dabei ist *var* der Name der Umgebungsvariablen, die definiert wird.

Zum Beispiel kann die Variable SERVER_NAME als Umgebungsvariable definiert werden:

```
%DEFINE SERVER_NAME=%ENVVAR
```

Ein Verweis auf sie könnte wie folgt aussehen:

```
The server is $(SERVER_NAME)
```

Die Ausgabe sähe wie folgt aus:

```
The server is www.ibm.com
```

Weitere Informationen zur Anweisung ENVVAR finden Sie im Handbuch *Net.Data Reference*.

Ausführbare Variablen

Sie können über einen Variablenverweis mit Hilfe ausführbarer Variablen andere Programme aufrufen.

Ausführbare Variablen werden in einem Net.Data-Makro mit Hilfe des Sprachkonstrukts EXEC im DEFINE-Block definiert. Weitere Informationen zum Sprachelement EXEC finden Sie im Kapitel über Sprachkonstrukte im Handbuch *Net.Data Reference*. Im folgenden Beispiel wird die Variable `runit` zur Ausführung des ausführbaren Programms `testProg` definiert:

```
%DEFINE runit=%EXEC "testProg"
```

Die Variable `runit` wird zu einer ausführbaren Variablen.

Net.Data führt das ausführbare Programm aus, wenn ein gültiger Variablenverweis in einem Net.Data-Makro erkannt wird. Zum Beispiel wird das Programm `testProg` ausgeführt, wenn in einem Net.Data-Makro ein gültiger Variablenverweis auf die Variable `runit` enthalten ist.

Eine einfache Methode besteht darin, auf eine ausführbare Variable aus einer anderen Variablendefinition heraus zu verweisen. Das folgende Beispiel illustriert diese Methode. Die Variable `date` wird als ausführbare Variable definiert. Anschließend wird `dateRpt` als Variablenverweis definiert, der die ausführbare Variable enthält.

```
%DEFINE date=%EXEC "date"
```

Für jedes Vorkommen des Variablenverweises `$(date)` im Net.Data-Makro sucht Net.Data nach dem ausführbaren Programm `date`. Wenn das Programm gefunden wird, zeigt Net.Data folgenden Wert an:

```
Today is Tue 11-07-1999
```

Wenn Net.Data eine ausführbare Variable in einem Makro feststellt, sucht Net.Data das angegebene ausführbare Programm nach folgender Methode:

1. Es durchsucht die Verzeichnisse, die in der Net.Data-Initialisierungsdatei durch EXEC_PATH definiert sind. Nähere Informationen finden Sie in „EXEC_PATH" auf Seite 25.
2. Wenn Net.Data das Programm nicht findet, durchsucht das System die Verzeichnisse, die in der Umgebungsvariablen PATH bzw. in der Bibliothekenliste definiert sind. Wird das ausführbare Programm gefunden, führt Net.Data das Programm aus.

Einschränkung: Setzen Sie eine ausführbare Variable nicht auf den Wert der Ausgabe des aufgerufenen ausführbaren Programms. Im vorangegangenen Beispiel ist der Wert der Variablen `date` gleich NULL. Wenn Sie diese Variable in einem Funktionsaufruf DTW_ASSIGN verwenden, um ihren Wert einer anderen Variablen zuzuordnen, ist der Wert der neuen Variablen nach der Zuordnung ebenfalls NULL. Der einzige Zweck einer ausführbaren Variablen besteht darin, das Programm aufzurufen, das sie definiert.

Außerdem können Sie Parameter an das auszuführende Programm übergeben, indem Sie diese bei der Variablendefinition mit dem Programmnamen angeben. Im folgenden Beispiel werden die Werte für `distance` und `time` an das Programm *calcMPH* übergeben.

```
%DEFINE mph=%EXEC "calcMPH $(distance) $(time)"
```

Verdeckte Variablen

Sie können verdeckte Variablen verwenden, um den tatsächlichen Namen einer Variablen für Anwendungsbenutzer unsichtbar zu machen, die die Quelle Ihrer Webseite mit ihrem Web-Browser anzeigen. Eine verdeckte Variable wird wie folgt definiert:

1. Definieren Sie eine Variable für jede Zeichenfolge, die Sie verdecken wollen, nach dem letzten Verweis auf die jeweilige Variable im HTML-Block. Variablen werden immer mit Hilfe des Sprachkonstrukts DEFINE definiert, nachdem sie im HTML-Block verwendet wurden, wie in folgendem Beispiel. Auf die Variablen $$(*variable*) wird zuerst verwiesen, anschließend werden sie definiert.

2. Verwenden Sie in dem HTML-Block, in dem auf die Variablen verwiesen wird, für einen Variablenverweis zwei Dollarzeichen anstelle eines einzelnen Dollarzeichens. Zum Beispiel $$(X) anstelle von $(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
%}

...
```

   Wenn ein Web-Browser das HTML-Formular anzeigt, werden $$(name) und $$(addr) durch $(name) bzw. $(addr) ersetzt, so dass die tatsächlichen Namen für Tabelle und Spalten im HTML-Formular nirgendwo vorkommen. Anwendungsbenutzer können nicht erkennen, dass die tatsächlichen Variablennamen verdeckt sind. Wenn der Benutzer das Formular übergibt, wird der Block HTML(REPORT) aufgerufen. Wenn @mySelect() den FUNCTION-Block aufruft, wird $(Field) in der SQL-Anweisung durch customer.name bzw. customer.addr in der SQL-Abfrage ersetzt.

Listenvariablen

Mit Listenvariablen können Sie eine begrenzte Wertefolge erstellen. Diese Variablenart ist besonders hilfreich beim Erstellen einer SQL-Abfrage mit mehreren Elementen, die in einigen WHERE- oder HAVING-Klauseln auftreten. Die Syntax für eine Listenvariable sieht wie folgt aus:

```
%LIST " value_separator " variable_name
```

Empfehlung: Leerzeichen sind signifikante Zeichen. Fügen Sie in den meisten Fällen ein Leerzeichen vor und nach dem Wertetrennzeichen ein. Die meisten Abfragen verwenden boolesche oder mathematische Operatoren (z. B. AND, OR oder >) als Wertetrennzeichen.

Das folgende Beispiel zeigt die Verwendung von Bedingungsvariablen, verdeckten Variablen und Listenvariablen:

```
   %HTML(INPUT){
<form method="post" action="/cgi-bin/db2www/example2.dtw/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()
%}
```

Wenn im HTML-Formular keine Kästchen ausgewählt werden, ist `conditions` leer, so dass `whereClause` in der Abfrage ebenfalls leer ist. Andernfalls hat `whereClause` die durch OR getrennten Werte, die ausgewählt wurden. Wenn beispielsweise alle drei Städte ausgewählt werden, lautet die SQL-Abfrage:

```
SELECT name, city FROM citylist
WHERE cond1='Sao Paolo' OR cond2='Seattle' OR cond3='Shanghai'
```

Im Folgenden wird `Seattle` ausgewählt, so dass die SQL-Abfrage wie folgt aussieht:

```
SELECT name, city FROM citylist
WHERE cond1='Seattle'
```

Tabellenvariablen

Die Tabellenvariable definiert eine Sammlung zusammengehöriger Daten. Sie enthält eine Gruppe von Zeilen und Spalten einschließlich einer Zeile mit Spaltenüberschriften. Eine Tabelle wird in einem Net.Data-Makro wie in der folgenden Anweisung definiert:

```
%DEFINE myTable=%TABLE(30)
```

Die Zahl nach `%TABLE` gibt die maximale Anzahl der Zeilen an, die diese Tabellenvariable enthalten kann. Wenn Sie eine Tabelle ohne Zeilenbegrenzung angeben wollen, müssen Sie den Standardwert bzw. ALL angeben:

```
%DEFINE myTable2=%TABLE
%DEFINE myTable3=%TABLE(ALL)
```

Eine Tabelle hat bei ihrer Definition null Zeilen und null Spalten. Die einzige Möglichkeit, eine Tabelle mit Werten zu füllen, besteht darin, sie als Parameter OUT oder INOUT an eine Funktion zu übergeben oder die in Net.Data integrierten Tabellenfunktionen zu verwenden. Die Sprachumgebung DTW_SQL fügt die Ergebnisse einer Anweisung SELECT automatisch in eine Tabelle ein.

Bei Nicht-Datenbanksprachumgebungen, wie zum Beispiel DTW_REXX oder DTW_PERL, ist die Sprachumgebung auch dafür verantwortlich, die Tabellenwerte zu setzen. Das Script bzw. das Programm der Sprachumgebung definiert die Tabellenwerte jedoch zellweise. Weitere Informationen dazu, wie Tabellenvariablen von Sprachumgebungen verwendet werden, finden Sie in „Kapitel 6. Verwenden der Sprachumgebungen" auf Seite 151.

Sie können eine Tabelle zwischen Funktionen übergeben, indem Sie auf den Namen der Tabellenvariablen verweisen. Auf die einzelnen Elemente der Tabelle kann in einem REPORT-Block einer Funktion oder durch die Verwendung der Net.Data-Tabellenfunktionen verwiesen werden. Informationen zum Zugreifen auf einzelne Elemente in einer Tabelle innerhalb eines REPORT-Blocks finden Sie in „Variablen zur Tabellenverarbeitung" auf Seite 121. Informationen zum Zugreifen auf einzelne Elemente einer Tabelle mit einer Tabellenfunktion finden Sie in „Tabellenfunktionen" auf Seite 134. Tabellenvariablen werden in der Regel in einer SQL-Funktion mit Werten gefüllt und anschließend als Eingabe für einen Bericht entweder in der SQL-Funktion oder in einer anderen Funktion verwendet, nachdem sie als Parameter an diese Funktion übergeben wurden. Sie können Tabellenvariablen an eine beliebige Nicht-SQL-Funktion als Parameter IN, OUT oder INOUT übergeben. Tabellen können nur als Parameter OUT an SQL-Funktionen übergeben werden.

Wenn Sie auf eine Tabellenvariable verweisen, wird der Inhalt der Tabelle angezeigt und basierend auf der Einstellung der Variablen DTW_HTML_TABLE formatiert. Im folgenden Beispiel wird der Inhalt von myTable angezeigt:

```
%HTML (output) {
  $(myTable)
}
```

Die Spaltennamen und Feldwerte in einer Tabelle werden als Feldgruppenelemente mit dem Ursprung 1 angegeben.

Zusätzliche Variablen

Dies sind durch Net.Data definierte Variablen, die zu folgenden Zwecken verwendet werden können:

- Beeinflussen der Net.Data-Verarbeitung
- Ermitteln des Status eines Funktionsaufrufs
- Abrufen von Informationen zur Ergebnismenge einer Datenbankabfrage
- Bestimmen von Informationen zu Dateiadressen und Datumsangaben

Zusätzliche Variablen können entweder von Net.Data bestimmte vordefinierte Werte oder von Ihnen festgelegte Werte besitzen. Zum Beispiel bestimmt Net.Data den Wert der Variablen DTW_CURRENT_FILENAME nach der aktuellen Datei, die momentan verarbeitet wird, während Sie festlegen können, ob Net.Data zusätzliche, durch Tabulatoren und Zeilenvorschubzeichen verursachte Leerzeichen entfernen soll.

Vordefinierte Variablen werden innerhalb des Makros als Variablenverweise verwendet und liefern Informationen zum aktuellen Status von Dateien, Datumsangaben oder den Status eines Funktionsaufrufs. Sie könnten beispielsweise folgende Variable verwenden, um den Namen der aktuellen Datei abzurufen:

```
%REPORT {
  <p>This file is <i>$(DTW_CURRENT_FILENAME)</i>.</p>
}
```

Änderbare Variablenwerte werden gewöhnlich mit Hilfe einer Anweisung DEFINE oder der Funktion @DTW_ASSIGN() festgelegt und geben Ihnen die Möglichkeit, die Verarbeitung des Makros durch Net.Data zu beeinflussen. Mit der folgenden Anweisung DEFINE könnten Sie zum Beispiel angeben, dass zusätzliche Leerzeichen (White Space) entfernt werden sollen:

```
%DEFINE DTW_REMOVE_WS="YES"
```

Variablen zur Tabellenverarbeitung

Net.Data definiert Variablen zur Tabellenverarbeitung, die in REPORT- und ROW-Blöcken verwendet werden können. Diese Variablen dienen zum Verweisen auf Werte aus SQL-Abfragen und Funktionsaufrufen.

Die Variablen zur Tabellenverarbeitung besitzen einen vordefinierten Wert, der von Net.Data festgelegt wird. Diese Variablen ermöglichen Ihnen, auf Werte aus den Ergebnismengen von SQL-Abfragen oder Funktionsaufrufen nach Spalte, Zeile oder Feld, die bzw. das verarbeitet wird, zu verweisen. Außerdem können Sie auf Informationen zur Anzahl der verarbeiteten Zeilen oder auf eine Liste aller Spaltennamen zugreifen.

Bei der Verarbeitung der Ergebnismenge aus einer SQL-Abfrage ordnet Net-.Data zum Beispiel den Wert der Variablen Nn für jeden aktuellen Spaltennamen zu, so dass N1 der ersten Spalte, N2 der zweiten Spalte usw. zugeordnet wird. Sie können für Ihre Webseitenausgabe auf den aktuellen Spaltennamen verweisen.

Variablen zur Tabellenverarbeitung werden als Variablenverweise innerhalb des Makros verwendet. Zum Beispiel können Sie den Namen der aktuellen Spalte, die verarbeitet wird, folgendermaßen abrufen:

```
%REPORT {
  <p>Column 1 is <i>$(N1)</i>.</p>
}
```

Variablen zur Tabellenverarbeitung liefern außerdem Informationen zu den Ergebnissen einer Abfrage. Sie können im Makro auf die Variable TOTAL-_ROWS verweisen, um die Anzahl der von einer SQL-Abfrage zurückgegebenen Zeilen abzufragen, wie im folgenden Beispiel gezeigt:

```
Names found: $(TOTAL_ROWS)
```

Einige der Variablen zur Tabellenverarbeitung werden von anderen Variablen oder integrierten Funktionen beeinflusst. Zum Beispiel setzt die Variable TOTAL_ROWS voraus, dass die SQL-Sprachumgebungsvariable DTW_SET-_TOTAL_ROWS aktiviert ist, so dass Net.Data den Wert von TOTAL_ROWS zuordnet, wenn die Ergebnisse aus einer SQL-Abfrage oder einem Funktionsaufruf verarbeitet werden, wie in folgendem Beispiel gezeigt wird:

```
%DEFINE DTW_SET_TOTAL_ROWS="YES"
...

Names found: $(TOTAL_ROWS)
```

Berichtsvariablen

Net.Data zeigt Webseitenausgaben, die durch das Makro generiert werden, in einem Standardberichtsformat an. In einem HTML-Block zeigt das Standardberichtsformat eine Tabelle mit den Befehlen <pre> </pre> oder mit HTML-Tabellenbefehlen an. In einem XML-Block werden die Befehle <RowSet>, <Row> und <Column> verwendet. Sie können den Standardbericht außer Kraft setzen, indem Sie einen REPORT-Block mit Anweisungen zum Anzeigen der Ausgabe definieren oder eine der Berichtsvariablen verwenden, um die Generierung des Standardberichts zu verhindern.

Mit Berichtsvariablen können Sie die Anzeige der Webseitenausgabe und deren Verwendung für Standardberichte und Net.Data-Tabellen anpassen. Diese Variablen müssen vor ihrer Verwendung mit Hilfe einer DEFINE-Anweisung oder der Funktion @DTW_ASSIGN() definiert werden.

Die Berichtsvariablen definieren die Verwendung von Leerzeichen, überschreiben Standardberichtsformate, geben an, ob die Tabellenausgabe in HTML oder Monospace angezeigt werden soll, und bestimmen weitere Anzeigemerkmale. Sie können beispielsweise für DTW_HTML_TABLE "yes" angeben, und Net-.Data generiert den Standardbericht mit HTML-Tabellenbefehlen und nicht als Tabelle mit unverschlüsseltem Textformat.

```
%DEFINE ALIGN="YES"
...
<p>Your query was on these columns: $(NLIST)
```

Mit der Berichtsvariablen START_ROW_NUM können Sie festlegen, ab welcher Zeile die Ergebnisse einer Abfrage angezeigt werden sollen. Zum Beispiel gibt der folgende Variablenwert an, dass Net.Data die Ergebnisse einer Abfrage ab der dritten Zeile anzeigen soll:

```
%DEFINE START_ROW_NUM = "3"
```

Sie können außerdem festlegen, ob Net.Data HTML-Befehle für die Standardformatierung verwenden soll. Wenn der Wert der Variablen DTW_HTML-_TABLE auf YES gesetzt ist, wird keine Tabelle mit formatiertem Text erstellt, sondern eine HTML-Tabelle.

```
%DEFINE DTW_HTML_TABLE="YES"

%FUNCTION(DTW_SQL){
SELECT NAME, ADDRESS FROM $(qTable)
%}
```

Sprachumgebungsvariablen
Diese Variablen werden mit Sprachumgebungen verwendet und beeinflussen die Verarbeitung einer Anforderung durch die Sprachumgebung.

Mit diesen Variablen können Sie Aufgaben wie das Herstellen von Verbindungen zu Datenbanken, das Bereitstellen von alternativem Text für Java-Applets, das Aktivieren von Sprachenunterstützung und das Feststellen der erfolgreichen Ausführung einer SQL-Anweisung durchführen.

Zum Beispiel können Sie mit der Variablen SQL_STATE auf den von der Datenbank zurückgegebenen SQLSTATE-Wert zugreifen bzw. diesen Wert anzeigen.

```
%FUNCTION (DTW_SQL) val1() {
 select * from customer
%REPORT {
 ...
%ROW {
 ...
%}
 SQLSTATE=$(SQL_STATE)
%}
```

Das folgende Beispiel zeigt, auf welche Datenbank zugegriffen werden soll.

```
%DEFINE DATABASE="CELDIAL"
```

Net.Data-Funktionen

Net.Data stellt integrierte Funktionen zur Verwendung in Ihren Anwendungen bereit, wie Funktionen zur Wort- und Zeichenfolgebearbeitung und Funktionen, mit denen Tabellenvariablenfunktionen abgerufen und festgelegt werden. Sie können auch Funktionen zur Verwendung mit Ihrer Anwendung definieren, z. B. zum Aufrufen eines externen Programms oder einer gespeicherten Prozedur.

Benutzerdefinierte Funktionen
Diese Funktionen, die Sie zur Verwendung mit Ihrer Anwendung definieren, rufen z. B. ein externes Programm oder eine gespeicherte Prozedur auf.

Integrierte Net.Data-Funktionen
Die Funktionen, die von Net.Data zur Verwendung in Ihren Anwendungen bereitgestellt werden, etwa Funktionen zur Wort- und Zeichenfolgebearbeitung und Funktionen, mit denen Tabellenvariablen abgerufen und festgelegt werden.

In diesen Abschnitten werden die folgenden Themen beschrieben:

- „Definieren von Funktionen"
- „Aufrufen von Funktionen" auf Seite 130
- „Aufrufen von integrierten Net.Data-Funktionen" auf Seite 131

Definieren von Funktionen

Verwenden Sie zum Definieren eigener Funktionen im Makro einen FUNCTION-Block oder einen MACRO_FUNCTION-Block:

FUNCTION-Block
Definiert eine Unterroutine, die von einem Net.Data-Makro aufgerufen wird und von einer Sprachumgebung verarbeitet wird. FUNCTION-Blöcke müssen Sprachanweisungen oder Aufrufe an ein externes Programm enthalten.

MACRO_FUNCTION-Block
Definiert eine Unterroutine, die von einem Net.Data-Makro aufgerufen wird und von Net.Data anstatt von einer Sprachumgebung verarbeitet wird. MACRO_FUNCTION-Blöcke können beliebige, in einem HTML- oder XML-Block zulässige Anweisungen enthalten.

Syntax: Verwenden Sie die folgende Syntax zur Definition von Funktionen:

FUNCTION-Block:

```
%FUNCTION(type) function-name([usage] [datatype] parameter, ...)
                [RETURNS(return-var)] {
  executable-statements
  [report-block]
  ...
  [report-block]
  [message-block]
%}
```

MACRO_FUNCTION-Block:

```
%MACRO_FUNCTION function-name([usage] parameter, ...)  {
  executable-statements
  [report-block]
  ...
  [report-block]
%}
```

Dabei gilt Folgendes:

type Gibt eine Sprachumgebung an, die in der Initialisierungsdatei konfiguriert wird. Die Sprachumgebung ruft einen speziellen Sprachprozessor auf (der die ausführbaren Anweisungen verarbeitet) und stellt eine Standardschnittstelle zwischen Net.Data und dem Sprachprozessor bereit.

function-name

Gibt den Namen des FUNCTION- oder MACRO_FUNCTION-Blocks an. Ein Funktionsaufruf gibt *function-name* mit einem vorangestellten kommerziellen A (@) an. Nähere Informationen finden Sie in „Aufrufen von Funktionen" auf Seite 130.

Sie können mehrere FUNCTION-oder MACRO_FUNCTION-Blöcke mit dem gleichen Namen definieren, so dass sie gleichzeitig verarbeitet werden. Jeder der Blöcke musseine identische Parameterliste besitzen. Wenn Net.Data die Funktion aufruft, werden alle gleichnamigen FUNCTION-Blöcke bzw. gleichnamigen MACRO_FUNCTION-Blöcke in der Reihenfolge ausgeführt, in der sie im Net.Data-Makro definiert sind.

syntax Gibt an, ob ein Parameter ein Eingabeparameter (IN), ein Ausgabeparameter (OUT) oder beides (INOUT) ist. Diese Angabe bestimmt, ob der Parameter an den FUNCTION-Block bzw. den MACRO_FUNCTION-Block übergeben und/oder von ihm empfangen wird. Die Verwendungsart gilt für alle nachfolgenden Parameter in der Parameterliste, bis sie durch eine andere Verwendungsart geändert wird. Die Standardart ist IN.

datatype

Der Datentyp des Parameters. Einige Sprachumgebungen erwarten Datentypen für die übergebenen Parameter. Die SQL-Sprachumgebung z. B. erwartet sie beim Aufrufen gespeicherter Prozeduren. Weitere Informationen zu den unterstützten Datentypen für die verwendete Sprachumgebung finden Sie in „Kapitel 6. Verwenden der Sprachumgebungen" auf Seite 151.

parameter

Der Name einer Variablen mit lokalem Geltungsbereich, die durch den Wert eines entsprechenden in einem Funktionsaufruf angegebenen Arguments ersetzt wird. Die Parameter werden an die Sprachumgebung übergeben, und ausführbare Anweisungen, die die spezifische Syntax dieser Sprache verwenden, können auf die Parameter zugreifen oder sie als Umgebungsvariablen verwenden. Verweise auf Parametervariablen sind außerhalb der FUNCTION- bzw. MACRO_FUNCTION-Blöcke ungültig.

return-var

Geben Sie diesen Parameter nach dem Schlüsselwort RETURNS an, um einen speziellen OUT-Parameter zu definieren. Der Wert der Rückkehrvariablen wird im Funktionsblock zugeordnet, und sein Wert wird an der Stelle im Makro zurückgegeben, von der die Funktion aufgerufen wurde.

Zum Beispiel wird im folgenden Satz `<p>My name is @my_name().` die Angabe `@my_name()` durch den Wert der Rückkehrvariablen ersetzt. Wenn Sie die Klausel RETURNS nicht angeben, hat der Funktionsaufruf einen der folgenden Werte:

- NULL, falls der Rückkehrcode aus dem Aufruf an die Sprachumgebung Null ist
- Den Wert des Rückkehrcodes, wenn der Rückkehrcode ungleich Null ist

executable-statements

Die Gruppe der Sprachanweisungen, die an die angegebene Sprachumgebung zur Verarbeitung übergeben wird, nachdem die Variablen ersetzt und die Funktionen verarbeitet wurden. *executable-statements* können Net.Data-Variablenverweise und Net.Data-Funktionsaufrufe enthalten. *executable-statements* umfassen auch solche ausführbaren Anweisungen, die in einem HTML-Block zulässig sind.

Bei FUNCTION-Blöcken ersetzt Net.Data alle Variablenverweise durch die Variablenwerte, führt alle Funktionsaufrufe aus und ersetzt die Funktionsaufrufe mit den sich jeweils ergebenden Werten, bevor die ausführbaren Anweisungen an die Sprachumgebung übergeben werden. Jede Sprachumgebung verarbeitet die Anweisungen auf andere Weise. Weitere Informationen zur Angabe ausführbarer Anweisungen oder zum Aufrufen ausführbarer Programme finden Sie in „Ausführbare Variablen" auf Seite 115.

Bei MACRO_FUNCTION-Blöcken sind die ausführbaren Anweisungen eine Kombination aus Text und Net.Data-Makrosprachkonstrukten. In diesem Fall spielt die Sprachumgebung keine Rolle, weil Net.Data als Programmiersprachenprozessor fungiert und die ausführbaren Anweisungen verarbeitet.

report-block

Definiert mindestens einen REPORT-Block zur Behandlung der Ausgabe des FUNCTION- oder MACRO_FUNCTION-Blocks. Siehe „REPORT-Blöcke" auf Seite 138.

message-block

Definiert den MESSAGE-Block, der alle vom FUNCTION-Block zurückgemeldeten Nachrichten auf Fehlerbedingungen verarbeitet. Weitere Informationen zum Aufzeichnen von Fehlerbedingungen finden Sie in „MESSAGE-Blöcke" auf Seite 128.

Definieren Sie Funktionen außerhalb eines anderen Blocks und bevor diese im Net.Data-Makro aufgerufen werden.

Verwenden von Sonderzeichen in Funktionen

Wenn Zeichen, die der Syntax der Net.Data-Sprachkonstrukte entsprechen, in für die Sprachanweisungen eines FUNCTION-Blocks als Teil eines syntaktisch gültigen, eingebetteten Programmcodes (wie für REXX oder Perl) verwendet werden, können sie fälschlicherweise als Net.Data-Sprachkonstrukte interpretiert werden und so Fehler oder unvorhersehbare Ergebnisse in einem Makro verursachen.

Zum Beispiel könnte eine Perl-Funktion die Begrenzungszeichen für COMMENT-Blöcke (%{) verwenden. Bei der Ausführung des Makros werden die Zeichen %{ als Anfang eines COMMENT-Blocks interpretiert. Net.Data sucht dann nach dem Ende des COMMENT-Blocks, das von Net.Data am Ende des FUNCTION-Blocks erwartet wird. Net.Data sucht dann nach dem Ende des FUNCTION-Blocks und gibt, wenn das Ende nicht gefunden wird, einen Fehler aus.

Mit einer der folgenden Methoden können Sie die Begrenzungszeichen für COMMENT-Blöcke sowie alle anderen Net.Data-Sonderzeichen als Teil Ihres eingebetteten Programmcodes verwenden, ohne dass sie von Net.Data als Sonderzeichen interpretiert werden:

- Verwenden Sie die Anweisung EXEC zum Aufrufen des Programmcodes, anstatt den Code inline einzufügen.
- Verwenden Sie einen Variablenverweis zur Angabe der Sonderzeichen.

Die folgende Perl-Funktion zum Beispiel enthält als Teil ihrer Perl-Sprachanweisungen Zeichen, die eine COMMENT-Blockbegrenzung, %{, darstellen:

```
%FUNCTION(DTW_PERL) func() {
  ...
  for $num_words (sort bynumber keys %{ $Rtitles{$num} }) {
  &make_links($Rtitles{$num}{$num_words});
  }
  ...
%}
```

Um sicherzustellen, dass Net.Data die Zeichen %{ als Perl-Quellencode und nicht als eine COMMENT-Blockbegrenzung von Net.Data interpretiert, sollte die Funktion auf eine der folgenden Weisen umgeschrieben werden:

- Mit der Anweisung %EXEC:

```
%FUNCTION(DTW_PERL) func() {
  %EXEC{ func.prl %}
%}
```

- Mit einem Variablenverweis zur Angabe der Zeichen %{:

```
%define percent_openbrace = "%{"

%FUNCTION(DTW_PERL) func() {
  ...
  for $num_words (sort by number keys $(percent_openbrace) $Rtitles{$num} } {
  &make_links($Rtitles{$num}{$num_words});
  }
  ...
%}
```

MESSAGE-Blöcke

Anhand des MESSAGE-Blocks können Sie die weitere Vorgehensweise festlegen, nachdem ein Funktionsaufruf erfolgreich ausgeführt wurde bzw. fehlgeschlagen ist, und Informationen für den Aufrufenden der Funktion anzeigen. Beim Verarbeiten einer Nachricht setzt Net.Data die Sprachumgebungsvariable RETURN_CODE für jeden Funktionsaufruf auf einen FUNCTION-Block. RETURN_CODE wird bei einem Funktionsaufruf nicht auf einen MACRO_FUNCTION-Block gesetzt.

Ein MESSAGE-Block besteht aus einer Reihe von Nachrichtenanweisungen, von denen jede einen Rückkehrcodewert, einen Nachrichtentext und eine durchzuführende Aktion definiert. Die Syntax eines MESSAGE-Blocks wird im Kapitel über die Sprachkonstrukte im Handbuch *Net.Data Reference* gezeigt.

Ein MESSAGE-Block kann für einen globalen oder lokalen Bereich gelten. Wenn er in der äußeren Makroebene angegeben wird, hat der MESSAGE-Block einen globalen Geltungsbereich und ist für alle im Net.Data-Makro ausgeführten Funktionsaufrufe aktiv. Wenn Sie mehrere globale MESSAGE-Blöcke definieren, ist der zuletzt definierte Block aktiv. Wenn der MESSAGE-Block jedoch in einem FUNCTION-Block definiert ist, ist sein Geltungsbereich für diesen FUNCTION-Block lokal (mit Ausnahme von integrierten Net.Data-Funktionen, deren Fehler durch globale MESSAGE-Blöcke bearbeitet werden).

Net.Data verwendet die folgenden Regeln zur Verarbeitung des Werts der Variablen RETURN_CODE oder SQL_STATE von einem Funktionsaufruf:

1. Der lokale MESSAGE-Block wird auf eine exakte Übereinstimmung des Werts von RETURN_CODE oder SQL_STATE hin überprüft, und die Verarbeitung wird je nach Angabe beendet (exit) oder fortgesetzt (continue).
2. Wenn der Wert ungleich 0 ist, wird der lokale MESSAGE-Block auf `+default` bzw. `-default` hin überprüft, und die Verarbeitung wird abhängig vom Vorzeichen des Wert je nach Angabe beendet oder fortgesetzt.
3. Wenn der Wert ungleich 0 ist, wird der lokale MESSAGE-Block auf `default` hin überprüft, und die Verarbeitung je nach Angabe beendet oder fortgesetzt.

4. Der globale MESSAGE-Block wird auf eine exakte Übereinstimmung von RETURN_CODE oder SQL_STATE hin überprüft, und die Verarbeitung wird je nach Angabe beendet oder fortgesetzt.
5. Wenn der Wert ungleich 0 ist, wird der globale MESSAGE-Block auf `+default` bzw. `-default` hin überprüft, und die Verarbeitung wird abhängig vom Vorzeichen des Wert je nach Angabe beendet oder fortgesetzt.
6. Wenn der Wert ungleich 0 ist, wird der globale MESSAGE-Block auf `default` hin überprüft, und die Verarbeitung je nach Angabe beendet oder fortgesetzt.
7. Wenn der Wert ungleich 0 ist, gibt Net.Data die interne Standardnachricht aus und beendet die Verarbeitung.

Das folgende Beispiel zeigt einen Teil eines Net.Data-Makros mit einem globalen MESSAGE-Block und einem MESSAGE-Block für eine Funktion:

```
%{ 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
  %}
```

Wenn *my_function()* mit einem RETURN_CODE-Wert von 50 beendet wird, verarbeitet Net.Data den Fehler in folgender Reihenfolge:

1. Es wird überprüft, ob es eine exakte Übereinstimmung im lokalen MESSAGE-Block vorhanden ist.
2. Es wird überprüft, ob `+default` im lokalen MESSAGE-Block vorhanden ist.
3. Es wird überprüft, ob `default` im lokalen MESSAGE-Block vorhanden ist.
4. Es wird überprüft, ob eine exakte Übereinstimmung im globalen MESSAGE-Block vorhanden ist.
5. Es wird überprüft, ob `+default` im globalen MESSAGE-Block vorhanden ist.

Wird eine Übereinstimmung gefunden, sendet Net.Data den Nachrichtentext an den Web-Browser und überprüft die angeforderte Aktion.

Wenn Sie `continue` angeben, setzt Net.Data die Verarbeitung des Net.Data-Makros fort, nachdem der Nachrichtentext angezeigt wurde. Angenommen, ein Makro ruft `my_functions()` fünf Mal auf und bei der Verarbeitung mit dem MESSAGE-Block aus obigem Beispiel wird der Fehler 100 gefunden. In diesem Fall könnte die Ausgabe des Programms folgendermaßen aussehen:

```
.
.
.
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
```

Aufrufen von Funktionen

Verwenden Sie eine Net.Data-Funktionsaufrufanweisung, um benutzerdefinierte oder integrierte Funktionen aufzurufen. Verwenden Sie das kommerzielle A (@), gefolgt von einem Funktionsnamen bzw. Makrofunktionsnamen:

`@`*`function_name`*`([ `*`argument`*`,... ])`

function_name

Dies ist der Name der aufzurufenden Funktion bzw. Makrofunktion. Die Funktion muss bereits im Net.Data-Makro definiert sein, es sei denn, es handelt sich um eine integrierte Funktion.

argument

Dies ist der Name einer Variablen, eine Zeichenfolge in Anführungszeichen, ein Variablenverweis oder ein Funktionsaufruf. Die Argumente in einem Funktionsaufruf werden mit den Parametern in einer Funktions- oder Makrofunktionsparameterliste abgeglichen. Außerdem wird jedem Parameter bei der Verarbeitung der Funktion bzw. Makrofunktion der Wert des entsprechenden Arguments zugeordnet. Die Argumente müssen dieselbe Anzahl und Art wie die zugehörigen Parameter aufweisen.

Zeichenfolgen in Anführungszeichen als Argumente können Variablenverweise und Funktionsaufrufe enthalten.

Beispiel 1: Funktionsaufruf mit einem Zeichenfolgeargument

```
@myFunction("abc")
```

Beispiel 2: Funktionsaufruf mit einer Variablen und Funktionsaufrufargumenten

```
@myFunction(myvar, @DTW_rADD("2","3"))
```

Beispiel 3: Funktionsaufruf mit einem Zeichenfolgeargument, das einen Variablenverweis und einen Funktionsaufruf enthält

```
@myFunction("abc$(myvar)def@DTW_rADD("2","3")ghi")
```

Aufrufen von integrierten Net.Data-Funktionen

Net.Data verfügt über zahlreiche integrierte Funktionen, die die Entwicklung von Webseiten vereinfachen. Diese Funktionen sind von Net.Data bereits definiert, d. h. Sie brauchen sie nicht mehr zu definieren. Sie können diese Funktionen so wie andere Funktionen aufrufen.

Abb. 23 zeigt die Interaktion zwischen den integrierten Net.Data-Funktionen und dem Makro.

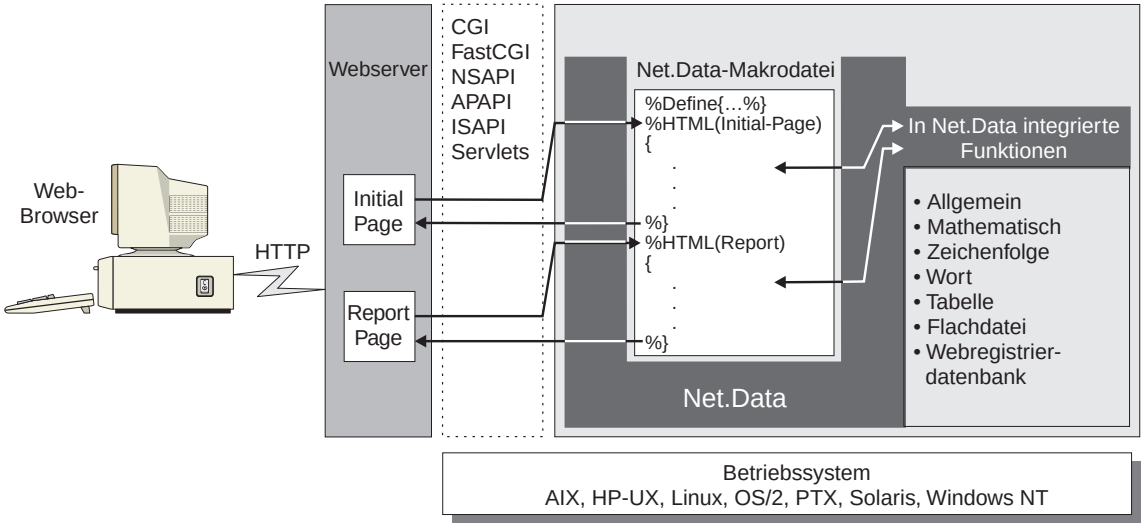


Abbildung 23. In Net.Data integrierte Funktionen

Es gibt drei Methoden, mit denen integrierte Funktionen ihre Ergebnisse zurückgeben können. Dies hängt vom Präfix ab.

- **DTW_, DTWF_ und DTWR_:** Die Ergebnisse des Aufrufs werden in einem Ausgabeparameter zurückgegeben, oder es wird kein Ergebnis zurückgegeben.(**DTWF_** ist das Präfix für Funktionen von unstrukturierten Textdateien. **DTWR_** ist das Präfix von Funktionen für Webregistrierungsdatenbanken.)
- **DTW_r und DTWR_r:** Die Ergebnisse des Funktionsaufrufs ersetzen den Funktionsaufruf im Makro. Ebenso ersetzt der Wert des Schlüsselworts RETURNS den Funktionsaufruf für eine benutzerdefinierte Funktion, in der das Schlüsselwort RETURNS angegeben ist.
- **DTW_m:** Mehrere Ergebnisse werden in allen Parametern zurückgegeben, die an die Funktion übergeben werden.

Einige integrierte Funktionen unterstützen nicht jede Art. Wenn Sie ermitteln möchten, um welche Art es sich bei einer bestimmten integrierten Funktion handelt, ziehen Sie das Kapitel zu den integrierten Net.Data-Funktionen im Handbuch *Net.Data Reference* zu Rate.

In den folgenden Abschnitten wird eine allgemeine Übersicht über die integrierten Net.Data-Funktionen gegeben. Mit diesen Funktionen können Sie allgemeine und mathematische Funktionen sowie Zeichenfolge-, Wort- bzw. Tabellenbearbeitungsfunktionen ausführen. Für einige dieser Funktionen müssen Variablen definiert werden, bevor sie verwendet werden können, oder sie müssen in einem spezifischen Kontext verwendet werden. Beschreibungen der einzelnen Funktionen mit Syntax und Beispielen finden Sie im Handbuch *Net.Data Reference*.

- „Allgemeine Funktionen”
- „Mathematische Funktionen” auf Seite 133
- „Zeichenfolgefunktionen” auf Seite 134
- „Wortfunktionen” auf Seite 134
- „Tabellenfunktionen” auf Seite 134
- „Funktionen für unstrukturierte Textdateien” auf Seite 135
- „Java-Applet-Funktionen” auf Seite 135
- „Funktionen für Webregistrierungsdatenbanken” auf Seite 136

Allgemeine Funktionen

Mit Hilfe dieser Gruppe von Funktionen können Sie Webseiten durch Ändern von Daten oder Zugreifen auf Systemservices entwickeln. Damit können Sie E-Mail senden, HTTP-Cookies verarbeiten, HTML-Escape-Codes generieren und andere nützliche Informationen vom System abrufen.

Sie verwenden zum Beispiel die Funktion DTW_EXIT, um festzulegen, dass Net.Data ein Makro verlassen soll, ohne die restlichen Anweisungen des Makros zu verarbeiten, wenn eine bestimmte Bedingung eintritt:

```
%HTML(sort_page) {

<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>
 %}
```

Eine weitere nützliche Funktion ist die Funktion DTW_URLESCSEQ, die Zeichen, die in einer URL-Adresse nicht zulässig sind, durch die entsprechenden Escape-Werte ersetzt. Wenn beispielsweise die Eingabevariable `string1` den Inhalt `"Guys & Dolls"` hat, ordnet DTW_URLESCSEQ die Ausgabevariable dem Wert `"Guys%20%26%20Dolls"` zu.

Mathematische Funktionen

Diese Funktionen führen mathematische Operationen aus, über die Sie numerische Daten berechnen und ändern können. Neben den mathematischen Standardoperationen können auch Modulus-Divisionen ausgeführt, eine Ergebnisgenauigkeit angegeben und Exponentialschreibweise verwendet werden.

Zum Beispiel potenziert die Funktion DTW_POWER den Wert des ersten Parameters mit dem Wert des zweiten Parameters und gibt das Ergebnis zurück. Siehe dazu das folgende Beispiel:

```
@DTW_POWER("2", "-3", result)
```

DTW_POWER gibt `".125"` in der Variablen `result` zurück.

Zeichenfolgefunktionen

Diese Funktionen können zum Bearbeiten von Zeichen in Zeichenfolgen verwendet werden. So können Sie zum Beispiel eine Zeichenfolge von Klein- in Großschreibung (oder umgekehrt) umsetzen, Zeichen einfügen oder löschen, einen Zeichenfolgewert einer anderen Variablen zuordnen und noch andere nützliche Funktionen ausführen.

Zum Beispiel können Sie DTW_ASSIGN verwenden, um einen Wert zuzuordnen oder den Wert einer Variablen zu ändern. Im folgenden Beispiel wird der Variablen `RC` der Wert Null zugeordnet.

```
@DTW_ASSIGN(RC, "0")
```

Weitere Zeichenfolgefunktionen sind DTW_CONCAT zum Verknüpfen von Zeichenfolgen und DTW_INSERT zum Einfügen von Zeichenfolgen an einer bestimmten Position sowie viele andere Funktionen zum Bearbeiten von Zeichenfolgen.

Wortfunktionen

Diese Funktionen können zum Bearbeiten von Wörtern in Zeichenfolgen verwendet werden. Die meisten dieser Funktionen arbeiten auf ähnliche Weise wie Zeichenfolgefunktionen, jedoch bezogen auf ganze Wörter. Hiermit können Sie zum Beispiel die Anzahl der Wörter in einer Zeichenfolge zählen, Wörter entfernen oder nach einem Wort in einer Zeichenfolge suchen. Verwenden Sie beispielsweise DTW_DELW0RD, um eine bestimmte Anzahl von Wörtern aus einer Zeichenfolge zu löschen:

```
@DTW_DELWORD("Now is the time", "2", "2", result)
```

DTW_DELWORD gibt die Zeichenfolge `"Now time"` zurück.

Weitere Wortfunktionen sind DTW_WORDLENGTH (gibt die Anzahl der Zeichen in einem Wort zurück) und DTW_WORDPOS (gibt die Position eines Worts in einer Zeichenfolge zurück).

Tabellenfunktionen

Diese Funktionen können zum Generieren von Berichten oder Formularen mit Hilfe der Daten in einer Net.Data-Tabellenvariable verwendet werden. Mit diesen Funktionen können Sie auch Net.Data-Tabellen erstellen sowie Werte in diesen Tabellen bearbeiten und abrufen. Tabellenvariablen enthalten eine Gruppe von Werten und ihre zugehörigen Spaltennamen. Sie bieten eine bequeme Möglichkeit, ganze Wertegruppen an eine Funktion weiterzugeben.

Zum Beispiel fügt DTW_TB_APPENDROW eine Zeile an die Tabelle an. Im folgenden Beispiel fügt Net.Data zehn Zeilen an die Tabelle `myTable` an:

```
@DTW_TB_APPENDROW(myTable, "10")
```

Außerdem gibt DTW_TB_DUMPH den Inhalt einer Makrotabellenvariablen zurück, die in die Befehle <pre></pre> eingeschlossen ist, wobei jede Zeile der Tabelle in einer anderen Zeile angezeigt wird. DTW_TB_CHECKBOX gibt mindestens einen HTML-Markierungsfeldeingabefehl aus einer Makrotabellenvariablen zurück.

Funktionen für unstrukturierte Textdateien

Die FFI-Schnittstelle (Flat File Interface - Schnittstelle für unstrukturierte Textdateien) kann zum Öffnen, Lesen und Bearbeiten von Daten aus unstrukturierten Textdateiquellen (Textdateien) sowie zum Speichern von Daten in unstrukturierte Textdateien verwendet werden.

Zum Beispiel schreibt DTWF_APPEND den Inhalt einer Tabellenvariablen an das Ende einer Datei, und DTWF_DELETE löscht Sätze aus einer Datei. Außerdem unterstützen die FFI-Funktionen Dateisperren mit DTWF_CLOSE und DTWF_OPEN. DTWF_OPEN sperrt eine Datei, damit sie eine andere Anforderung weder lesen noch aktualisieren kann. DTWF_CLOSE gibt die Datei frei, wenn Net.Data sie nicht länger benötigt, damit andere Anforderungen auf die Datei zugreifen können.

Java-Applet-Funktionen

Mit Hilfe der Java-Applet-Funktionen können Sie <applet>- und <parm>-Befehle auf der Grundlage von Net.Data-Variablen auf einfache Weise für Ihre Webseite generieren.

Wenn Sie beispielsweise über das Applet `myApplet` verfügen und einige Parameter an das Applet übergeben wollen (einschließlich einer Tabellenvariablen), könnten Sie Folgendes angeben:

```
%define REMOTE_USER = %ENVVAR
 %define myTable = %TABLE(all)
 ...
%HTML(report){
 ...
 @DTWA_myApplet(REMOTE_USER, myTable)
 ...
 %}
```

Hierdurch wird Net.Data angewiesen, für jeden Wert in der Tabelle und für den Wert der Umgebungsvariablen REMOTE_USER einen Befehl <applet> und <parm> zu generieren.

Außerdem können Sie eine einzelne Spalte der Tabelle übergeben. Beispiel:

```
@DTWA_myApplet(REMOTE_USER, DTW_COLUMN(mycol)myTable)
```

Dieses Beispiel übergibt die Spalte `mycol` der Net.Data-Tabellenvariablen `myTable`.

Funktionen für Webregistrierungsdatenbanken

Die Funktionen für Webregistrierungsdatenbanken können zum Verwalten der registrierungsdatenbanken und der darin enthaltenen Einträge verwendet werden. Eine Webregistrierungsdatenbank ist eine Datei mit einem von Net-.Data verwalteten Schlüssel, die das einfache Hinzufügen, Abrufen und Löschen von Einträgen ermöglicht.

Zum Beispiel fügt DTWR_ADDENTRY Einträge hinzu, während DTWR-_DELENTRY Einträge löscht. DTWR_LISTSUB gibt Informationen zu den registrierungsdatenbankeinträgen in einem OUT-Tabellenparameter zurück, und DTWR_UPDATEENTRY ersetzt vorhandene Werte für einen angegebenen Registrierungsdatenbankeintrag durch einen neuen Wert.

Generieren der Dokumentformatierung

Net.Data generiert HTML- oder XML-Dokumente, die eine Client-Anwendung, z. B. ein Web-Browser, verwenden soll, dynamisch. Die folgenden Abschnitte beschreiben die verschiedenen Konstrukte, mit denen Sie Dokumente mit Net.Data-Makros formatieren können. Informationen zur spezifischen Syntax finden Sie im Kapitel über die Sprachkonstrukte im Handbuch *Net.Data Reference*.

HTML- und XML-Blöcke

Die Client-Anwendung ruft Net.Data auf, indem sie den Makronamen und den Namen eines Makroeingangspunkts angibt. Der Eingangspunkt für das Makro kann ein HTML- oder ein XML-Block sein. Diese Blöcke enthalten die Net.Data-Sprach- und -Textdarstellungsanweisungen, die die resultierende Seite generieren.

Da der Eingangspunktblock die Ausführung des Makros steuert, muss mindestens ein Eingangspunkt in einem Makro enthalten sein. Es können mehrere HTML- oder XML-Blöcke vorhanden sein. Es wird jedoch nur einer pro Client-Anforderung ausgeführt. Und mit jeder Anforderung wird ein einzelnes Dokument an den Client geliefert. Wenn Sie eine Anwendung erstellen möchten, die aus mehreren Client-Dokumenten besteht, können Sie Net.Data mehrfach aufrufen, um verschiedene HTML- oder XML-Blöcke in verschiedenen Makros mit normalen Navigationsverfahren, wie z. B. Verbindungen (Links) und Formularen, zu verarbeiten.

Jede Textdarstellungsanweisung kann in einem HTML- oder XML-Block erscheinen, solange die Anweisungen für den Client gültig sind. HTML-Blöcke können beispielsweise HTML oder JavaScript enthalten. JavaScript wird nicht von Net.Data ausgeführt, sondern mit dem Rest der Ausgabe zur Ausführung und Anzeige an den Client gesendet. In einen HTML- oder XML-Block können Sie auch Funktionsaufrufe, Variablenverweise und INCLUDE-Anweisungen einfügen.

Das folgende Beispiel zeigt eine gängige Verwendung eines HTML-Blocks in einem Net.Data-Makro:

```
%HTML(input){
<h1>Hardware Query Form</h1>
<hr/>
<form method="post" action="report">
<dl>
</dt>What hardware do you want to list?
<dd><input type="radio" name="hdware" value="MON" checked />Monitors</ddl>
<dd><input type="radio" name="hdware" value="PNT" />Pointing devices</ddl>
<dd><input type="radio" name="hdware" value="PRT" />Printers</ddl>
<dd><input type="radio" name="hdware" value="SCN" />Scanners</ddl>
</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()
%}
```

Sie können das Net.Data-Makro über eine HTML-Verbindung aufrufen.

```
<a href="http://www.ibm.com/cgi-bin/db2www/equiplst.dtw/input">
  List of hardware</a>
```

Wenn der Anwendungsbenutzer diese Verbindung (Link) anklickt, ruft der Web-Browser Net.Data auf, und Net.Data analysiert das Makro. Wenn Net.Data mit der Verarbeitung des im Aufruf angegebenen HTML-Blocks, in diesem Fall input, beginnt, fängt Net.Data an, den Text innerhalb des Blocks zu verarbeiten. Alles, was Net.Data nicht als Net.Data-Makrosprachkonstrukt erkennt, wird zum Anzeigen an den Browser gesendet.

Wenn der Benutzer eine Auswahl getroffen und den Knopf zur Übergabe (Submit) angeklickt hat, fordert der Client die im Aktionsattribut des HTML-Formulars angegebene Aktion an. Diese Aktion gibt einen Aufruf des HTML-Blocks output des Makros an. Net.Data verarbeitet dann den HTML-Block output wie den HTML-Block input.

Anschließend verarbeitet Net.Data den Funktionsaufruf `myQuery()`, der wiederum den Block FUNCTION der SQL-Sprachumgebung aufruft. Nachdem der Variablenverweis `$(hdware)` in der SQL-Anweisung durch den vom Eingabeformular zurückgegebenen Wert ersetzt wurde, führt Net.Data die Abfrage aus. An dieser Stelle nimmt Net.Data die Verarbeitung des Berichts wieder auf, der die Ergebnisse der Abfrage gemäß den im REPORT-Block definierten Textdarstellungsanweisungen anzeigt.

Nachdem Net.Data die Verarbeitung des REPORT-Blocks abgeschlossen hat, kehrt es zum HTML-Block output zurück und beendet die Verarbeitung.

REPORT-Blöcke

Das Sprachkonstrukt des REPORT-Blocks dient zum Formatieren und Anzeigen der Datenausgabe eines FUNCTION-Blocks. Diese Ausgabe besteht in der Regel aus Tabellendaten, obwohl jede gültige Kombination aus Text, Makrovariablenverweisen und Funktionsaufrufen angegeben werden kann. Optional kann im REPORT-Block ein Tabellenname angegeben werden. Mit Ausnahme der SQL- und ODBC-Sprachumgebungen gilt Folgendes: Wenn kein Tabellenname angegeben wird, verwendet Net.Data die Tabellendaten aus der ersten Ausgabetabelle in der FUNCTION-Parameterliste.

Der REPORT-Block besteht aus drei Teilen, die jeweils optional sind:

- Kopfdaten mit Text, der einmal vor den Daten in der Tabellenzeile angezeigt wird
- Ein ROW-Block mit Text und Tabellenvariablen, die einmal für jede Zeile der Ergebnistabelle angezeigt werden
- Fußzeilendaten mit Text, der einmal nach den Daten in der Tabellenzeile angezeigt wird

Beispiel:

```
%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.dtw/details?name=$(V1)&loc=$(V2)">$(V1)</a>
    </td>
    <td>$(V2)</td>
  </tr>
  %}
</table>
%}
```

Richtlinien für REPORT-Blöcke

Halten Sie die folgenden Richtlinien ein, wenn Sie REPORT-Blöcke erstellen:

- Wenn Sie keine Tabellenausgabe des ROW-Blocks anzeigen wollen, nehmen Sie keine Angaben im ROW-Block vor, oder übergehen Sie ihn ganz.
- Verwenden Sie verschiedene von Net.Data bereitgestellte Variablen im REPORT-Block für den Zugriff auf die Daten in der Net.Data-Makroergebnistabelle.

  Diese Variablen sind in „Variablen zur Tabellenverarbeitung" auf Seite 121 beschrieben. Weitere Einzelheiten finden Sie im Abschnitt über die Berichtsvariablen im Handbuch *Net.Data Reference*.
- Wenn Sie Kopf- und Fußzeilendaten zur Verfügung stellen wollen, geben Sie den Text vor und nach dem ROW-Block an. Net.Data verarbeitet alle Angaben, die vor einem ROW-Block angetroffen werden, als Kopfdaten. Net.Data verarbeitet alle Angaben, die nach dem ROW-Block angetroffen werden, als Fußzeilendaten. Wie beim HTML-Block behandelt Net.Data alle Angaben im Kopfdatenblock, ROW-Block und Fußzeilendatenblock, die nicht als Makrosprachkonstrukte erkannt werden, als Textdarstellungsanweisungen und sendet diese Anweisungen an den Browser.
- Sie können Funktionen und Verweisvariablen in einem REPORT-Block aufrufen.
- Wenn Net.Data einen Standardbericht mit vorformatiertem Text ausgeben soll, nehmen Sie den REPORT-Block nicht in das Makro auf. Das folgende Beispiel zeigt das Standardberichtsformat, wenn die Funktion in einem HTML-Block aufgerufen wird:

```
SHIPDATE   | RECDATE    | SHIPNO   |
-------------------------------------
25/05/1997 | 30/05/1997 | 1495194B |
-------------------------------------
25/05/1997 | 28/05/1997 | 2942821G |
-------------------------------------
```

- Wenn Sie die HTML-Befehle anstelle des vorformatierten Textes verwenden wollen, setzen Sie DTW_HTML_TABLE auf YES.
- Um die Ausgabe des Standardberichts zu inaktivieren, setzen Sie DTW_DEFAULT_REPORT auf den Wert NO oder geben einen leeren REPORT-Block an. Beispiel:

```
%REPORT{%}
```

Beispiel: Anpassen eines Berichts

Das folgende Beispiel zeigt, wie Sie Berichtsformate mit Hilfe spezieller Variablen und HTML-Befehle anpassen können. Es werden die Namen, Telefonnummern und Faxnummern aus der Tabelle `CustomerTbl` angezeigt:

```
%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)
  %}
%}
```

Der hieraus erstellte Bericht sieht im Web-Browser wie folgt aus:

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

Der Bericht wurde von Net.Data wie folgt generiert:

1. Der Titel *Phone Query Results:* wird einmal am Anfang des Berichts ausgegeben. Dieser Text bildet zusammen mit einer Trennlinie den Kopfzeilenbereich des REPORT-Blocks.
2. Die Variablen `V1`, `V2` und `V3` werden beim Abruf der einzelnen Zeilen durch die Werte für *Name*, *Phone* und *Fax* ersetzt.

3. Die Zeichenfolge *Total records retrieved:* und der Wert für TOTAL_ROWS werden einmal am Ende des Berichts ausgegeben. (Dieser Text bildet den Fußzeilenbereich des REPORT-Blocks.)

Mehrere REPORT-Blöcke

Sie können mehrere REPORT-Blöcke in einem einzelnen FUNCTION- oder MACRO FUNCTION-Block angeben, um mehrere Berichte mit einem Funktionsaufruf zu generieren.

In der Regel werden mehrere REPORT-Blöcke in der Sprachumgebung DTW_SQL mit einer Funktion verwendet, die eine gespeicherte Prozedur aufruft, die mehrere Ergebnismengen zurückgibt (siehe „Gespeicherte Prozeduren" auf Seite 164). Mehrere REPORT-Blöcke können jedoch in jeder Sprachumgebung verwendet werden, um mehrere Berichte zu erstellen.

Stellen Sie zur Verwendung mehrerer REPORT-Blöcke für jede Ergebnismenge einen Ergebnismengennamen in die Anweisung CALL der gespeicherten Prozedur. Wenn mehr Ergebnismengen von der gespeicherten Prozedur zurückgegeben werden als Sie REPORT-Blöcke angegeben haben und wenn die Angabe der integrierten Net.Data-Funktion DTW_DEFAULT_REPORT = "MULTIPLE" ist, werden Standardberichte für jede Tabelle definiert, die keinem REPORT-Block zugeordnet ist. Wenn keine REPORT-Blöcke angegeben sind und DTW_DEFAULT_REPORT = "YES" ist, wird nur ein Standardbericht generiert. Beachten Sie, dass der Wert "YES" für DTW_DEFAULT_REPORT nur in der SQL-Sprachumgebung dem Wert "MULTIPLE" entspricht.

Beispiele: Die folgenden Beispiele zeigen, wie Sie mehrere REPORT-Blöcke verwenden können.

Gehen Sie wie folgt vor, um mehrere Berichte mit den Standardberichtsformaten anzuzeigen:

Beispiel 1: Sprachumgebung DTW_SQL

```
%DEFINE DTW_DEFAULT_REPORT = "MULTIPLE"
%FUNCTION(dtw_sql) myStoredProc  () {
     CALL myproc (table1, table2)  %}
```

In diesem Beispiel gibt die gespeicherte Prozedur myproc zwei Ergebnismengen zurück, die in table1 und table2 gestellt werden. Weil keine REPORT-Blöcke angegeben sind, werden Standardberichte für beide Tabellen angezeigt, zuerst für table1 und dann für table2.

Beispiel 2: MACRO_FUNCTION-Block. In diesem Beispiel werden zwei Tabellen an den MACRO_FUNCTION-Block übergeben. Wenn DTW_DEFAULT_REPORT=″MULTIPLE″ angegeben wird, generiert Net.Data Berichte für beide Tabellen.

```
%DEFINE DTW_DEFAULT_REPORT = "MULTIPLE"
%MACRO_FUNCTION multReport (INOUT tablename1, tablename2) {
%}
```

In diesem Beispiel werden zwei Tabellen an die MACRO_FUNCTION-Funktion `multReport` übergeben. Net.Data zeigt wiederum Standardberichte für die zwei Tabellen in der Reihenfolge an, in der sie in der Parameterliste des MACRO FUNCTION-Blocks erscheinen: zuerst für `table1` und dann für `table2`.

Beispiel 3: Sprachumgebung DTW_REXX

```
%DEFINE DTW_DEFAULT_REPORT = "YES"
%FUNCTION (dtw_rexx) multReport (INOUT table1, table2) {
     SAY 'Generating multiple default reports...<br />'
%}
```

In diesem Beispiel werden zwei Tabellen an die REXX-Funktion `multReport` übergeben. Da DTW_DEFAULT_REPORT=″YES″ angegeben ist, zeigt Net.Data nur für die erste Tabelle einen Standardbericht an.

Gehen Sie wie folgt vor, um mehrere Berichte durch Angabe von REPORT-Blöcken zur Anzeigeverarbeitung anzuzeigen:

Beispiel 1: Benannte REPORT-Blöcke

```
%FUNCTION(dtw_sql) myStoredProc  () {
   CALL myproc (table1, table2)

  %REPORT(table2) {
     ...
    %ROW {  ....   %}
     ...
   %}

   %REPORT(table1) {
     ...
 %row {  ....   %}
     ...
   %}
%}
```

In diesem Beispiel wurden REPORT-Blöcke für beide der in der Parameterliste des FUNCTION-Blocks angegebenen Tabellen angegeben. Die Tabellen werden in der Reihenfolge angezeigt, in der sie in den REPORT-Blöcken angegeben sind: `table2` zuerst und dann `table1`. Durch Angabe eines Tabellennamens in den REPORT-Blöcken und in der Anweisung CALL können Sie die Reihenfolge steuern, in der die Berichte angezeigt werden.

Beispiel 2: Nicht benannte REPORT-Blöcke

```
%FUNCTION(dtw_sql) myStoredProc  () {
     CALL  myproc

%REPORT {
      ...
    %ROW {  ....   %}
      ...
   %}
%REPORT {
      ...
    %ROW {  ....   %}
      ...
   %}
%}
```

In diesem Beispiel wurden REPORT-Blöcke für zwei von myproc zurückgegebene Ergebnismengen angegeben. Da keine Tabellennamen in den REPORT-Blöcken angegeben sind, werden die REPORT-Blöcke für die ersten beiden Ergebnismengen in der Reihenfolge ausgeführt, in der sie von der gespeicherten Prozedur zurückgegeben werden.

Gehen Sie wie folgt vor, um mehrere Berichte mit einer Kombination aus Standardberichten und REPORT-Blöcken anzuzeigen:

Beispiel: Eine Kombination aus Standardberichten und REPORT-Blöcken

```
%DEFINE DTW_DEFAULT_REPORT = "MULTIPLE"
%FUNCTION(dtw_system) editTables (INOUT table1, table2, table3) {
  %REPORT(table2) {
     ...
    %ROW {  ....   %}
     ...
  %}
%}
```

In diesem Beispiel ist nur ein REPORT-Block angegeben. Da der Block `table2` angibt und `table2` die zweite in der Anweisung CALL aufgelistete Ergebnismenge ist, wird die zweite Ergebnismenge zum Anzeigen des Berichts verwendet.

Da weniger REPORT-Blöcke angegeben sind als Ergebnismengen von der gespeicherten Prozedur übergeben werden, werden Standardberichte für die

restlichen Ergebnismengen in der folgenden Reihenfolge angezeigt: zuerst ein Standardbericht für die erste Ergebnismenge `table1` und dann ein Standardbericht für die dritte Ergebnismenge `table3`. Es ist eine Ausgabetabelle angegeben, `table1`, die zur späteren Verarbeitung im Makro verwendet werden kann.

Richtlinien und Einschränkungen für mehrere REPORT-Blöcke: Beachten Sie die folgenden Richtlinien und Einschränkungen für die Angabe mehrerer REPORT-Blöcke in einem FUNCTION- oder MACRO_FUNCTION-Block.

Richtlinien:

- Sie können einen REPORT-Block pro Ergebnismenge angeben.
- Geben Sie die REPORT-Blöcke für mehrere Tabellen in der Reihenfolge an, in der sie verarbeitet werden sollen.
- Soll die Standardverarbeitung erfolgen, wenn kein REPORT-Block für eine Tabelle angegeben ist, definieren Sie DTW_DEFAULT_REPORT = ″MULTIPLE″. Beim Aufbau der Webseite zeigt Net.Data die Standardberichte für Tabellen nach den Berichten für Tabellen mit REPORT-Blöcken an.
- Wenn Net.Data Tabellen ohne REPORT-Blöcke nicht anzeigen soll, definieren Sie DTW_DEFAULT_REPORT = ″NO″.
- Bei Verwendung der Variablen DTW_SAVE_TABLE_IN mit einer Funktion, die mehrere Ergebnismengen zurückgibt, wird die erste von der Funktion zurückgegebene Ergebnismenge der Tabelle DTW_SAVE_TABLE_IN zugeordnet.
- Mehrere REPORT-Blöcke können in jeder Sprachumgebung verwendet werden.

Einschränkungen:

- Die Werte aller Berichtsvariablen in einer Funktion, z. B. START_R_N und RPT_M_R, gelten für alle REPORT-Blöcke in dieser Funktion. Sie können den Wert einer Berichtsvariablen nicht für einzelne REPORT-Blöcke ändern.
- Ein MESSAGE-Block muss sich entweder vor oder nach einer Reihe von REPORT-Blöcken befinden. Er darf nicht zwischen REPORT-Blöcken stehen.
- Wenn der erste REPORT-Block einen Tabellennamen angibt, dann müssen alle REPORT-Blöcke Tabellennamen angeben.
- Wenn der erste REPORT-Block keinen Tabellennamen angibt, dann darf kein REPORT-Block Tabellennamen angeben.
- Die maximale Anzahl von Tabellen für eine einzige gespeicherte Prozedur ist 32.

Bedingte Logik und Schleifen in einem Makro

Net.Data ermöglicht Ihnen, bedingte Logik und Schleifen in Ihr Net.Data-Makro mit Hilfe von IF- und WHILE-Blöcken zu integrieren.

IF- und WHILE-Blöcke verwenden eine Bedingungsliste, mit der Sie mindestens eine Bedingung testen können. Anschließend können Sie auf der Grundlage des Ergebnisses des Bedingungstests einen Block von Anweisungen ausführen. Die Bedingungsliste enthält logische Operatoren, wie = und <=, und Terme, die sich aus Zeichenfolgen in Anführungszeichen, Variablen, Variablenverweisen und Funktionsaufrufen zusammensetzen. Zeichenfolgen in Anführungszeichen können zudem Variablenverweise und Funktionsaufrufe enthalten. Sie können die Bedingungsliste verschachteln.

In den folgenden Abschnitten wird die Verwendung der bedingten Logik und von Schleifen beschrieben:

- „Bedingte Logik: IF-Blöcke"
- „Schleifenkonstrukte: WHILE-Blöcke" auf Seite 148

Bedingte Logik: IF-Blöcke

Mit Hilfe des IF-Blocks können Sie in einem Net.Data-Makro eine bedingte Verarbeitung durchführen. Der IF-Block ist den IF-Anweisungen der meisten höheren Programmiersprachen ähnlich, weil er die Möglichkeit bietet, mindestens eine Bedingung zu testen und anschließend auf der Grundlage des Ergebnisses des Bedingungstests einen Block von Anweisungen auszuführen.

Sie können IF-Blöcke fast überall in einem Makro verwenden und verschachteln. Die Syntax eines IF-Blocks wird im Kapitel über die Sprachkonstrukte im Handbuch *Net.Data Reference* dargestellt.

Regeln für IF-Block: Die Syntaxregeln für einen IF-Block werden durch die Position des Blocks im Makro bestimmt. Die Elemente, die im Block der ausführbaren Anweisungen eines IF-Blocks zulässig sind, hängen von der Position des IF-Blocks selbst ab.

- Jedes Element, das innerhalb des Blocks, in dem sich der IF-Block befindet, gültig ist, ist auch innerhalb dieses IF-Blocks gültig. Wenn Sie zum Beispiel einen IF-Block innerhalb eines HTML-Blocks angeben, ist jedes Element, das in dem HTML-Block zulässig ist, auch in diesem IF-Block zulässig, wie zum Beispiel INCLUDE-Anweisungen und WHILE-Blöcke.

```
%HTML block
...
     %IF block
...
    %INCLUDE
...
    %WHILE
...
%ENDIF
%}
```

- Wenn Sie den IF-Block außerhalb jedes anderen Blocks im Deklarationsabschnitt des Net.Data-Makros angeben, sind analog nur solche Elemente in dem IF-Block zulässig, die außerhalb jedes anderen Blocks zulässig sind (wie zum Beispiel ein DEFINE-Block oder ein FUNCTION-Block).

```
%IF
...
  %DEFINE
...
  %FUNCTION
...
%ENDIF
```

- Wenn ein IF-Block in einem IF-Block verschachtelt ist, der sich außerhalb jedes anderen Blocks im Deklarationsabschnitt befindet, können in diesem Block alle Elemente verwendet werden, die im außerhalb liegenden Block verwendet werden können. Wenn ein IF-Block in einem anderen Block verschachtelt ist, der sich in einem IF-Block befindet, übernimmt er die Syntaxregeln des Blocks, in dem er sich befindet.

  Im folgenden Beispiel muss der verschachtelte IF-Block den Regeln folgen, die innerhalb eines HTML-Blocks gelten.

```
%IF
...
  %HTML {
...
%IF
...
%ENDIF
  %}
...
%ENDIF
```

  Ausnahme: Geben Sie keinen ROW-Block in einem IF-Block an.

Zeichenfolgevergleiche für IF-Blöcke

Net.Data verarbeitet die Bedingungsliste des IF-Blocks auf eine von zwei Arten, je nach dem Inhalt der Ausdrücke, aus denen die Bedingungen bestehen. Die Standardaktion besteht darin, alle Ausdrücke als Zeichenfolgen zu behandeln und Zeichenfolgevergleiche wie in den Bedingungen angegeben durchzuführen. Wenn jedoch zwei Zeichenfolgen, die ganze Zahlen darstellen, verglichen werden, ist der Vergleich numerisch. Net.Data geht davon aus, dass eine Zeichenfolge numerisch ist, wenn sie nur Ziffern enthält, denen optional ein Zeichen '+' oder '-' vorangestellt sein kann. Die Zeichenfolge darf keine Nichtziffernzeichen außer '+' bzw. '-' enthalten. Net.Data unterstützt keinen numerischen Vergleich von nicht ganzzahligen Zahlen.

Beispiele für gültige Ganzzahlenfolgen:

```
+1234567890
-47
000812
92000
```

Beispiele für ungültige Ganzzahlenfolgen:

```
-  20     (enthält Leerzeichen)
234,000   (enthält ein Komma)
57.987    (enthält einen Punkt)
```

Net.Data wertet die IF-Bedingung zum Zeitpunkt der Ausführung des Blocks aus. Dieser Zeitpunkt muss nicht mit dem Zeitpunkt, zu dem der Block ursprünglich von Net.Data gelesen wurde, übereinstimmen. Wenn Sie zum Beispiel einen IF-Block in einem REPORT-Block angeben, wertet Net.Data die Bedingungsliste des IF-Blocks nicht beim Lesen der FUNCTION-Blockdefinition mit dem REPORT-Block aus, sondern erst beim Aufrufen und Ausführen der Funktion. Dies gilt sowohl für den Abschnitt mit der Bedingungsliste des IF-Blocks als auch für den Block der auszuführenden Anweisungen.

Beispiel für IF-Block: Ein Makro mit IF-Blöcken in anderen Blöcken

```
%{ 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)
  %}
%}
```

Schleifenkonstrukte: WHILE-Blöcke

Mit Hilfe des WHILE-Blocks können Schleifen in einem Net.Data-Makro durchgeführt werden. Wie der IF-Block bietet der WHILE-Block die Möglichkeit, mindestens eine Bedingung zu testen und anschließend auf der Grundlage des Ergebnisses des Bedingungstests einen Block von Anweisungen auszuführen. Im Gegensatz zum IF-Block kann der Anweisungsblock auf der Grundlage des Ergebnisses des Bedingungstests beliebig oft ausgeführt werden.

Sie können WHILE-Blöcke innerhalb von HTML-Blöcken, REPORT-Blöcken, ROW-Blöcken, MACRO_FUNCTION-Blöcken und IF-Blöcken angeben, und Sie können sie verschachteln. Die Syntax eines WHILE-Blocks wird im Kapitel über die Sprachkonstrukte im Handbuch *Net.Data Reference* gezeigt.

Net.Data verarbeitet den WHILE-Block genau so wie den IF-Block, jedoch wird die Bedingung nach jeder Ausführung des Blocks erneut ausgewertet. Und wie bei jedem bedingten Schleifenkonstrukt kann die Verarbeitung zu einer Endlosschleife führen, wenn die Bedingung nicht korrekt codiert wurde.

Beispiel: Ein Makro mit einem WHILE-Block

```
%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)
  %}
%}
```

Kapitel 6. Verwenden der Sprachumgebungen

Net.Data stellt Sprachumgebungen bereit, mit denen Sie auf Datenquellen zugreifen und Anwendungsprogramme mit Geschäftslogik ausführen können. Mit der SQL-Sprachumgebung können Sie zum Beispiel SQL-Anweisungen an eine DB2-Datenbank übergeben, und mit der REXX-Sprachumgebung können Sie REXX-Programme aufrufen. Mit der SYSTEM-Sprachumgebung können Sie ein Programm ausführen oder einen Befehl absetzen.

In Net.Data können Sie benutzerdefinierte Sprachumgebungen wie Plug-Ins hinzufügen. Jede benutzerdefinierte Sprachumgebung muss eine Standardgruppe von Schnittstellen unterstützen, die von Net.Data definiert werden, und muss als Dynamic Link Library (DLL) oder gemeinsam benutzte Bibliothek implementiert werden. Ausführliche Informationen zu den von Net.Data bereitgestellten Sprachumgebungen und zum Erstellen einer benutzerdefinierten Sprachumgebung finden Sie im Handbuch *Net.Data Language Environment Interface Reference*.

Abb. 24 auf Seite 152 zeigt die Beziehung zwischen dem Webserver, Net.Data und den Net.Data-Sprachumgebungen.

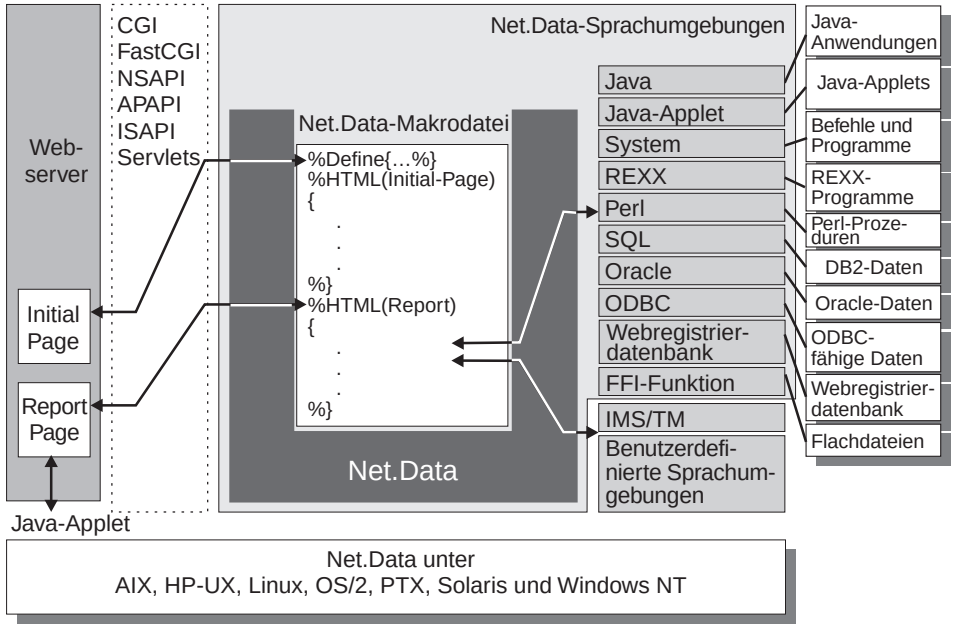


Abbildung 24. Die Net.Data-Sprachumgebungen

In den folgenden Abschnitten werden die Net.Data-Sprachumgebungen und ihre Verwendung in Ihren Makros beschrieben:

- „Übersicht über die von Net.Data bereitgestellten Sprachumgebungen"
- „Aufrufen einer Sprachumgebung" auf Seite 154
- „Sprachumgebungen für relationale Datenbanken" auf Seite 155
- „Sprachumgebungen für Programmiersprachen" auf Seite 181

Konfigurationsinformationen zu den von Net.Data bereitgestellten Sprachumgebungen finden Sie in „Definieren der Net.Data-Sprachumgebungen" auf Seite 32.

Informationen zur Leistungssteigerung bei Verwendung der Sprachumgebungen finden Sie in „Optimieren der Sprachumgebungen" auf Seite 233.

Übersicht über die von Net.Data bereitgestellten Sprachumgebungen

Net.Data stellt Sprachumgebungen bereit, mit denen Sie auf Daten und Programmierungsressourcen für Ihre Anwendung zugreifen können.

Net.Data stellt zwei Arten von Sprachumgebungen bereit:

- „Sprachumgebungen für relationale Datenbanken" auf Seite 155
- „Sprachumgebungen für Programmiersprachen" auf Seite 181

Tabelle 7 liefert eine Kurzbeschreibung jeder Sprachumgebung. Informationen dazu, welche Sprachumgebungen auf welchen Betriebssystemen unterstützt werden, finden Sie im Anhang zu Betriebssystemen des Handbuchs *Net.Data Reference*.

Tabelle 7. Net.Data-Sprachumgebungen

Sprach-umgebung	Umgebungsname	Beschreibung
IMS-Web	HWS_LE	In der IMS-Web-Sprachumgebung können Sie eine IMS-Transaktion über IMS-Web übergeben und die Ausgabe der Transaktion über Ihren Web-Browser empfangen.
Java-Anwendung	DTW_JAVAPPS	Net.Data unterstützt Ihre vorhandenen Java-Anwendungen in der Java-Sprachumgebung.
ODBC	DTW_ODBC	Die ODBC-Sprachumgebung führt SQL-Anweisungen über eine ODBC-Schnittstelle zum Zugriff auf mehrere Datenbankverwaltungs-systeme aus. Die Ergebnisse der ODBC-Anweisung können in einer Tabellenvariablen zurückgegeben werden.
Oracle	DTW_ORA	In der Oracle-Sprachumgebung können Sie direkt auf Ihre Oracle-Daten zugreifen.
Perl	DTW_PERL	Die Perl-Sprachumgebung interpretiert interne Perl-Scripts, die in einem FUNCTION-Block des Net.Data-Makros angegeben sind, oder sie führt externe Perl-Scripts aus, die in separaten Dateien gespeichert sind.
REXX	DTW_REXX	Die REXX-Sprachumgebung interpretiert interne REXX-Programme, die in einem FUNCTION-Block des Net.Data-Makros angegeben sind, oder sie führt externe REXX-Programme aus, die in einer separaten Datei gespeichert sind.
SQL	DTW_SQL	Die SQL-Sprachumgebung führt SQL-Anweisungen über DB2 aus. Die Ergebnisse der SQL-Anweisung können in einer Tabellen-variablen zurückgegeben werden. Die Ergebnisse der ODBC-Anweisung können in einer Tabellen-variablen zurückgegeben werden.
System	DTW_SYSTEM	Die SYSTEM-Sprachumgebung unterstützt das Ausführen von Befehlen und das Aufrufen externer Programme.
Webregistrie-rungsdaten-bank	DTW_WEBREG	Die Sprachumgebung für Webregistrierungs-datenbanken stellt Funktionen zur permanenten Speicherung von anwendungsbezogenen Daten bereit.

Aufrufen einer Sprachumgebung

Gehen Sie wie folgt vor, um eine Sprachumgebung aufzurufen:

- Definieren Sie mit einer Anweisung FUNCTION eine Funktion, die die Sprachumgebung aufruft, indem sie Sprachanweisungen oder eine %EXEC-Anweisung bereitstellt.
- Verwenden Sie einen Funktionsaufruf an die Sprachumgebung.

Beispiel:

```
%FUNCTION(DTW_SQL) custinfo() {
 select CUSTNAME, CUSTNO from ibmuser.customer
 %}
 ...
%HTML(REPORT){
 @custinfo()
 %}
```

Richtlinien zum Behandeln von Fehlerbedingungen

Wenn in einer Sprachumgebungsfunktion ein Fehler festgestellt wird, legt die Sprachumgebung die Net.Data-Variable RETURN_CODE mit einem Fehlercode fest.

Sie können Fehlerbedingungen mit Hilfe der folgenden Ressourcen behandeln:

- Die von Net.Data bereitgestellten Sprachumgebungen geben Fehlercodes zurück, die im Handbuch *Net.Data Nachrichten und Codes* dokumentiert sind.
- Die Datenbanksprachumgebungen, wie z. B. die SQL-Sprachumgebung, weisen der Variablen RETURN_CODE den SQLCODE-Wert und der Variablen SQL_STATE den SQLSTATE-Wert zu, der von der Datenbank zurückgegeben worden ist. Weitere Informationen zu den von Ihrem Datenbankverwaltungssystem verwendeten SQLCODE- und SQLSTATE-Werten finden Sie im Handbuch zu Fehlernachrichten Ihres Datenbankverwaltungssystems.

Sicherheit

Stellen Sie sicher, dass die Benutzer-ID, unter der Net.Data ausgeführt wird, über die entsprechende Berechtigung zum Zugreifen auf ein Objekt verfügt, auf das eventuell von einer Sprachumgebungsanweisung verwiesen wird. Die SQL-Sprachumgebung führt z. B. SQL-Anweisungen aus, weshalb die Benutzer-ID, unter der Net.Data ausgeführt wird, die Berechtigung haben muss, auf die Datenbankressourcen zuzugreifen, damit die Anweisung erfolgreich ausgeführt werden kann.

Sprachumgebungen für relationale Datenbanken

Net.Data stellt Sprachumgebungen für relationale Datenbanken bereit, um Ihnen den Zugriff auf Ihre relationalen Datenquellen zu erleichtern. Die SQL-Anweisungen, die Sie für den Zugriff auf die relationalen Daten bereitstellen, werden als dynamisches SQL ausgeführt. Weitere Informationen zu dynamischem SQL finden Sie in Ihrer Datenbankdokumentation.

In den folgenden Abschnitten werden die Sprachumgebungen und ihre Verwendung beschrieben:

ODBC-Sprachumgebung

Die ODBC-Sprachumgebung (ODBC - Open Database Connectivity) führt SQL-Anweisungen über eine ODBC-Schnittstelle aus. ODBC basiert auf der X/Open SQL CAE-Spezifikation, mit der eine einzelne Anwendung auf viele Datenbankverwaltungssysteme zugreifen kann.

Gehen Sie wie folgt vor, um die ODBC-Sprachumgebung zu verwenden:

Damit Sie die ODBC-Sprachumgebung verwenden können, müssen Sie zuerst einen ODBC-Treiber und einen Treibermanager erwerben und installieren. Informationen zur Installation und Konfiguration der ODBC-Umgebung finden Sie in Ihrer ODBC-Treiberdokumentation.

Prüfen Sie, ob sich eine Konfigurationsanweisung wie die folgende in der Net.Data-Initialisierungsdatei befindet und in einer Zeile steht.

Anmerkung: Der Pfad in dem folgenden Beispiel kann je nach Ihrem Betriebssystem abweichen.

```
ENVIRONMENT
(DTW_ODBC) d:/net.data/lib/dtwodbc.dll ( IN DATABASE, LOGIN, PASSWORD,
  TRANSACTION_SCOPE, START_ROW_NUM, DTW_SET_TOTAL_ROWS)
```

Einschränkungen:

- Die ODBC-Sprachumgebung unterstützt gespeicherte Prozeduren nur beim Herstellen der Verbindung zu DB2.
- Bei Angabe der Variablen DATABASE müssen Sie die gleiche Datenbank angeben wie die Datenquelle in der ODBC-Initialisierungsdatei.
- SQL-Anweisungen im Inline-Anweisungsblock können bis zu 64 KB groß sein. Bei DB2 Universal Database gibt es die folgenden Einschränkungen:
  - Version 6 oder höher: 64 KB
  - Version 5 Release 2 oder niedriger: 32 KB

  Ihre Datenbank hat eventuell andere Begrenzungen. Ermitteln Sie anhand der Dokumentation für Ihre Datenbank, ob Ihre Datenbank eine andere Begrenzung hat.

Oracle-Sprachumgebung

Die Oracle-Sprachumgebung bietet Basiszugriff auf Ihre Oracle-Daten. Sie können von Net.Data über CGI, FastCGI, NSAPI, ISAPI oder APAPI auf Oracle-Datenbanken zugreifen. Diese Sprachumgebung unterstützt Oracle Version 8.1.5.

Stellen Sie sicher, dass die folgende Konfigurationsanweisung in der Initialisierungsdatei enthalten ist und in einer Zeile steht, damit Sie die Oracle-Sprachumgebung verwenden können.

Anmerkung: Der Pfad in dieser Konfigurationsanweisung kann je nach Ihrem Betriebssystem oder Ihrer Installation abweichen.

```
ENVIRONMENT (DTW_ORA) /net.data/lib/dtwora.so (IN DATABASE, LOGIN, PASSWORD,
  TRANSACTION_SCOPE, START_ROW_NUM, DTW_SET_TOTAL_ROWS)
```

Informationen zum weiteren Definieren der Oracle-Sprachumgebung finden Sie in „Definieren der Oracle-Sprachumgebung” auf Seite 33.

Einschränkungen:

- Die Variable DATABASE wird nicht zum Zugriff auf Oracle-Datenbanken verwendet.
- Die Variable LOGIN muss den Oracle-Datenbankexemplarnamen enthalten. Zum Beispiel ist *ora73* der definierte Exemplarname in der folgenden Variablen LOGIN:

  ```
  LOGIN=admin@ora73
  ```

- Sie müssen bei Verwendung einer anderen Schnittstelle als CGI die Direktverbindung verwenden.
- Ein langer Datentyp wird wie eine normale Zeichenfolge gebündelt und darf ***nicht*** größer als 32 KB sein.

- Net.Data unterstützt keine gespeicherten Prozeduren in Oracle, die Ergebnismengen zurückgeben.

SQL-Sprachumgebung

Die SQL-Sprachumgebung bietet Zugriff auf DB2-Datenbanken. Verwenden Sie diese Sprachumgebung für optimale Leistung beim Zugreifen auf DB2.

Stellen Sie sicher, dass die folgende Konfigurationsanweisung in der Initialisierungsdatei enthalten ist und in einer Zeile steht, damit Sie die SQL-Sprachumgebung verwenden können.

Anmerkung: Der Pfad in dieser Konfigurationsanweisung kann je nach Ihrem Betriebssystem oder Ihrer Installation abweichen.

```
ENVIRONMENT (DTW_SQL) d:/net.data/lib/dtwsql.dll (IN DATABASE, LOGIN, PASSWORD,
  TRANSACTION_SCOPE, START_ROW_NUM, DTW_SET_TOTAL_ROWS)
```

Verschachtelte SQL-Anweisungen

Sie können SQL-Funktionen innerhalb einer anderen SQL-Funktion aufrufen. Stellen Sie beim Übergeben von Tabellen sicher, dass Sie in jeder Funktion eindeutige Tabellennamen verwenden; andernfalls können unvorhersehbare Ergebnisse auftreten.

Beispiel: Aufrufen einer SQL-Funktion vom ROW-Block einer anderen SQL-Funktion aus

```
%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) %}
```

Einschränkungen:

- Verschachtelte SQL-Anweisungen werden von Linux S/390 nicht unterstützt.

- SQL-Anweisungen im Inline-Anweisungsblock können bis zu 64 KB groß sein. Bei DB2 Universal Database gibt es die folgenden Einschränkungen:
- Version 6 oder höher: 64 KB
- Version 5 Release 2 oder niedriger: 32 KB

  Ihre Datenbank hat eventuell andere Begrenzungen. Ermitteln Sie anhand der Dokumentation für Ihre Datenbank, ob Ihr DBMS eine andere Begrenzung hat.
- Beim Verschachteln von SQL-Anweisungen können zu jeder Zeit maximal 32 Ergebnismengen verarbeitet werden. Sie können z. B. drei Ebenen verschachteln, von denen jede 10 Ergebnismengen zurückgibt. Oder verschachteln Sie 32 Ebenen, wobei jede Ebene eine Ergebnismenge zurückgibt.

Verwenden von DB2-Parametermarken

Wenn sie richtig verwendet werden, können Parametermarken die Leistung Ihrer Abfragen verbessern, indem sie DB2 anweisen, den Cache zu verwenden. Eine Parametermarke ist ein Fragezeichen (?) in einer SQL-Anweisung und gibt eine Position an, an der ein von einer Anwendung angegebener Wert ersetzt wird, wenn die Anweisung ausgeführt wird. Der Wert wird aus einer Net.Data-Variable in der Parameterliste der Net.Data-SQL-Funktionsdefinition abgerufen. Die Art und Weise, wie diese Werte abgerufen werden, hängt davon ab, wie Sie Parametermarken verwenden.

Sie können Parametermarken auf die folgenden zwei Arten verwenden:

- Explizit
- Implizit

Explizite Verwendung von Parametermarken:

Wenn Sie eine SQL-Anweisung erstellen, können Sie Ihrer Abfrage wie folgt manuelle Parametermarken hinzufügen:

Beispiel:

```
%FUNCTION (DTW_SQL) select_staff(in id, in dept){
     select * from staff
     where id = ? and dept = ?
     and salary = 35,000%}
```

Für jede Parametermarke gibt es in der Funktion DTW_SQL einen entsprechenden Parameter IN. Die Zuordnungsreihenfolge sowohl für die SQL- als auch für die Funktionsparameterliste ist von links nach rechts. Funktionsparameter, die ***keiner*** SQL-Parametermarke zugeordnet sind, können an das Ende der Funktionsparameterliste gestellt werden.

Implizite Verwendung von Parametermarken:

Die implizite Verwendung von Parametermarken wird aktiviert, indem die Markierung `DTW_USE_DB2_PREPARE_CACHE = YES` in der Initialisierungsdatei oder im Makro angegeben wird. Wenn die Konfigurationsvariable DTW_USE_DB2_PREPARE_CACHE auf YES gesetzt ist, ersetzt Net.Data jede Variable in der SQL-Anweisung durch eine Parametermarke. Die Daten werden an jede Parametermarke gebunden und werden nicht von der Net.Data-Parameterliste übergeben (wie das bei der expliziten Verwendung von Parametermarken der Fall ist).

Beispiel:

```
%FUNCTION (DTW_SQL) select_staff() {
     select * from staff
     where id = $(ID) and dept = $(dept)
     and salary = 35,000%}
```

Einschränkungen:

- Parametermarken sind nur für DB2 verfügbar.
- Stellen Sie sicher, dass die Konfigurationsvariablenmarkierung DTW_USE_DB2_PREPARE_CACHE in der Initialisierungsdatei oder im Makro auf NO gesetzt ist, um explizite Parametermarken zu verwenden.
- Alle Net.Data-Variablen in der SQL-Anweisung müssen sich an der Stelle einer Marke befinden, um implizite Parametermarken zu verwenden. Wenn dies nicht der Fall ist, wird ein Syntaxfehler gemeldet, weil Net.Data nicht bestimmen kann, welche Variable eine zulässige Marke ist.

Verwalten von Transaktionen in einer Net.Data-Anwendung

Wenn Sie den Inhalt einer Datenbank mit INSERT-, DELETE- oder UPDATE-Anweisungen ändern, werden diese Änderungen erst festgeschrieben, nachdem die Datenbank eine COMMIT-Anweisung von Net.Data empfangen hat. Wenn ein Fehler auftritt, sendet Net.Data eine ROLLBACK-Anweisung an die Datenbank, die alle Änderungen seit der letzten COMMIT-Operation zurücknimmt.

Wie Net.Data die COMMIT-Anweisung und die etwaige ROLLBACK-Anweisung sendet, hängt davon ab, wie Sie TRANSACTION_SCOPE festlegen und ob COMMIT-Anweisungen explizit im Makro angegeben sind. Zulässige Werte für TRANSACTION_SCOPE sind MULTIPLE und SINGLE. Der Standardwert ist MULTIPLE. Soll SINGLE für TRANSACTION_SCOPE angegeben werden, verwenden Sie eine Anweisung %DEFINE oder einen @DTW_ASSIGN()-Aufruf und übergeben die Variable in der Anweisung ENVIRONMENT für die entsprechende Sprachumgebung. Weitere Informationen finden Sie im Abschnitt zur Anpassung der Net.Data-Initialisierungsdatei in Kapitel 2 dieses Handbuchs.

SINGLE

Gibt an, dass Net.Data nach jeder erfolgreichen SQL-Anweisung eine COMMIT-Anweisung absetzt. Wenn die SQL-Anweisung einen Fehler zurückgibt, wird eine ROLLBACK-Anweisung abgesetzt. Die Angabe SINGLE für TRANSACTION_SCOPE stellt eine sofortige Datenbankänderung sicher. Allerdings kann eine Änderung später mit Hilfe einer ROLLBACK-Anweisung nicht widerrufen werden.

MULTIPLE

Gibt an, dass Net.Data alle SQL-Anweisungen vor dem Absetzen einer COMMIT-Anweisung ausführt. Net.Data sendet die COMMIT-Anweisung am Ende der Anforderung, und wenn jede SQL-Anweisung erfolgreich abgesetzt wird, schreibt die COMMIT-Anweisung alle Änderungen in der Datenbank fest. Wenn durch eine der Anweisungen ein Fehler zurückgegeben wird, setzt Net.Data eine ROLLBACK-Anweisung am Fehlerpunkt ab, die die Datenbank auf ihren vorherigen Status zurücksetzt.

Wenn Sie als Anwendungsentwickler die Standardeinstellung MULTIPLE für TRANSACTION_SCOPE beibehalten und wenn Sie COMMIT-Anweisungen am Ende der Anweisungsgruppen absetzen, die als Transaktion in Frage kommen, haben Sie volle Kontrolle über das COMMIT- und ROLLBACK-Verhalten in Ihrer Anwendung. Zum Beispiel kann das Absetzen von COMMIT-Anweisungen nach jeder Aktualisierung in Ihrem Makro zur Integritätsbewahrung Ihrer Daten beitragen.

Zum Absetzen einer SQL-Anweisung COMMIT können Sie eine Funktion definieren, die Sie an einem beliebigen Punkt in Ihrem HTML-Block aufrufen können:

```
%FUNCTION(DTW_SQL) user_commit() {
  commit
%}

...

%HTML {
  ...
  @user_commit()
  ...
%}
```

Einschränkungen:

Die Einstellung von TRANSACTION_SCOPE kann nicht mehr geändert werden, nachdem eine Verbindung zur Datenbank hergestellt wurde. Deshalb unterliegen alle SQL-Transaktionen in einem Makro derselben Verarbeitung.

Wenn Sie Net.Data als Teil von Net.Commerce verwenden, beachten Sie, dass Net.Commerce über eine eigene Transaktionsverarbeitung verfügt und die Transaktionsverarbeitung von Net.Data inaktiviert.

Verwenden großer Objekte

Sie können LOB-Dateien (Large Object) in DB2-Datenbanken speichern und mit der SQL- oder ODBC-Sprachumgebung von Net.Data in Ihre Webanwendungen einbinden.

Wenn die Sprachumgebung eine SQL-Anweisung SELECT oder eine gespeicherte Prozedur ausführt, die ein LOB zurückgibt, wird das Objekt weder einer Variablen zur Tabellenverarbeitung V(*n*) noch einem Net.Data-Tabellenfeld zugeordnet. Stattdessen wird das LOB in einer Datei gespeichert, die Net.Data erstellt, und nur der Name der Datei wird in der Variablen zur Tabellenverarbeitung V(*n*) oder in einem Net.Data-Tabellenfeld zurückgegeben. In Ihrem Net.Data-Makro können Sie den Namen verwenden, um auf die LOB-Datei zu verweisen; Sie können z. B. ein HTML-Ankerelement mit einem Hypertext-Verweis oder ein Bildelement, das eine URL für die Datei enthält, verwenden. Net.Data stellt die Datei mit dem LOB in das durch die Konfigurationsvariable HTML_PATH angegebene Verzeichnis. Diese Pfadanweisung befindet sich in der Net.Data-Initialisierungsdatei (db2www.ini). Der Schreibzugriff auf die LOB-Datei ist auf die Benutzer-ID beschränkt, die der Net.Data-Anforderung zugeordnet ist, mit der das LOB abgerufen wurde.

Der Dateiname für das LOB wird dynamisch erstellt und weist folgendes Format auf:

```
name[.extension]
```

Dabei gilt Folgendes:

name Dies ist eine dynamisch generierte eindeutige Zeichenfolge, die das große Objekt angibt.

extension
Dies ist eine Zeichenfolge, die den Objekttyp angibt. Bei CLOBs und DBCLOBs ist die Erweiterung .txt. Bei BLOBs ermittelt die SQL-Sprachumgebung die Erweiterung, indem sie in den ersten Bytes der LOB-Datei nach einer Kennung sucht. Tabelle 8 zeigt die LOB-Erweiterungen, die die SQL-Sprachumgebung verwendet:

Tabelle 8. In der SQL-Sprachumgebung verwendete LOB-Erweiterungen

Erweiterung	Objekttyp
.bmp	Bitmap-Image
.gif	Graphical Image Format
.jpg	JPEG-Image (Joint Photographic Experts Group)
.tif	Tagged Image File Format
.ps	PostScript

Tabelle 8. In der SQL-Sprachumgebung verwendete LOB-Erweiterungen (Forts.)

Erweiterung	Objekttyp
.mid	MIDI-Audio
.aif	AIFF-Audio
.avi	AVI
.au	Basisaudio
.ra	RA
.wav	WAV
.pdf	Portable Document Format
.rmi	MIDI-Sequenz

Wird der Objekttyp des binären großen Objekts (BLOB) nicht erkannt, wird dem Dateinamen keine Erweiterung hinzugefügt.

Wenn Net.Data den Namen der Datei zurückgibt, die ein LOB enthält, erhält der Dateiname das Präfix `/tmplobs/` unter Verwendung der folgenden Syntax:

`/tmplobs/`*`name`*`.[`*`extension`*`]`

Mit Hilfe dieses Präfix können Sie Ihr LOB-Verzeichnis in einem anderen Verzeichnis als dem Stammverzeichnis des Webservers anlegen.

Fügen Sie der Konfigurationsdatei Ihres Webservers die folgende Pass-Anweisung hinzu, um sicherzustellen, dass Verweise auf LOB-Dateien richtig aufgelöst werden:

```
Pass    /tmplobs/*     <full_path>/tmplobs/*
```

<full_path> ist der Wert, der für die Konfigurationsvariable HTML_PATH in der Net.Data-Initialisierungsdatei angegeben wurde.

Hinweis zur Planung: Jede Abfrage, die LOBs zurückgibt, führt zur Erstellung von Dateien in dem durch die Pfadkonfigurationsvariable HTML_PATH angegebenen Verzeichnis. Bei der Verwendung von LOBs ist die jeweilige Systemausstattung zu berücksichtigen, da diese Objekte die Systemressourcen schnell aufbrauchen können. Bereinigen Sie das Verzeichnis regelmäßig, oder führen Sie den Dämon `dtwclean` aus. Weitere Informationen hierzu finden Sie in „Verwalten temporärer LOBs" auf Seite 164. Es wird empfohlen, DataLinks zu verwenden. Dadurch braucht die SQL-Sprachumgebung Dateien nicht mehr in Verzeichnissen zu speichern, was zu einer Leistungsoptimierung und einer wesentlichen Reduzierung der belegten Systemressourcen führt.

Beispiel: Die folgende Anwendung verwendet eine MPEG-Audiodatei (.mpa). Da die SQL-Sprachumgebung diesen Dateityp nicht erkennt, wird eine EXEC-Variable dazu verwendet, die Erweiterung .mpa an den Dateinamen anzuhängen. Ein Benutzer dieser Anwendung muss den Dateinamen anklicken, um die Anzeigefunktion für MPEG-Audiodateien aufzurufen.

```
%DEFINE{
lobdir="/u/IBMUSER/tmplobs""
myFile=%EXEC "rename $(lobdir)$(filename) $(lobdir)$(filename).mpa"
%}
%{ where rename is the command on your operating system to rename files %}
%FUNCTION(DTW_SQL) queryData() {
SELECT Name, IDPhoto, Voice FROM RepProfile
   %REPORT{
    <p>Here is the information you selected:</p>
   %ROW{
      @DTW_ASSIGN(filename, @DTW_rSUBSTR(V3, @DTW_rLASTPOS("/", V3)))
$(myFile)
      $(V1) <img src="$(V2)">
            <a href="$(V3).mpa">Voice sample</a><p>
    %}
  %}
%}

%HTML (Report){
@queryData()
%}
```

Wenn die Tabelle RepProfile Informationen zu Kinson Yamamoto und Merilee Lau enthält, wird der Webseite, die generiert wird, bei der Ausführung des REPORT-Blocks der folgende HTML-Text hinzugefügt:

```
<p>Here is the information you selected:</p>
Kinson Yamamoto <img src="/tmplobs/p2345n1.gif">
<a href="/tmplobs/p2345n2.mpa">Voice sample</a><p>
Merilee Lau <img src="/tmplobs/p2345n3.gif">
<a href="/tmplobs/p2345n4.mpa">Voice sample</a><p>
```

Der REPORT-Block im vorangegangenen Beispiel verwendet die impliziten Tabellenvariablen V1, V2 und V3.

- Der Wert von V1 ist der Name einer Person. Dabei handelt es sich um Zeichendaten.
- Der Wert von V2 ist der Name einer GIF-Datei, die das Foto der Person enthält. Das Bild wird inline in der generierten Webseite angezeigt.
- Der Wert von V3 ist der Name einer MPA-Datei mit dem Stimmmuster der Person. Da Net.Data das Dateiformat MPA nicht erkennt, fügt es dem Dateinamen keine Erweiterung hinzu, wenn es die Datei für das LOB im durch HTML_PATH angegebenen Verzeichnis erstellt. Dieses Beispiel veranschaulicht die Verwendung einer EXEC-Variablen zum Anfügen der Erweiterung .mpa an den Dateinamen. Das Stimmmuster wird wiedergegeben, wenn der Benutzer den Text "Voice sample" anklickt; dabei handelt es sich um einen Hyperlink-Text.

Zugriffsrechte für LOBs:

Das Standardverzeichnis `tmplobs` für LOBs ist unter dem Verzeichnis angeordnet, das durch HTML_PATH in der mitgelieferten Net.Data-Initialisierungsdatei angegeben ist. Eine beliebige Benutzer-ID kann darauf zugreifen. Wenn der Wert von HTML_PATH geändert wird, stellen Sie sicher, dass die Benutzer-ID, unter der der Webserver ausgeführt wird, Schreibzugriff für das durch HTML_PATH angegebene Verzeichnis hat. (Weitere Informationen finden Sie in „HTML_PATH" auf Seite 18.) **Verwalten temporärer LOBs:**

Net.Data speichert temporäre LOBs im Unterverzeichnis `tmplobs` des Verzeichnisses, das in der Pfadkonfigurationsvariablen HTML_PATH angegeben ist. Diese Dateien können groß sein und sollten regelmäßig bereinigt werden, um eine Leistungseinbuße zu vermeiden.

Net.Data stellt den Dämon `dtwclean` zum periodischen Verwalten des Verzeichnisses `tmplobs` bereit. `dtwclean` verwendet den Anschluss 7127.

Gehen Sie wie folgt vor, um den Dämon `dtwclean` auszuführen: Geben Sie den folgenden Befehl im Befehlszeilenfenster ein:

```
dtwclean [-t xx] [-d|-l]
```

Dabei gilt Folgendes:

-t Diese Markierung gibt das Intervall an, in dem `dtwclean` das Verzeichnis bereinigt.

xx Dieses Intervall gibt die Sekunden an, die eine Datei im Verzeichnis verbleibt, bevor `dtwclean` sie löscht. Für diesen Wert gibt es keine Begrenzung. Der Standardwert ist 3600 Sekunden.

-d Diese Markierung gibt den Debug-Modus an. Im Befehlsfenster werden Trace-Informationen angezeigt.

-l Diese Markierung gibt den Protokollierungsmodus an. Trace-Informationen werden in eine Protokolldatei ausgegeben.

Gespeicherte Prozeduren

Eine gespeicherte Prozedur ist ein kompiliertes Programm, das in einer Datenbank gespeichert ist und SQL-Anweisungen ausführen kann. In Net.Data werden gespeicherte Prozeduren von Net.Data-Funktionen mit Hilfe der Anweisung CALL aufgerufen. Parameter für gespeicherte Prozeduren werden aus der Parameterliste der Net.Data-Funktion übergeben. Sie können gespeicherte Prozeduren einsetzen, um die Leistung und die Integrität zu verbessern, indem Sie kompilierte SQL-Anweisungen auf dem Datenbankserver speichern. Net.Data unterstützt die Verwendung gespeicherter Prozeduren unter DB2 über die SQL- und ODBC-Sprachumgebungen. Gespeicherte Prozeduren von Oracle werden über die Oracle-Sprachumgebung unterstützt. Net.Data unterstützt insbesondere für DB2 gespeicherte Prozeduren, die mindestens eine Ergebnismenge zurückgeben.

In diesem Abschnitt werden folgende Themen behandelt:

- „Syntax für gespeicherte Prozeduren"
- „Aufrufen einer gespeicherten Prozedur" auf Seite 166
- „Übergeben von Parametern" auf Seite 168
- Nur für DB2: „Verarbeiten von Ergebnismengen aus gespeicherten DB2-Prozeduren" auf Seite 168

Syntax für gespeicherte Prozeduren

Die Syntax für gespeicherte Prozeduren enthält die Anweisung FUNCTION, die Anweisung CALL und optional einen REPORT-Block.

```
%FUNCTION (DTW_SQL) function_name ([IN datatype arg1, INOUT datatype arg2,
    OUT resultsetname, ...]) {
  CALL stored_procedure [(resultsetname, ...)]
[%REPORT [(resultsetname)] { %}]
...
[%REPORT [(resultsetname)] { %}]
[%MESSAGE %}]

%}
```

Dabei gilt Folgendes:

function_name
: Ist der Name der Net.Data-Funktion, die den Aufruf der gespeicherten Prozedur initialisiert.

stored_procedure
: Ist der Name der gespeicherten Prozedur.

datatype
: Ist einer der von Net.Data unterstützten Datentypen der Datenbank wie in Tabelle 9 auf Seite 166 und Tabelle 10 auf Seite 166 gezeigt. Die in der Parameterliste angegebenen Datentypen müssen mit den Datentypen in der gespeicherten Prozedur übereinstimmen. Weitere Informationen zu diesen Datentypen finden Sie in Ihrer Datenbankdokumentation.

Nur für DB2: *tablename*
: Ist der Name der Net.Data-Tabelle, in der die Ergebnismenge gespeichert werden soll (wird nur verwendet, wenn die Ergebnismenge in einer Net.Data-Tabelle gespeichert werden soll). Wenn dieser Parametername angegeben ist, muss er mit dem zugehörigen Parameternamen für *resultsetname* übereinstimmen.

Nur für DB2: *resultsetname*
: Ist der Name, der eine von einer gespeicherten Prozedur zurückgegebene Ergebnismenge einem REPORT-Block und/oder einem Tabellennamen in der Funktionsparameterliste zuordnet. Die Angabe *resultsetname* in einem REPORT-Block muss mit einer Ergebnismenge in der Anweisung CALL übereinstimmen.

Tabelle 9. Unterstützte Datentypen gespeicherter Prozeduren für DB2

BIGINT	DOUBLEPRECISION	SMALLINT
CHAR	FLOAT	TIME
CLOB[1]	INTEGER	TIMESTAMP
DATE	GRAPHIC	VARCHAR
DECIMAL	LONGVARCHAR	VARGRAPHIC
DOUBLE	LONGVARGRAPHIC	

[1] CLOB kann nur als Parameter für OUT und INOUT verwendet werden, und Net.Data interpretiert die Größe in Byte. Wenn Sie z. B. eine Variable OUT CLOB(20000) angeben, wird ein CLOB der Größe 20 K als Ausgabeparameter verwendet.

Tabelle 10. Unterstützte Datentypen gespeicherter Prozeduren für Oracle

BIGINT	LONG
CHAR	LONG RAW
DATE	NUMBER
DECIMAL	RAW
FLOAT	VARCHAR / VACHAR2
INTEGER	

Wichtig: Wenn Net.Data unter Windows oder Unix eine gespeicherte Prozedur in DB2 unter OS/390 und OS/400 aufruft, muss die gespeicherte Prozedur unter diesen Betriebssystemen den Host-Variablentyp DOUBLE oder FLOAT verwenden, wenn DECIMAL-Daten aus der DB2-Datenbank abgerufen werden. Durch Verwendung des Host-Variablentyps DOUBLE oder FLOAT wird sichergestellt, dass die zurückgegebenen Daten ein lesbares Format aufweisen.

Aufrufen einer gespeicherten Prozedur

1. Definieren Sie eine Funktion, die einen Aufruf an die gespeicherte Prozedur initialisiert.

   ```
   %FUNCTION (DTW_SQL) function_name()
   ```

2. (Optional) Geben Sie beliebige Parameter IN, INOUT oder OUT für die gespeicherte Prozedur an. Bei gespeicherten DB2-Prozeduren kann der Parameter einen Tabellenvariablennamen für die Speicherung einer Ergebnismenge in einer Net.Data-Tabelle enthalten (Sie müssen eine Net.Data-Tabelle nur angeben, wenn die Ergebnismenge in einer Net. Data-Tabelle gespeichert werden soll).

   ```
   %FUNCTION (DTW_SQL) function_name (IN datatype
   arg1, INOUT datatype arg2,
       OUT resultsetname...)
   ```

3. Geben Sie mit Hilfe der Anweisung CALL den Namen der gespeicherten Prozedur an.

   ```
   CALL stored_procedure
   ```

4. Für DB2: Wenn die gespeicherte DB2-Prozedur nur eine Ergebnismenge generiert, können Sie optional einen REPORT-Block angeben, um zu definieren, wie Net.Data die Ergebnismenge anzeigen soll.

   ```
   %REPORT [(resultsetname)] {
   ...
   %}
   ```

   Beispiel:

   ```
   %FUNCTION (DTW_SQL) mystoredproc (IN CHAR(30) arg1)  {
        CALL  myproc
    %REPORT (mytable){
     ...
     %ROW {  ...   %}
     ...
    %}
   %}
   ```

5. Wenn die gespeicherte Prozedur mehrere Ergebnismengen generiert, haben Sie folgende Möglichkeiten:
   - Geben Sie die Namen der Ergebnismengen in der Anweisung CALL an.

     ```
     CALL stored_procedure[ (resultsetname1[, resultsetname2, ...]) ]
     ```

   - Geben Sie optional mindestens einen REPORT-Block an, um zu definieren, wie Net.Data die Ergebnismengen anzeigen soll.

     ```
     %REPORT[(resultsetname1)] {
     ...
     %}
     ```

     Beispiel:

     ```
     %FUNCTION (DTW_SQL) mystoredproc (IN CHAR(30) arg1, OUT table1) {
        CALL myproc (table1, table2)
      %REPORT(table2) {
           ...
      %ROW {  ...   %}
           ...
      %}
     %}
     ```

6. Neben gespeicherten Prozeduren stellt Oracle außerdem gespeicherte Funktionen zur Verfügung. Zum Aufrufen einer gespeicherten Oracle-Funktion muss die Net.Data-Variable DTWORA_RESULT verwendet werden. Nach der Ausführung enthält DTWORA_RESULT den Rückgabewert der gespeicherten Funktion.

   Beispiel:

   ```
   %FUNCTION (DTW_ORA) orastp (IN datatype arg1, OUT datatype arg2,...)
      returns (DTWORA_RESULT) {
      CALL stored_oracle_function
      %}
   ```

Übergeben von Parametern

Sie können Parameter an eine gespeicherte Prozedur übergeben und die gespeicherte Prozedur die Parameterwerte aktualisieren lassen, so dass die neuen Werte an das Net.Data-Makro zurückgegeben werden. Die Anzahl und der Typ der Parameter in der Funktionsparameterliste müssen mit der Anzahl und dem Typ übereinstimmen, die für die gespeicherte Prozedur definiert sind. Wenn z. B. ein Parameter in der für die gespeicherte Prozedur definierten Parameterliste INOUT ist, dann muss der entsprechende Parameter in der Funktionsparameterliste auch INOUT sein. Wenn ein Parameter in der für die gespeicherte Prozedur definierten Parameterliste vom Typ CHAR(30) ist, dann muss der entsprechende Parameter in der Funktionsparameterliste auch CHAR(30) sein.

Beispiel 1: Übergeben eines Parameterwerts an die gespeicherte Prozedur

```
%FUNCTION (DTW_SQL) mystoredproc (IN CHAR(30) valuein)  {
     CALL  myproc
...
```

Beispiel 2: Zurückgeben eines Werts aus einer gespeicherten Prozedur

```
%FUNCTION (DTW_SQL) mystoredproc (OUT VARCHAR(9) retvalue)  {
     CALL  myproc
...
```

Verarbeiten von Ergebnismengen aus gespeicherten DB2-Prozeduren

Sie können mindestens eine Ergebnismenge aus einer gespeicherten Prozedur zurückgeben, wenn Sie die SQL- oder ODBC-Sprachumgebung verwenden. Die Ergebnismengen können in Net.Data-Tabellen zur weiteren Verarbeitung innerhalb Ihres Makros gespeichert oder mit Hilfe eines REPORT-Blocks verarbeitet werden. Wenn eine gespeicherte Prozedur mehrere Ergebnismengen generiert, müssen Sie jeder durch die gespeicherte Prozedur generierten Ergebnismenge einen Namen zuordnen. Dies geschieht durch die Angabe von Parametern in der Anweisung CALL. Der Name, den Sie für eine Ergebnismenge angeben, kann anschließend einem REPORT-Block oder einer Net.Data-Tabelle zugeordnet werden, so dass Sie festlegen können, wie die einzelnen Ergebnismengen von Net.Data verarbeitet werden. Sie haben folgende Möglichkeiten:

- Sie können das Ergebnis in der Standarddarstellung für Net.Data-Berichte verarbeiten, indem Sie keinen REPORT-Block für die Ergebnismenge definieren.
- Sie können eine Ergebnismenge einem REPORT-Block zuordnen, um Ihre eigene Berichtsdarstellung anzuwenden. Sie können im REPORT-Block mit Net.Data-Variablen, Net.Data-Textverarbeitungsanweisungen wie HTML oder JavaScript oder anderen Funktionen angeben, wie die Berichtsdaten vom Browser angezeigt werden sollen.
- Sie können die Ergebnismengen in Net.Data-Tabellen speichern, wenn Sie wollen, dass Net.Data die Daten später im Makro verwendet.

Zum Beispiel können Sie die Net.Data-Tabelle an eine andere Funktion übergeben, die die Daten zu Berechnungen und zum Anzeigen der berechneten Ergebnisse verwenden kann.

Richtlinien und Einschränkungen zur Verwendung mehrerer REPORT-Blöcke finden Sie in „Richtlinien und Einschränkungen für mehrere REPORT-Blöcke" auf Seite 144.

Gehen Sie wie folgt vor, wenn nur eine Ergebnismenge zurückgegeben und diese in einem Standardbericht angezeigt werden soll:

Verwenden Sie die folgende Syntax:

```
%FUNCTION (DTW_SQL) function_name () {
     CALL stored_procedure
%}
```

Beispiel:

```
 %FUNCTION (DTW_SQL) mystoredproc() {
     CALL  myproc
%}
```

Gehen Sie wie folgt vor, wenn nur eine Ergebnismenge zurückgegeben und ein REPORT-Block angegeben werden soll:

Verwenden Sie die folgende Syntax:

```
%FUNCTION (DTW_SQL) function_name () {
     CALL stored_procedure  [(resultsetname)]
 %REPORT [(resultsetname)] {
  ...
 %}
%}
```

Beispiel 1:

```
%FUNCTION (DTW_SQL) mystoredproc () {     CALL  myproc
%REPORT {
      ...
  %ROW {  ...   %}
      ...
 %}
%}
```

Beispiel 2:

```
%FUNCTION (DTW_SQL) mystoredproc () {
    CALL myproc (mytable1)
 %REPORT (mytable1) {
  ...
```

```
  %ROW {  ...   %}
  ...
 %}
%}
```

Gehen Sie wie folgt vor, wenn nur eine Ergebnismenge in einer Net.Data-Tabelle zur weiteren Verarbeitung gespeichert werden soll:

Verwenden Sie die folgende Syntax:

```
%FUNCTION (DTW_SQL) function_name (OUT tablename) {
     CALL stored_procedure  [(resultsetname)]
%}
```

Beispiel:

```
%DEFINE DTW_DEFAULT_REPORT = "NO"

%FUNCTION (DTW_SQL) mystoredproc (OUT mytable1) {
     CALL  myproc
%}
```

Beachten Sie, dass die Variable DTW_DEFAULT_REPORT auf den Wert NO gesetzt ist, so dass kein Standardbericht für die Ergebnismenge generiert wird.

Gehen Sie wie folgt vor, wenn mehrere Ergebnismengen zurückgegeben und diese im Standardberichtsformat angezeigt werden sollen:

Verwenden Sie die folgende Syntax:

```
%FUNCTION (DTW_SQL) function_name () {
    CALL stored_procedure [(resultsetname1, resultsetname2, ...)]
%}
```

Dabei wird kein REPORT-Block angegeben.

Beispiel:

```
%DEFINE DTW_DEFAULT_REPORT = "YES"
%FUNCTION (DTW_SQL) mystoredproc () {
     CALL  myproc
%}
```

Gehen Sie wie folgt vor, wenn mehrere Ergebnismengen zurückgegeben und die Ergebnismengen in Net.Data-Tabellen zur weiteren Verarbeitung gespeichert werden sollen:

Verwenden Sie die folgende Syntax:

```
 %FUNCTION (DTW_SQL) function_name (OUT (resultsetname1, resultsetname2, ...) {
    CALL stored_procedure (resultsetname1, resultsetname2, ...)
%}
```

Beispiel:

```
%DEFINE DTW_DEFAULT_REPORT = "NO"

%FUNCTION (DTW_SQL) mystoredproc (OUT mytable1, mytable2) {
    CALL myproc (mytable1, mytable2)
%}
```

Beachten Sie, dass die Variable DTW_DEFAULT_REPORT auf den Wert NO gesetzt ist, so dass kein Standardbericht für die Ergebnismengen generiert wird.

Gehen Sie wie folgt vor, wenn mehrere Ergebnismengen zurückgegeben und REPORT-Blöcke für die Verarbeitung der Anzeige angegeben werden sollen:

Jede Ergebnismenge ist einem REPORT-Block oder mehreren REPORT-Blöcken zugeordnet. Verwenden Sie die folgende Syntax:

```
%FUNCTION (DTW_SQL) function_name (, ...) {
   CALL stored_procedure (resultsetname1, resultsetname2, ...)
 %REPORT (resultsetname1)
   ...
  %ROW {  ...   %}
   ...
 %}
 %REPORT (resultsetname2)
   ...
  %ROW {  ...   %}
   ...
 %}

...
%}
```

Beispiel:

```
%FUNCTION (DTW_SQL) mystoredproc () {
   CALL myproc (mytable1, mytable2)

 %REPORT (mytable1) {
  ...
  %ROW {  ...   %}
  ...
 %}

 %REPORT(mytable2) {
  ...
  %ROW {  ...   %}
  ...
 %}
%}
```

Gehen Sie wie folgt vor, wenn mehrere Ergebnismengen zurückgegeben und für jede Ergebnismenge verschiedene Anzeige- oder Verarbeitungsoptionen angegeben werden sollen:

Sie können für jede Ergebnismenge andere Verarbeitungsoptionen angeben, indem Sie eindeutige Parameternamen verwenden. **Beispiel 1:**

```
%FUNCTION (DTW_SQL) mystoredproc (OUT mytable2) {
    CALL myproc (mytable1, mytable2, mytable3)

 %REPORT (mytable1) {
 ...
  %ROW {  ...   %}
 ...
 %}
%}
```

Die Ergebnismenge `mytable1` wird vom entsprechenden REPORT-Block verarbeitet und wie vom Makroautor angegeben angezeigt. Die Ergebnismenge `mytable2` wird in der Net.Data-Tabelle `mytable2` gespeichert und kann nun zur weiteren Verarbeitung, zum Beispiel zur Übergabe an eine andere Funktion, verwendet werden. Die Ergebnismenge `mytable3` wird im Standardberichtsformat von Net.Data angezeigt, weil für sie kein REPORT-Block angegeben wurde.

Beispiel 2:

```
%FUNCTION(DTW_SQL) mystoredproc(OUT mytable4, OUT mytable3) {
    CALL myproc (mytable1, mytable2, mytable3, mytable4)
 %REPORT(mytable2) {
    ...
  %ROW {  ...   %}
    ...
    %}
 %REPORT (mytable1) {
    ...
  %ROW {  ...   %}
    ...
    %}
    %REPORT(mytable4) {
    ...
  %ROW {  ...   %}
    ...
    %}
 %}
```

Die Ergebnismengen *mytable2*, *mytable1* und *mytable4* werden durch den entsprechenden REPORT-Block in dieser Reihenfolge verarbeitet und wie angegeben angezeigt. Die Ergebnismengen *mytable4* und *mytable3* werden zur weiteren Verarbeitung in Tabellenvariablen gespeichert. Die Ergebnismenge *mytable3* wird auch mit Hilfe des Net.Data-Standardberichtsformats angezeigt, sobald die Verarbeitung der drei REPORT-Blöcke abgeschlossen ist.

Codieren von DataLink-URL-Adressen in Ergebnismengen

Der Datentyp DATALINK ist einer der Grundbausteine für die Erweiterung der Datentypen, die in Datenbankdateien gespeichert werden können. Bei DATALINK sind die in der Spalte gespeicherten Daten lediglich ein Zeiger zur Datei. Diese Datei kann in einem beliebigen Dateityp vorliegen, z. B. eine Abbilddatei, eine Stimmaufzeichnung oder eine Textdatei. DataLink-Datentypen speichern eine URL-Adresse, um die Speicherposition der Datei aufzulösen.

Für den Datentyp DATALINK ist die Verwendung von DataLink File Manager erforderlich. Weitere Informationen zu DataLink File Manager finden Sie in der DataLinks-Dokumentation für Ihr Betriebssystem. Vor der Verwendung des Datentyps DATALINK müssen Sie sicherstellen, dass der Webserver Zugriff auf das durch den Server mit DB2 File Manager verwaltete Dateisystem hat.

Wenn eine SQL-Abfrage eine Ergebnismenge mit DataLinks zurückgibt und die DataLink-Spalte mit den DataLink-Optionen FILE LINK CONTROL und READ PERMISSION DB erstellt wird, enthalten die Dateipfade in der DataLink-Spalte ein Zugriffs-Token. DB2 überprüft mit dem Zugriffs-Token den Zugriff auf die Datei. Ohne dieses Zugriffs-Token schlagen alle Zugriffsversuche auf die Datei mit einer Berechtigungsverletzung fehl. Das Zugriffs-Token enthält jedoch unter Umständen Zeichen, die in einer URL-Adresse (die an einen Browser zurückgegeben werden soll) nicht verwendet werden können, z. B. das Semikolon (;). Beispiel:

```
/datalink/pics/UN1B;0YPVKGG346KEBE;baibien.jpg
```

Die URL-Adresse ist ungültig, weil sie Semikolons (;) enthält. Die Semikolons müssen mit der integrierten Net.Data-Funktion DTW_URLESCSEQ codiert werden, um die Ungültigkeit der URL-Adresse aufzuheben. Vor der Anwendung dieser Funktion muss die Zeichenfolge allerdings noch bearbeitet werden, weil diese Funktion auch Schrägstriche (/) codiert.

Sie können eine Net.Data-Makrofunktion (MACRO_FUNCTION) schreiben, um die Bearbeitung der Zeichenfolge zu automatisieren und die Funktion DTW_URLESCSEQ zu verwenden. Verwenden Sie dieses Verfahren in jedem Makro, das Daten aus einer Spalte mit dem Datentyp DATALINK abruft.

Beispiel 1: MACRO_FUNCTION zur Automatisierung der Codierung von URL-Adressen, die von DB2 UDB zurückgegeben werden

```
%{ 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" ) ) ) )
%}
```

Nach der Verwendung dieser Makrofunktion ist die URL-Adresse ordnungsgemäß codiert, und auf die in der Spalte DATALINK angegebene Datei kann in einem beliebigen Web-Browser verwiesen werden.

Beispiel 2: Ein Net.Data-Makro zur Angabe der SQL-Abfrage, die die DATALINK-URL-Adresse zurückgibt

```
%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>
   %}
 %}
%}
```

Beachten Sie, dass eine Funktion von DataLink File Manager verwendet wird. Die Funktion `dlurlcomplete` gibt eine vollständige URL-Adresse zurück.

Beispiele für die Sprachumgebung für relationale Datenbanken

Die folgenden Beispiele zeigen, wie Sie die Sprachumgebung für relationalen Datenbanken von Ihren Makros aus aufrufen können:

ODBC

Das folgende Beispiel definiert und ruft Mehrfachfunktionen für die ODBC-Sprachumgebung auf.

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

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

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

%function(dtw_odbc) sql3() {
insert into $(table) (int2) values (222)
```

```
%}

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

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

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

Oracle Das folgende Beispiel zeigt ein Makro mit einer Funktionsdefinition DTW_ORA, die die Oracle-Datenbank udatabase abfragt. Dabei wird mit einem Variablenverweis die abzufragende Datenbanktabelle ermittelt. Der FUNCTION-Block enthält auch einen MESSAGE-Block, der Fehlerbedingungen behandelt. Net.Data zeigt nach der Verarbeitung des Makros im Browser einen Standardbericht an.

```
%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

Das folgende Beispiel zeigt ein Makro mit einer DTW_SQL-Funktionsdefinition, die eine gespeicherte SQL-Prozedur aufruft. Darin sind drei Parameter mit unterschiedlichen Datentypen enthalten. Die Sprachumgebung DTW_SQL übergibt jeden Parameter in Übereinstimmung mit dem Datentyp des Parameters an die gespeicherte Prozedur. Wenn die Verarbeitung der gespeicherten Prozedur abgeschlossen ist, werden Ausgabeparameter zurückgegeben, und Net.Data aktualisiert die Variablen entsprechend.

```
%{************************************************************
                    DEFINE BLOCK
************************************************************%}
%DEFINE {
 MACRO_NAME     = "TEST ALL TYPES"
 DTW_HTML_TABLE = "YES"
 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 TESTTYPE
%}
%HTML(report){
<head>
<title>Net.Data : SQL Stored Procedure: Example '$(MACRO_NAME)'.
</title>
</head>
<body bgcolor="#bbffff" text="#000000" link="#000000">
<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:
</h2>
<b>parm1  (SMALLINT)<p></b>
$(parm1)<br />
<b>parm2  (INT)</b>
$(parm2)<br />
<b>parm3  (DECIMAL)</b>
$(parm3)
<hr/>
```

```
<h2>
Calling the function that executes the stored procedure.
</h2>
<p>
 @myProc(parm1,parm2,parm3)
</p><hr/>
<h2>
Values of the INOUT parameters after
calling the stored procedure:<p>
</h2>
<p><b>parm1  (SMALLINT)</b><br />
$(parm1)<br />
<b>parm2  (INT)</b>
$(parm2)<br />
<b>parm3  (DECIMAL)</b>
$(parm3)
</p></body>
%}
```

Sprachumgebung für Webregistrierungsdatenbanken

Die Net.Data-Webregistrierungsdatenbank bietet permanente Speicherung für anwendungsbezogene Daten. In einer Webregistrierungsdatenbank können Sie Konfigurationsdaten und andere Daten speichern, auf die Sie während der Ausführung durch webgestützte Anwendungen dynamisch zugreifen können. Sie können nur über Net.Data-Makros, die Net.Data und die integrierte Unterstützung für Webregistrierungsdatenbanken verwenden, und von für diesen Zweck geschriebenen CGI-Programmen aus auf Webregistrierungsdatenbanken zugreifen. Die Webregistrierungsdatenbank ist in einem Teil der Betriebssysteme verfügbar. Im Handbuch *Net.Data Reference* finden Sie eine Beschreibung und eine Syntaxdarstellung der integrierten Funktionen für Webregistrierungsdatenbanken sowie eine Liste der Betriebssysteme, die die Sprachumgebung unterstützen.

Bei der Entwicklung einer Webseite müssen URL-Adressen standardmäßig direkt in die HTML-Quelle für die Seite gestellt werden. Dies erschwert die Änderung von Verbindungen (Links). Durch die statische Beschaffenheit werden zudem die Verbindungsarten begrenzt, die einfach auf einer Webseite platziert werden können. Die Verwendung einer Webregistrierungsdatenbank zum Speichern von anwendungsbezogenen Daten, z. B. URL-Adressen, kann sich beim Erstellen von HTML-Seiten mit dynamisch festgelegten Verbindungen (Links) als nützlich herausstellen.

Daten können durch Anwendungsentwickler und Webadministratoren mit entsprechendem Schreibzugriff in einer Registrierungsdatenbank gespeichert und verwaltet werden. Anwendungen rufen die Daten während der Laufzeit aus den zugeordneten Registrierungsdatenbanken ab. Dies ermöglicht den Entwurf flexibler Anwendungen und Spielraum für Anwendungen und Server. Sie können mit Net.Data-Makros unter Verwendung dynamisch festgelegter Verbindungen (Links) HTML-Seiten erstellen.

Daten werden in einer Webregistrierungsdatenbank in Form von Registrierungsdatenbankeinträgen gespeichert. Jeder Registrierungsdatenbankeintrag besteht aus einem Paar von Zeichenfolgen: einer RegistryVariable-Zeichenfolge und einer entsprechenden RegistryData-Zeichenfolge. Alle Daten, die durch ein Zeichenfolgepaar dargestellt werden können, können als Registrierungsdatenbankeintrag gespeichert werden. Net.Data verwendet die Variablenzeichenfolge als Suchkriterium zum Lokalisieren und Abrufen spezifischer Einträge aus einer Registrierungsdatenbank.

Tabelle 11 zeigt eine Beispiel-Webregistrierungsdatenbank:

Tabelle 11. Beispiel-Webregistrierungsdatenbank

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

Gründe zur Verwendung einer Webregistrierungsdatenbank:

- In einer Webregistrierungsdatenbank können Sie Aliasnamen für Server und URL-Adressen speichern und damit die Verlagerung von Anwendungen und Servern erleichtern.
- Anwendungsentwickler können ihre webgestützten Anwendungen mit in der Registrierungsdatenbank vordefinierten Daten, wie URL-Adressen, ausliefern. Der Endbenutzer kann die Registrierungsdatenbankdaten ändern, um das Verhalten der Anwendung anzupassen.
- In einer Webregistrierungsdatenbank können Sie URL-Suchvorgänge basierend auf Produktname, Landessprache, Hersteller usw. ausführen.

Indexierte Einträge in der Webregistrierungsdatenbank sind Einträge, an deren RegistryVariable-Zeichenfolgen eine zusätzliche Indexzeichenfolge mit folgender Syntax angehängt ist:

```
RegistryVariable/Index
```

Der Benutzer stellt die Werte der Indexzeichenfolge in einem separaten Parameter für eine integrierte Funktion bereit, die für indexierte Einträge entworfen wurde. Mehrere indexierte Registrierungsdatenbankeinträge können den gleichen RegistryVariable-Zeichenfolgewert haben, durch unterschiedliche Indexzeichenfolgewerte kann ihre Eindeutigkeit jedoch bewahrt werden.

Tabelle 12. Indexierte Beispiel-Webregistrierungsdatenbank

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

Obwohl die beiden obigen indexierten Einträge den gleichen RegistryVariable-Zeichenfolgewert `Smith` haben, ist die Indexzeichenfolge in beiden Fällen unterschiedlich. Sie werden von den Funktionen für Webregistrierungsdatenbanken als zwei unterschiedliche Einträge behandelt.

Konfigurieren der Sprachumgebung für Webregistrierungsdatenbanken

Prüfen Sie, ob sich die folgende Konfigurationsanweisung in der Initialisierungsdatei befindet und in einer Zeile steht:

```
ENVIRONMENT (DTW_WEBREG)   DTWWEB    ( OUT RETURN_CODE )
```

Weitere Informationen zur Net.Data-Initialisierungsdatei sowie zu Anweisungen ENVIRONMENT für die Sprachumgebung finden Sie in „Umgebungskonfigurationsanweisungen" auf Seite 29.

Aufrufen der integrierten Funktionen für Webregistrierungsdatenbanken

Rufen Sie eine Funktion für Webregistrierungsdatenbanken so wie jede andere Funktion auf. Definieren Sie mit einer Anweisung DEFINE die zu übergebenden Parameter als Variablen. Beispiel:

```
%DEFINE {
    name = "smith"
 %}
```

Verwenden Sie dann eine Funktionsaufrufanweisung zum Aufrufen der Funktion, z. B.:

```
@DTWR_ADDENTRY("URLLIST", name, "http://www.ibm.com/software/",
    "WORK_URL"
```

Beispiel

Das folgende Beispiel erstellt eine Webregistrierungsdatenbank und fügt Einträge hinzu. Es zeigt anschließend einen Bericht mit den Einträgen an.

```
%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)

%}
```

Sprachumgebungen für Programmiersprachen

Net.Data stellt die folgenden Sprachumgebungen für das Aufrufen externer Programme bereit:

- „Java-Anwendungssprachumgebung"
- „Perl-Sprachumgebung" auf Seite 185
- „REXX-Sprachumgebung" auf Seite 189
- „SYSTEM-Sprachumgebung" auf Seite 196

Zugriffsrechte: Stellen Sie sicher, dass die Benutzer-ID, unter der Net.Data ausgeführt wird, Zugriffsrechte für die Programmausführung sowie für alle Objekte besitzt, auf die das Programm zugreift. Weitere Informationen hierzu finden Sie in „Erteilen von Zugriffsrechten für Dateien, auf die Net.Data zugreift" auf Seite 65.

Java-Anwendungssprachumgebung

Net.Data unterstützt Ihre vorhandenen Java-Anwendungen in der Java-Sprachumgebung. Mit der Unterstützung für Java-Applets und Java-Methoden (bzw. Anwendungen) können Sie über die API für Java Database Connectivity (JDBC**) auf DB2 zugreifen.

Konfigurieren der Java-Sprachumgebung

Zur Verwendung der Java-Sprachumgebung müssen Sie die Net.Data-Einstellungen für die Initialisierung prüfen und die Sprachumgebung definieren.

Prüfen Sie, ob sich eine Konfigurationsanweisung wie die folgende in der Initialisierungsdatei befindet und in einer Zeile steht:

```
ENVIRONMENT (DTW_JAVAPPS)  /opt/IBMNetData/lib/libdtwjava.so ( OUT RETURN_CODE )
   {% CLIETTE "DTW_JAVAPPS" %}
```

In diesem Beispiel ist die CLIETTE-Zeichenfolge am Ende des Eintrags ENVIRONMENT als Kommentar markiert. Wenn die CLIETTE-Zeichenfolge nicht als Kommentar markiert ist, versucht Net.Data, die Java-Anwendung über die Net.Data-Direktverbindung aufzurufen. Sie müssen die Net.Data-Direktverbindung verwenden, wenn Sie Net.Data mit einer anderen Schnittstelle als CGI oder FCGI ausführen. Wenn die CLIETTE-Zeichenfolge ***nicht*** als Kommentar markiert ist und die Net.Data-Direktverbindung ***nicht*** aktiv ist, versucht Net.Data, die Java-Anwendung direkt auszuführen. Wenn Sie Net.Data mit einer anderen Schnittstelle als CGI oder FCGI ausführen, kann dies allerdings ein unvorhersehbares Verhalten verursachen.

Weitere Informationen zur Net.Data-Initialisierungsdatei sowie zu Anweisungen ENVIRONMENT für die Sprachumgebung finden Sie in „Umgebungskonfigurationsanweisungen" auf Seite 29.

Wichtig: In „Definieren der Java-Sprachumgebung mit Cliette" auf Seite 32 finden Sie eine Einführung, wie Sie die Java-Sprachumgebung für den Aufruf von Java-Anwendungen über die Direktverbindung definieren.

Aufrufen von Java-Funktionen

Mit der Java-Sprachumgebung können Sie Java-Funktionsaufrufe von dem Net.Data-Makro aus mit Net.Data-Zeichenfolgen absetzen, die als Parameter verwendet werden. Diese aufgerufenen Java-Funktionsaufrufe können eine Zeichenfolge zurückgeben.

Net.Data bietet die folgenden zwei Methoden, um Java-Funktionen aufzurufen:

1. Net.Data kann die Java-Anwendung direkt startet, indem die Java-Sprachumgebung in den Net.Data-Prozess geladen wird.

   Das ist die einfachste Methode in Bezug auf Installation und Konfiguration. Sie wird empfohlen, wenn Sie Net.Data über CGI oder FCGI ausführen. Diese Methode steht unter Windows, AIX, Linux und Solaris zur Verfügung.
2. Net.Data stellt die Verbindung zur Java-Sprachumgebung über die Net.Data-Direktverbindung her.

   Diese Methode wird empfohlen, wenn Sie Net.Data als Webserver-API ausführen. Die Java-Anwendung wird als getrennter Prozess über die Direktverbindung ausgeführt und ist von einer möglichen Interferenz durch andere stattfindende Java-Verarbeitungen "abgeschirmt". Diese Methode steht unter Windows, OS/2 und AIX zur Verfügung.

Der Prozess zur Einstellung von Net.Data, damit Java-Funktionen aufgerufen werden, hängt von der von Ihnen verwendeten Methode ab.

Aufrufen von Java-Funktionen durch Laden der Java-Anwendung: Gehen Sie wie folgt vor, um die Java-Funktionen durch das Laden der Java-Anwendung aufzurufen:

1. Schreiben Sie Ihre Java-Funktionen, und stellen Sie den Quellcode in die Java-Funktionenbeispieldatei `UserFunctions.java`, die von Net.Data angeboten wird.
2. Fügen Sie Ihrer CLASSPATH-Einstellung die Datei `gnu.regexp-1.0.8.jar` hinzu. Fügen Sie unter AIX z. B. den folgenden Pfad hinzu:

   ```
   /opt/IBMNetData/bkends/javaapps/gnu.regexp-1.0.8.jar
   ```

   Die Datei gnu.regexp-1.0.8.jar wird mit Net.Data ausgeliefert.
3. Erstellen Sie wie folgt die Java-Klassendateien.
   a. Führen Sie die Stapeldatei zur Wiederherstellung in dem Verzeichnis aus, das die Datei `UserFunctions.java` enthält.

b. Nach erfolgreicher Beendigung generiert diese Datei die zwei Java-Klassendateien `dtw_getsignature.class` und `dtw_userfunction.class`.

4. Stellen Sie die Java-Klassendateien `dtw_getsignature.class` und `dtw_userfunction.class` in ein Verzeichnis, in dem die Java-Sprachumgebung von Net.Data sie finden kann. Normalerweise ist das ein Verzeichnis, das in der CLASSPATH-Einstellung angegeben wurde.
5. Führen Sie das Net.Data-Makro aus, das die Java-Sprachumgebung aufruft. Stellen Sie sicher, dass der Sprachumgebungseintrag für die Java-Anwendung in der Konfigurationsdatei von Net.Data nicht die Cliette-Zeichenfolge `CLIETTE "DTW_JAVAPPS"` enthält oder dass diese Cliette-Zeichenfolge als Kommentar markiert ist.

Anmerkung: Führen Sie diese Schritte jedesmal aus, wenn sich der Quellcode Ihrer Java-Funktion ändert. Diese Methode steht unter Windows, AIX, Linux und Solaris zur Verfügung.

Aufrufen von Java-Funktionen durch die Net.Data-Direktverbindung:
Gehen Sie wie folgt vor, um Java-Funktionen über die Direktverbindung aufzurufen:

1. Schreiben Sie Ihre Java-Funktionen.
2. Erstellen Sie eine Net.Data-Cliette für alle Ihre Java-Funktionen.

   Net.Data-Cliettes starten die virtuelle Java-Maschine, über die Ihre Java-Funktionen ausgeführt werden.
3. Definieren Sie eine Cliette in der Java-Anweisung ENVIRONMENT in der Konfigurationsdatei für Direktverbindungen. Bei jeder Einführung neuer Java-Funktionen müssen Sie die Java-Cliette erneut erstellen.
4. Starten Sie Connection Manager.
5. Führen Sie das Net.Data-Makro aus, das die Java-Sprachumgebung aufruft.

Definieren der Java-Sprachumgebungs-Cliette: Ändern Sie die Beispieldatei `makeClas.bat`, oder erstellen Sie eine neue BAT-Datei zum Generieren der Net.Data-Cliette-Klasse `dtw_samp.class` für alle Ihre Java-Funktionen. Das folgende Beispiel zeigt, wie die Stapeldatei `CreateServer` drei Java-Funktionen verarbeitet:

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

Die Stapeldatei verarbeitet die folgenden Dateien zusammen mit der von Net.Data bereitgestellten Stub-Datei `Stub.java`, um `dtw_samp.class` zu erstellen.

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

Erstellen der Java-Funktion: Ändern Sie die Java-Funktionsbeispieldatei `UserFunctions.java` mit Ihren eigenen Java-Funktionen:

```
====================UserFuctions.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)
}
```

Dateistruktur der Java-Sprachumgebung

Net.Data erstellt während der Net.Data-Installation mehrere Verzeichnisse. Diese Verzeichnisse enthalten die Dateien, die zum Erstellen Ihrer Java-Funktionen, Definieren der Cliette und Ausführen des Makros in der Java-Sprachumgebung erforderlich sind:

- Die Java-Beispielfunktion `UserFunctions.java`.
- Die Beispieldatei `makeClas`. Diese Datei erstellt bei ihrer Ausführung eine Net.Data-Cliette-Klasse für Ihre Java-Funktion.
- Die Beispieldatei `launchjv`, die von der Net.Data-Cliette verwendet wird, um die virtuelle Java-Maschine zu starten und Ihre Java-Funktion auszuführen.
- Die Beispieldatei `rebuild`, die von Net.Data verwendet wird, um Ihre eigenen Java-Funktionen zur direkten Ausführung durch das Laden der Java-Sprachumgebung zu erstellen.

Tabelle 13 beschreibt die Verzeichnis- und Dateinamen für die Dateien auf Ihrem Betriebssystem.

Tabelle 13. Zum Erstellen von Java-Funktionen verwendete Dateien

Betriebssystem	Dateiname	Verzeichnis
OS/2	UserFunctions.java	javaapps
	launchjv.com	connect
Windows NT	UserFunctions.java	javaclas
	makeClas.bat	javaclas
	launchjv.bat	connect

Tabelle 13. Zum Erstellen von Java-Funktionen verwendete Dateien (Forts.)

Betriebssystem	Dateiname	Verzeichnis
AIX	UserFunctions.java	javaapps
	launchjv	javaapps
	rebuild	javaapps
	gnu.regexp-1.0.8.jar	javaapps
Linux & Solaris	User Functions.java	javaapps
	rebuild	javaapps
	gnu.regexp-1.0.8.jar	javaapps

Beispiel für die Java-Sprachumgebung

Nach dem Erstellen der Java-Funktion, Definieren der Cliette-Klasse, Ausführen der Datei "rebuild" und Konfigurieren von Net.Data können Sie das Makro mit Verweisen auf die Java-Funktion ausführen.

Das folgende Beispiel-Makro zeigt die Funktionsdefinition und den Funktionsaufruf der Java-Anwendungsfunktion reverse_line().

```
%{ to call the sample }
%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.
%}
```

Perl-Sprachumgebung

Die Perl-Sprachumgebung kann interne Perl-Scripts interpretieren, die in einem FUNCTION-Block des Net.Data-Makros angegeben sind, oder sie kann externe Perl-Scripts verarbeiten, die in separaten Dateien auf dem Server gespeichert sind.

Konfigurieren der Perl-Sprachumgebung

Prüfen Sie, ob sich die folgende Konfigurationsanweisung in der Net.Data-Initialisierungsdatei befindet und in einer Zeile steht:

```
ENVIRONMENT (DTW_PERL)     DTWPERL   ( OUT RETURN_CODE )
```

Weitere Informationen zur Net.Data-Initialisierungsdatei sowie zu Anweisungen ENVIRONMENT für die Sprachumgebung finden Sie in „Umgebungskonfigurationsanweisungen" auf Seite 29.

Für Benutzer der japanischen Version: Einige Zeichen im japanischen SJIS-Zeichensatz können von Perl als Steuerzeichen fehlinterpretiert werden. Zur Lösung dieses Problems gibt es ein Open Source-Paket mit dem Namen **jperl**. Laden Sie das Paket herunter, installieren Sie es, und fügen Sie anschließend die Anweisung `use I18N::Japanese.pm` in den Header des Perl-Scripts ein.

Aufrufen externer Perl-Scripts

Aufrufe an externe Perl-Scripts werden in einem FUNCTION-Block durch eine EXEC-Anweisung mit der folgenden Syntax angegeben:

```
%EXEC{ perl_script_name [optional parameters] %}
```

Wichtiger Hinweis: Stellen Sie sicher, dass *perl_script_name*, der Perl-Script-Name, in einem für die Konfigurationsvariable EXEC_PATH angegebenen Pfad in der Net.Data-Initialisierungsdatei aufgelistet ist.

```
%FUNCTION(DTW_PERL) perl1() {
%EXEC{ MyPerl.pl %}
%}
```

Übergeben von Parametern

Es gibt zwei Möglichkeiten, Informationen an ein Programm zu übergeben, das durch die Perl-Sprachumgebung (DTW_PERL) aufgerufen wird: direkt und indirekt.

Direkt Übergeben Sie Parameter direkt beim Aufruf des Perl-Scripts. Beispiel:

```
%DEFINE INPARM1 = "SWITCH1"

%FUNCTION(DTW_PERL) sys1() {
 %EXEC{
    MyPerl.pl $(INPARM1) "literal string"
 %}
%}
```

Die Net.Data-Variable INPARM1 wird mit einem Verweis versehen und an das Perl-Script übergeben. Die Parameter werden auf gleiche Weise an das Perl-Script übergeben wie bei einem Aufruf des Perl-Scripts über die Befehlszeile. Die Parameter, die dem Perl-Script mit dieser Methode übergeben werden, werden als Eingabeparameter behandelt und Änderungen an den Werten werden nicht an Net.Data zurückgemeldet.

Indirekt

Übergeben Sie Parameter indirekt beim Aufruf des Perl-Scripts mit einer der folgenden Methoden:

- Lassen Sie Net.Data Eingabeparameter an das Perl-Script als Umgebungsvariablen übergeben. Das Perl-Script kann die Parameter dann über die Umgebungsvariablen abrufen.
- Lassen Sie das Perl-Script Ausgabeparameter an die Sprachumgebung durch Schreiben der Daten in eine Datei zurückgeben, deren Namen Net.Data in der Umgebungsvariablen DTWPIPE übergibt. Die Daten, die das Perl-Script an Net.Data übergibt, sollten die folgende Syntax haben:

  name="value"

  Wenn mehrere Datenelemente vorhanden sind, trennen Sie die einzelnen Elemente durch ein Zeilenvorschubzeichen oder ein Leerzeichen.

  Wenn ein Variablenname den gleichen Namen wie ein Parameter OUT oder INOUT hat und die obige Syntax verwendet, ersetzt der neue Wert den aktuellen Wert. Wenn ein Variablenname mit keinem Parameter OUT oder INOUT übereinstimmt, wird er von Net.Data ignoriert.

Das folgende Beispiel zeigt, wie Net.Data Variablen von einem Makro übergibt.

```
%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()
%}
```

Wenn das Perl-Script die externe Datei `today.pl` ist, kann die gleiche Funktion so wie im nächsten Beispiel geschrieben werden:

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

Sie können Net.Data-Tabellen an ein Perl-Script übergeben, das von der Perl-Sprachumgebung aufgerufen wird. Das Perl-Script greift auf die Werte eines Net.Data-Makrotabellenparameters anhand ihrer Net.Data-Namen zu. Die Spaltenüberschriften und Feldwerte sind in Variablen enthalten, die mit dem Tabellennamen und der Spaltennummer angegeben werden. Zum Beispiel sind in der Tabelle `myTable` die Spaltenüberschriften `myTable_N_` *`j`* und die Feldwerte `myTable_V_i_`*`j`*, wobei *i* die Zeilennummer und *j* die Spaltennummer ist. Die Anzahl der Zeilen und Spalten für die Tabelle sind `myTable_ROWS` und `myTable_COLS`.

REPORT- und MESSAGE-Blöcke in FUNCTION-Blöcken

REPORT- und MESSAGE-Blöcke sind so wie in jedem anderen FUNCTION-Abschnitt zulässig. Sie werden von Net.Data und nicht von der Sprachumgebung verarbeitet. Ein Perl-Script kann jedoch Text in den Standardausgabedatenstrom schreiben, um als Teil der Webseite aufgenommen zu werden.

Beispiel für die Perl-Sprachumgebung

Das folgende Beispiel zeigt, wie Net.Data eine Tabelle durch Ausführen des externen Perl-Scripts generiert.

```
%define {
 c = %TABLE(20)
 rows = "5"
 columns = "5" %}
 %function(DTW_PERL) genTable(in rows, in columns, out table) {
 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);  %}

 %message{
 default: "genTable: Unexpected Error"
 %}
```

```
 %}

%HTML(REPORT){
 @genTable(rows, columns, c)
 return code is $(RETURN_CODE)
 %}
```

Ergebnisse: `genTable` generiert Folgendes:

```
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
```

REXX-Sprachumgebung

In der REXX-Sprachumgebung können Sie REXX-Programme ausführen, die für die Ausführung in der DTW_REXX-Umgebung geschrieben wurden. Die Net.Data REXX-Sprachumgebung verfügt über Steuerelemente, mit denen REXX-Programme große Datenmengen auf einfache Weise zurückgeben können.

Net.Data stellt außerdem Unterstützung für die REXX-Anweisung SAY zur Verfügung, die die Ausgabe an den Browser leitet, und zwar unabhängig von der Webserverumgebung, die Sie für Net.Data verwenden. Wenn Sie natives REXX mit Hilfe der FastCGI-, GWAPI- oder Servlet-Konfiguration des Webservers ausführen, wird die Ausgabe der REXX-Anweisung SAY an die Protokolldatei der Webserver und nicht an den Browser weitergeleitet. Dies gilt nicht für REXX-Programme, die für die Ausführung in der DTW_REXX-Umgebung geschrieben wurden.

Unterstützung für Variablen: Damit REXX-Programme große Datenmengen einfach zurückgeben können, fügt Net.Data automatisch Code am Anfang und am Ende des REXX-Programms ein. Dieser Code soll Variablen bearbeiten, die in der DTW_REXX-Funktionsanweisung angegeben werden.

Unterstützung für REXX-Anweisung SAY (FastCGI-, GWAPI- und SERVLET-Umgebung): REXX-Anweisungen SAY werden automatisch in REXX-Zuordnungsanweisungen konvertiert, bevor das REXX-Programm ausgeführt wird. Net.Data fügt dem REXX-Programm automatisch Code hinzu, der die Ausgabe der ursprünglichen REXX-Anweisungen SAY an den Browser leiten soll.

Verwendung von REXX-Subroutinen und -Funktionen: Da Net.Data Code am Anfang und am Ende des REXX-Programms einfügt, muss die REXX-Hauptroutine mit der letzten Anweisung des REXX-Programms enden. Wenn Sie REXX-Subroutinen oder -Funktionen verwenden, müssen Sie sicherstellen, dass die letzte Anweisung des REXX-Programms der REXX-Hauptroutine zugeordnet ist. Das folgende Beispiel zeigt die Verwendung einer Subroutine und Funktion in einem REXX-Programm, das für die Ausführung in der DTW_REXX-Umgebung geschrieben wurde:

```
%function(DTW_REXX) genData(out s1,s2) {
         call subrtn1
         s2=funrtn1()
        signal rexxEnd  /* Go to end of Program */
      subrtn1: PROCEDURE EXPOSE s1
         string1 = "data for s1"
         return 0
      funrtn1: PROCEDURE
         retvar = "data for s2"
         return retvar
      rexxEnd:          /* End of Main Program */
         return 0
    %}
    %HTML (Report) {


    @genData(a,c)

  Value for s1: $(a)

  Value for s2: $(c)
   %}
```

Verwendung der REXX-Anweisungen EXIT und RETURN: Net.Data fügt REXX-Programmen automatisch Code hinzu, der Werte für Ausgabevariablen zur Verfügung stellt und die Ausgabe von SAY-Anweisungen an den Browser weiterleitet. Wenn das REXX-Programm eine Anweisung RETURN von der Hauptroutine oder eine Anweisung EXIT an einer beliebigen Position (mit Ausnahme der letzten Anweisung des REXX-Programms) ausgibt, wird der Code, den Net.Data an das REXX-Programm anfügte, nicht ausgeführt. Dies hat einen Verlust von Ausgabevariablen und der Ausgabe der SAY-Anweisungen zur Folge.

Wenn Sie ein REXX-Programm vor Erreichen der letzten Anweisung beenden müssen, sollten Sie zur letzten Anweisung in dem REXX-Programm verzweigen, die normalerweise die Beendigung durchführt. Wenn Sie die Anweisung RETURN oder EXIT zum Beenden des REXX-Hauptprogramms verwenden, muss sie die letzte Anweisung im REXX-Programm sein. Dies schließt REXX-Kommentaranweisungen ein. Beispiel:

```
%function(DTW_REXX) genData(out s1,s2) {
......
If S2 < 0 Then signal rexxEnd
......
......
rexxEnd:
/* This comment must be before the following
RETURN statement */
return 0
%}
%HTML (Report) {
@genData(a,c)
......
%}
```

Externe REXX-Programme von einer DTW_REXX-Funktion aufrufen: Sie können ein REXX-Programm mit Hilfe der Net.Data-Anweisung %EXEC von einer DTW_REXX-Funktion oder mit Hilfe von REXX-Methoden von einem REXX-Programm aus aufrufen.

Wird ein externes REXX-Programm mit Hilfe der Net.Data-Anweisung %EXEC aufgerufen, fügt Net.Data automatisch Code am Anfang und am Ende des REXX-Programms ein, um Ausgabevariablen zu bearbeiten und Ausgabe von REXX-Anweisungen SAY an den Browser weiterzuleiten.

Wird ein REXX-Programm mit Hilfe von REXX-Methoden aufgerufen, erhält Net.Data nicht die Steuerung und fügt dem REXX-Programm keinen Code hinzu. Das aufgerufene REXX-Programm muss die Ausgabe mit Hilfe von REXX-Standardkonventionen zurück an das aufrufende REXX-Programm übergeben. Bei einer Ausführung in einer GWAPI- oder SERVLET-Umgebung wird die Ausgabe der REXX-Anweisungen SAY an die Protokolldatei der Webserver gesendet.

Konfigurieren der REXX-Sprachumgebung

Zur Verwendung der REXX-Sprachumgebung müssen Sie die Net.Data-Einstellungen für die Initialisierung prüfen und die Sprachumgebung definieren.

Prüfen Sie, ob sich die folgende Konfigurationsanweisung in der Initialisierungsdatei befindet und in einer Zeile steht:

```
ENVIRONMENT (DTW_REXX)     DTWREXX   ( OUT RETURN_CODE )
```

Weitere Informationen zur Net.Data-Initialisierungsdatei sowie zu ENVIRONMENT-Anweisungen für die Sprachumgebung finden Sie in *Net.Data Verwaltung und Programmierung*.

Ausführen von REXX-Programmen

In der REXX-Sprachumgebung können Sie sowohl interne REXX-Programme als auch externe REXX-Programme ausführen. Ein internes REXX-Programm ist ein REXX-Programm, dessen Quelle sich im Makro befindet. Bei einem externen REXX-Programm befindet sich die Quelle des REXX-Programms in einer externen Datei.

Gehen Sie wie folgt vor, um ein internes REXX-Programm auszuführen:

Definieren Sie eine Funktion, die die REXX-Sprachumgebung (DTW_REXX) verwendet und den REXX-Code in der Funktion (Abschnitt zur Ausführung in der Sprachumgebung) enthält.

Beispiel: Eine Funktion mit einem internen REXX-Programm

```
%function(DTW_REXX) helloWorld() {
        SAY 'Hello World'
%}
```

Gehen Sie wie folgt vor, um ein externes REXX-Programm auszuführen:

Definieren Sie eine Funktion, die die REXX-Sprachumgebung (DTW_REXX) verwendet und einen Pfad zu dem REXX-Programm enthält, das in einer EXEC-Anweisung ausgeführt werden soll.

Beispiel: Eine Funktion mit einer EXEC-Anweisung, die auf ein externes Programm zeigt

```
%function(DTW_REXX) externalHelloWorld() {
%EXEC{ helloworld.cmd%}
%}
```

Wichtiger Hinweis: Stellen Sie sicher, dass der REXX-Dateiname in einem Pfad für die Konfigurationsvariable EXEC_PATH in der Net.Data-Initialisierungsdatei aufgelistet ist. Informationen zum Definieren der Konfigurationsvariablen EXEC_PATH finden Sie in „EXEC_PATH” auf Seite 25.

Übergeben von Parametern an REXX-Programme

Es gibt zwei Möglichkeiten, Informationen an ein REXX-Programm zu übergeben, das durch die REXX-Sprachumgebung (DTW_REXX) aufgerufen wird: direkt und indirekt.

Direkt Mit der Anweisung %EXEC übergeben Sie Parameter direkt an ein externes REXX-Programm. Beispiel:

```
%FUNCTION(DTW_REXX) rexx1() {
  %EXEC{CALL1.CMD $(INPARM) "literal string"   %}
%}
```

Die Net.Data-Variable INPARM1 wird mit einem Verweis versehen und an das externe REXX-Programm übergeben. Das REXX-Programm kann durch Verwendung der Anweisung REXX PARSE ARG auf die Variable verweisen. Die Parameter, die dem REXX-Programm mit dieser Methode übergeben werden, werden als Eingabeparameter behandelt, und Änderungen der Werte werden nicht an Net.Data zurückgemeldet. (Die dem Programm übergebenen Parameter können vom Programm verwendet und bearbeitet werden, Änderungen an den Parametern werden jedoch nicht an Net.Data zurückgemeldet.)

Indirekt

Mit dem *Variablenpool* des REXX-Programms übergeben Sie Parameter indirekt. Wenn ein REXX-Programm gestartet wird, wird vom REXX-Interpreter ein Bereich, der Informationen zu allen Variablen enthält, erstellt und verwaltet. Dieser Bereich wird als Variablenpool bezeichnet.

Wenn eine Funktion der REXX-Sprachumgebung (DTW_REXX) aufgerufen wird, werden Eingabeparameter (IN) oder Ein-/Ausgabeparameter (INOUT) von der REXX-Sprachumgebung gespeichert, bevor das REXX-Programm ausgeführt wird. Wenn das REXX-Programm aufgerufen wird, kann es direkt auf diese Variablen zugreifen. Nach erfolgreicher Beendigung des REXX-Programms ermittelt die Sprachumgebung DTW_REXX, ob es OUT- oder INOUT-Funktionsparameter gibt. Trifft dies zu, ruft die Sprachumgebung den Wert, der dem Funktionsparameter entspricht, aus dem Variablenpool ab und aktualisiert den Funktionsparameterwert mit dem neuen Wert. Wenn Net.Data die Steuerung erhält, aktualisiert es alle OUT- oder INOUT-Parameter mit den neuen, aus der REXX-Sprachumgebung erhaltenen Werten. Beispiel:

```
%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)
%}
```

Im obigen Beispiel übergibt der Aufruf *@double_func* zwei Parameter, *a* und *b*. Die REXX-Funktion *double_func* verdoppelt den ersten Parameter und speichert das Ergebnis im zweiten Parameter. Wenn Net.Data das Makro aufruft, hat *b* den Wert 6.

Sie können Net.Data-Tabellen an ein REXX-Programm übergeben. Ein REXX-Programm greift auf die Werte eines Net.Data-Makrotabellenparameters als REXX-Stammvariablen zu. Für ein REXX-Programm sind die Spaltenüberschriften und Feldwerte in Variablen enthalten, die mit dem Tabellennamen und der Spaltennummer angegeben werden. Zum Beispiel sind in der Tabelle `myTable` die Spaltenüberschriften `myTable_V.`*`j`* und die Feldwerte `myTable_V.`*`i.j`*, wobei *i* die Zeilennummer und *j* die Spaltennummer ist. Die Anzahl der Zeilen in der Tabelle ist `myTable_`*`ROWS`*, die Anzahl der Spalten `myTable_`*`COLS`*.

Leistungsoptimierung für das Betriebssystem AIX:
Wenn Sie die REXX-Sprachumgebung auf Ihrem AIX-System oft aufrufen, sollten Sie die Umgebungsvariable RXQUEUE_OWNER_PID auf 0 setzen. Makros, die viele Aufrufe an die REXX-Sprachumgebung ausführen, können viele Prozesse erstellen, die die Systemressourcen aufbrauchen.

Sie haben drei Möglichkeiten, die Umgebungsvariable einzustellen:

- Im Makro mit Hilfe der integrierten Funktion DTW_SETENV:
  ```
  @DTW_rSETENV("RXQUEUE_OWNER_PID", "0")
  ```
- In der AIX-Systemumgebungsdatei durch Einfügen der folgenden Anweisung:
  ```
  /etc/environment: RXQUEUE_OWNER_PID = 0
  ```

  Diese Methode beeinflusst das Verhalten von REXX auf der gesamten Maschine.
- In der Umgebungsdatei des HTTP-Webservers; fügen Sie z. B. für Domino Go Webserver die folgende Anweisung ein:
  ```
  InheritEnv RXQUEUE_OWNER_PID = 0
  ```

  Diese Methode beeinflusst das Verhalten von REXX auf dem Webserver.

Beispiel für die REXX-Sprachumgebung

Das folgende Beispiel zeigt ein Makro, das eine REXX-Funktion aufruft, um eine Net.Data-Tabelle zu generieren, die zwei Spalten und drei Zeilen enthält. Nach dem Aufruf der REXX-Funktion wird eine integrierte Funktion, DTW_TB_TABLE(), aufgerufen, um eine HTML-Tabelle zu generieren, die an den Browser zurückgesendet wird.

```
%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)
%}
```

Ergebnisse:

```
<table>
  <tr>
<th>COL1</th>
<th>COL2</th>
</tr>
<tr>
<td>[1 1]</td>
<td>[1 2],</td>
</tr>
<tr>
<td>[2 1]</td>
<td>[2 2],</td>
</tr>
<tr>
<td>[3 1]</td>
<td>[3 2],</td>
</tr>
</table>
```

SYSTEM-Sprachumgebung

Die SYSTEM-Sprachumgebung unterstützt das Ausführen von Befehlen und das Aufrufen externer Programme.

Konfigurieren der SYSTEM-Sprachumgebung

Fügen Sie der Initialisierungsdatei die folgende Konfigurationsanweisung in einer Zeile hinzu:

```
ENVIRONMENT (DTW_SYSTEM)   DTWSYS    ( OUT RETURN_CODE )
```

Weitere Informationen zur Net.Data-Initialisierungsdatei sowie zu ENVIRONMENT-Anweisungen für die Sprachumgebung finden Sie in *Net.Data Verwaltung und Programmierung*.

Absetzen von Befehlen und Aufrufen von Programmen

Definieren Sie zum Absetzen eines Befehls eine Funktion, die die SYSTEM-Sprachumgebung (DTW_SYSTEM) verwendet und einen Pfad zu dem Befehl enthält, der in einer EXEC-Anweisung abgesetzt werden soll. Beispiel:

```
%FUNCTION(DTW_SYSTEM) sys1() {
      %EXEC { ADDLIBLE.CMD %}
 %}
```

Sie können den Pfad zu ausführbaren Objekten kürzen, wenn Sie die Konfigurationsvariable EXEC_PATH zum Definieren von Pfaden zu Verzeichnissen mit den Objekten (wie Befehle und Programme) verwenden. Informationen zum Definieren der Konfigurationsvariablen EXEC_PATH finden Sie in „EXEC_PATH" auf Seite 25.

Beispiel 1: Aufrufen eines Programms

```
%FUNCTION(DTW_SYSTEM) sys3() {
      %EXEC {MYPGM.EXE %}
 %
```

Übergeben von Parametern an Programme

Es gibt zwei Möglichkeiten, Informationen an ein Programm zu übergeben, das durch die SYSTEM-Sprachumgebung (DTW_SYSTEM) aufgerufen wird: direkt und indirekt.

Direkt Sie übergeben Parameter direkt beim Aufruf des Programms. Beispiel:

```
%DEFINE INPARM1 = "SWITCH1"

%FUNCTION(DTW_SYSTEM) sys1() {
 %EXEC{
  CALL1.CMD  $(INPARM1) "literal string"
 %}
%}
```

Die Net.Data-Variable INPARM1 wird mit einem Verweis versehen und an das Programm übergeben. Die Parameter werden dem Programm auf die gleiche Weise übergeben wie bei einem Aufruf des Programms von der Befehlszeile aus. Die Parameter, die dem Programm mit dieser Methode übergeben werden, werden als Eingabeparameter behandelt, und Änderungen der Werte werden nicht an Net.Data zurückgemeldet (die dem Programm übergebenen Parameter können vom Programm verwendet und bearbeitet werden, Änderungen an den Parametern werden jedoch nicht an Net.Data zurückgemeldet).

Indirekt

Die SYSTEM-Sprachumgebung kann Net.Data-Variablen nicht direkt übergeben oder abrufen. Daher werden sie Programmen auf folgende Art zur Verfügung gestellt:

- Net.Data übergibt Eingabeparameter als Umgebungsvariablen an das Programm. Das Programm kann die Parameter dann über die Umgebungsvariablen abrufen.
- Das Programm übergibt Ausgabeparameter an die Sprachumgebung zurück, indem es in eine benannte Pipe schreibt, deren Name von Net.Data an die Umgebungsvariable DTWPIPE übergeben wird. Schreiben Sie mit der folgenden Syntax Daten in die benannte Pipe:

  name="value"

  Wenn mehrere Datenelemente vorhanden sind, trennen Sie die einzelnen Elemente durch ein Zeilenvorschubzeichen oder ein Leerzeichen.

  Wenn ein Variablenname den gleichen Namen wie ein Ausgabeparameter hat und die obige Syntax verwendet, ersetzt der neue Wert den aktuellen Wert. Wenn ein Variablenname mit keinem Ausgabeparameter übereinstimmt, wird er von Net.Data ignoriert.

Das folgende Beispiel zeigt, wie Net.Data Variablen von einem Makro übergibt.

```
%FUNCTION(DTW_SYSTEM) sys1 (IN P1, OUT P2, P3) {
  %EXEC {
     UPDPGM
  %}
%}
```

Sie können Net.Data-Tabellen an ein Programm übergeben, das von der SYSTEM-Sprachumgebung aufgerufen wird. Das Programm greift auf die Werte eines Net.Data-Makrotabellenparameters anhand der Net.Data-Namen dieser Werte zu. Die Spaltenüberschriften und Feldwerte sind in Variablen enthalten, die mit dem Tabellennamen und der Spaltennummer angegeben werden. Zum Beispiel sind in der Tabelle myTable die Spaltenüberschriften myTable_N_*j* und die Feldwerte myTable_V_i_*j*, wobei *i* die Zeilennummer und *j* die Spaltennummer ist. Die Anzahl der Zeilen in der Tabelle ist myTable_ROWS, die Anzahl der Spalten myTable_COLS.

Beispiel für die SYSTEM-Sprachumgebung

Das folgende Beispiel zeigt ein Makro, das eine Funktionsdefinition mit den drei Parametern P1, P2 und P3 enthält. P1 ist ein Eingabeparameter (IN), P2 und P3 sind Ausgabeparameter (OUT). Die Funktion ruft ein Programm, UPDPGM, auf, das den Parameter P2 mit dem Wert von P1 aktualisiert und P3 auf eine Zeichenfolge setzt. Vor der Verarbeitung der Anweisung im %EXEC-Block speichert die Sprachumgebung DTW_SYSTEM P1 und den zugehörigen Wert im Umgebungsbereich.

```
%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.
<p>
 @sys1("ValueOfParm1", MYPARM2, MYPARM3)

<p>
 After the function call, the value of MYPARM2 is "$(MYPARM2)",
 and the value of MYPARM3 is "$(MYPARM3)".
<p>
 %}
```

Kapitel 7. Optimieren der Leistung

Das Optimieren der Leistung ist ein wichtiger Bestandteil der Systemoptimierung. In diesem Kapitel werden Strategien zum Optimieren der Leistung von Net.Data erörtert. Folgende Themen werden behandelt:

- Verwenden der Webserver-APIs
- „Verwalten von Verbindungen" auf Seite 200
- „Net.Data-Caching" auf Seite 205
- „Festlegen der Fehlerprotokollstufe" auf Seite 233
- „Optimieren der Sprachumgebungen" auf Seite 233

Stellen Sie zudem sicher, dass Ihr Webserver entsprechend optimiert wurde. Die Leistung Ihres Webservers wirkt sich direkt auf die Antwortzeit aus, unabhängig davon, wie schnell Net.Data ein Makro oder eine Direktanforderung verarbeitet.

Verwenden der Webserver-APIs

Sie können die Leistung optimieren, indem Sie Net.Data mit einer Webserver-API, wie z. B. APAPI, anstelle von CGI aufrufen. Wenn Net.Data über eine Webserver-API ausgeführt wird, erfolgt die Ausführung von Net.Data als Thread im Webserverprozess. Der Webserverprozess besteht aus mehreren Threads und erstellt eine Umgebung, in der mehrere Net.Data-Anforderungen verarbeitet werden können. Diese Anforderungen werden im gleichen Adressraum gleichzeitig verarbeitet. Dadurch entfällt der beim Aufrufen von Net.Data als CGI-Prozess entstehende Systemaufwand.

Hinweis: Die Verwendung einer Webserver-API optimiert die Leistung ohne Anwendungsisolation. Weil Net.Data in einer Umgebungen mit mehreren Threads ausgeführt wird, können Probleme mit dem Webserver auftreten, die ihn möglicherweise inaktivieren könnten. Dazu gehören z. B. Fehler in benutzerdefinierten Sprachumgebungen, ungültige Aufrufe und sogar Datenbankausfälle. Wenn Sie die Wahl treffen, ob Sie eine der Webserver-APIs verwenden wollen, sollten Sie bestimmen, ob für Ihre Anwendung die Leistung oder die Anwendungsisolation wichtiger ist.

Verwenden von FastCGI

FastCGI bietet optimierte Leistung mit der Anwendungsisolation von CGI. Sie können Net.Data mit FastCGI auf allen Webservern verwenden, die FastCGI unterstützen. Weitere Informationen zur Konfiguration von FastCGI finden Sie in „Verwalten von Verbindungen".

Sie können FastCGI so optimieren, dass die Anzahl von Prozessen ausgeführt wird, die für die Barbeitung der eingehenden Anforderungen erforderlich ist. Wenn Sie z. B. 100 Anforderungen pro Sekunde verarbeiten wollen und jede Anforderung eine halbe Sekunde dauert, sollten Sie die Anweisung NumProcessess in der Konfigurationsdatei FastCGI auf 50 setzen.

FastCGI wird in allen Sprachumgebungen unterstützt; bei Verwendung von Oracle und ODBC ist jedoch eine Direktverbindung (Live Connection) erforderlich.

Gehen Sie wie folgt vor, um die Anzahl simultaner Prozesse zu optimieren:

1. Öffnen Sie die Konfigurationsdatei, in der der Konfigurationsparameter für Prozesse definiert ist.

   Bei Apache und IBM HTTP ist dies die Datei `httpd.conf`.
2. Ändern Sie wie folgt den Wert für den Konfigurationsparameter, der die Anzahl von Prozessen angibt:
   - Bei Apache: `Process=`*num*
   - Bei ISC: `NumProcess=`*num*

Dabei ist *num* die Anzahl der Prozesse.

Verwalten von Verbindungen

Net.Data enthält eine Komponente namens Direktverbindung für die Verwaltung von Verbindungen der Datenbank und der virtuellen Java-Maschinen. Die Direktverbindung hält dauerhaft Verbindungen aufrecht, um die Leistung zu verbessern. Für einige Net.Data-Funktionen ist eine lange Startzeit erforderlich. Bevor eine Datenbankabfrage abgesetzt werden kann, muss sich z. B. der Prozess gegenüber dem Datenbankverwaltungssystem (DMBS) identifizieren und eine Verbindung zur Datenbank herstellen. Dieser Vorgang nimmt oft einen bedeutenden Teil der für die Ausführung von Net.Data-Makros, die auf eine Datenbank zugreifen, erforderlichen Verarbeitungszeit in Anspruch. Aufgrund der Arbeitsweise der CGI-Programme fallen diese Startkosten bei jeder Anforderung an den Webserver an. Net.Data bietet eine Direktverbindung für die Betriebssysteme OS/2, Windows NT, AIX, Solaris und Linux, um dauerhaft Verbindungen aufrecht zu erhalten.

In den folgenden Abschnitten wird die Direktverbindung beschrieben.

- „Informationen zur Direktverbindung"
- „Vorteile der Direktverbindung" auf Seite 202
- „Einsatzmöglichkeiten der Direktverbindung" auf Seite 203
- „Starten von Connection Manager" auf Seite 203
- „Verarbeitungsablauf für Net.Data und Direktverbindung" auf Seite 204

Informationen zur Direktverbindung

Mit Hilfe einer Direktverbindung kann eine entscheidende Leistungssteigerung erzielt werden, da sie den Startaufwand verringert. Der Spareffekt ist darauf zurückzuführen, dass ständig mindestens ein Prozess ausgeführt wird, der die Startfunktionen übernimmt. Diese Prozesse warten dann auf Serviceanforderungen. Sie können Direktverbindungen ausführen, wenn Sie Net.Data als CGI- oder FastCGI-Programm verwenden oder wenn Sie ein Plug-In für eine Webserver-API verwenden.

Die Direktverbindung besteht aus Connection Manager und Cliettes. *Cliettes* sind von Connection Manager gestartete Prozesse, die solange aktiv bleiben, wie auch der Server aktiv ist. Cliettes verarbeiten Daten und kommunizieren mit Net.Data-Sprachumgebungen, die Sie in der Initialisierungsdatei mit dem Schlüsselwort CLIETTE angeben. Jeder Cliette-Typ ist für die Bearbeitung einer spezifischen Sprachumgebungsfunktion konzipiert, z. B. die DB2-Cliette, die die Verbindung zur DB2-Datenbank herstellt und Operationen zur Ausführung von SQL-Aufrufen definiert, bevor Net.Data-Makros von Net.Data verarbeitet werden.

Der Name der ausführbaren Datei wird in der Konfigurationsdatei für die Direktverbindung angegeben (`dtwcm.cnf`). Abb. 25 auf Seite 202 zeigt die Interaktion zwischen der Direktverbindung, dem Makro und den Sprachumgebungen.

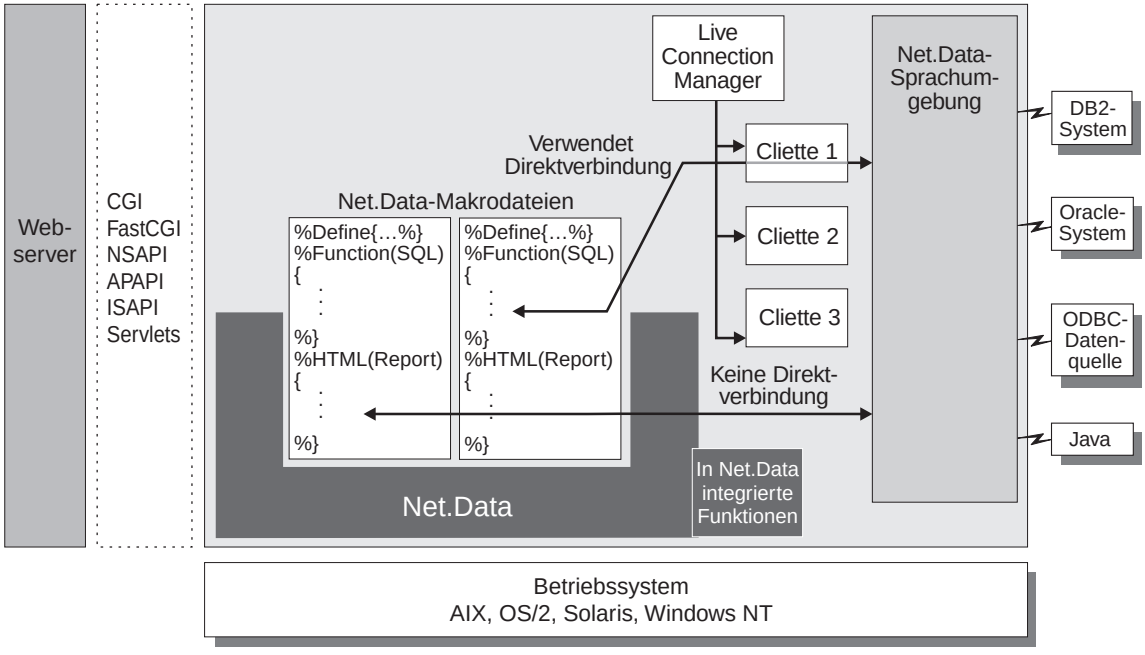


Abbildung 25. Direktverbindung mit Cliettes

In den folgenden Abschnitten wird die Direktverbindung näher beschrieben. Informationen zum Konfigurieren der Direktverbindung finden Sie in „Konfigurieren der Direktverbindung" auf Seite 36.

Vorteile der Direktverbindung

Die Verwendung der Direktverbindung bietet die folgenden Vorteile:

- **Verbesserte Leistung**

  Der wiederholte Einsatz von bestehenden Verbindungen ist effizienter als die Verbindungen immer wieder neu herstellen zu müssen. In der Regel schlägt die Verbindungszeit zu Buche, wenn Sie kleine SQL-Anweisungen anfordern (z. B. einfache Abfragen einer Datenbank mit weniger als 100.000 Zeilen) oder wenn Ihre Verbindung komplex ist (z. B. ferne Server).

- **Zugriff auf mehrere Datenbanken**

  Über eine Direktverbindung können Sie mit einem Net.Data-Makro zu mehreren Datenbanken gleichzeitig eine Verbindung herstellen. Dies wird dadurch möglich, dass jede Datenbank über eindeutige Cliettes verfügt, d. h., Net.Data kommuniziert einfach mit mehreren Cliettes.

Einsatzmöglichkeiten der Direktverbindung

Sie können die Direktverbindung im CGI-, FastCGI- oder API-Modus für die Kommunikation mit der Datenbank oder der virtuellen Java-Maschine verwenden. Sie können zudem die Vorteile der Direktverbindung nutzen, wenn Ihre Anwendung Daten aus mehreren Datenbanken erfordert.

Die Direktverbindung in Kombination mit einem API-Plug-In verbessert bei vielen Systemen je nach Auslastung und Konfiguration die Leistung. Experimentieren Sie mit Ihrem System, um die für Sie optimal geeignete Konfiguration zu ermitteln.

Für viele Anwendungen sind Leistungssteigerungen ohne Direktverbindung möglich, wenn der Befehl ACTIVATE DATABASE verwendet wird, um Zeit beim Herstellen von Datenbankverbindungen zu sparen. Nähere Angaben zu dem von Ihrer Datenbank verwendeten Befehl finden Sie in der Datenbankdokumentation. Prüfen Sie außerdem anhand der Dokumentation zu Ihrem Betriebssystem, ob weitere Möglichkeiten zur Optimierung der Leistung beschrieben sind.

Anforderung: Im FastCGI- oder API-Modus ist für die Sprachumgebungen für ODBC und Oracle eine Direktverbindung erforderlich.

Starten von Connection Manager

Connection Manager ist eine separate, ausführbare Datei, die mit Net.Data geliefert wird und den Namen `dtwcm` hat. Starten Sie Connection Manager, wenn Sie den Webserver starten.

Connection Manager liest nach dem Start eine Konfigurationsdatei und startet eine Gruppe von Prozessen. In jedem Prozess fängt Connection Manager mit der Ausführung einer bestimmten Cliette an. Informationen zum Konfigurieren der Direktverbindung finden Sie in „Konfigurieren der Direktverbindung" auf Seite 36.

Gehen Sie wie folgt vor, um Connection Manager unter Windows NT und OS/2 zu starten:

1. Wechseln Sie in der Befehlszeile in das Verzeichnis <*inst_dir*>\connect\.
2. Geben Sie `dtwcm` ein.

Dabei steht <*inst_dir*> für das Net.Data-Installationsverzeichnis.

Gehen Sie wie folgt vor, um Connection Manager unter AIX zu starten:

1. Wechseln Sie in der Befehlszeile in das Verzeichnis `/usr/lpp/internet/db2www/db2/`.
2. Geben Sie `dtwcm` ein.

Gehen Sie wie folgt vor, um Connection Manager mit der Nachrichtenoption zu starten:

Nachrichten für Connection Manager werden standardmäßig unterdrückt. Wenn Nachrichten für Connection Manager angezeigt werden sollen, verwenden Sie die Option -d beim Start von Connection Manager.

Geben Sie in der Befehlszeile `dtwcm -d` ein.

Nach der Verwendung der Option -d müssen Sie Connection Manager erneut starten, um die Nachrichten wieder zu unterdrücken.

Gehen Sie wie folgt vor, um Connection Manager automatisch als Windows NT-Service zu starten:

Unter Windows NT können Sie angeben, dass Connection Manager nicht von der Befehlszeile aus, sondern als Windows NT-Service gestartet werden soll. Die Ausführung von Connection Manager als Windows NT-Service ermöglicht den automatischen Start von Connection Manager bei jedem Start der Maschine.

Hinweis: Starten Sie Connection Manager von der Befehlszeile aus, bevor Sie den automatischen Start definieren, um sicherzustellen, dass die Konfigurationsdatei für Direktverbindungen korrekt ist.

1. Wählen Sie in der NT-Task-Leiste **Start->Einstellungen->Systemsteuerung->Dienste** aus.
2. Wählen Sie **Net.Data Live Connection** aus, und klicken Sie danach den Knopf **Starten** an.
3. Wählen Sie **Startart** aus, und klicken Sie anschließend **OK** an.

Anmerkung: Das Protokollieren ist standardmäßig ausgeschaltet. Wählen Sie die Net.Data-Direktverbindung im Fenster **Dienste** aus, und geben Sie `-l` im Feld **Startparameter** ein.

Verarbeitungsablauf für Net.Data und Direktverbindung

Nach der Konfiguration und dem Start der Datenbank, des Webservers und von Connection Manager umfasst die Net.Data-Verarbeitung bei Aktivierung der Direktverbindung in der Regel die folgenden drei Schritte:

1. Der Webserver empfängt eine Anforderung und startet einen FastCGI-, CGI- oder API-Prozess zum Ausführen von Net.Data.

2. Net.Data startet die Verarbeitung des Net.Data-Makros.
3. Wenn Net.Data einen Funktionsaufruf feststellt, der eine Direktverbindung verwendet, wird ermittelt, welcher Typ von Cliette aus der Initialisierungsdatei erforderlich ist. Bei DB2 ist der Cliette-Typ häufig ein auf dem DB2-Datenbanknamen basierender Name wie DTW_SQL:CELDIAL.
4. Net.Data fordert von Connection Manager eine Cliette dieses Typs an.
5. Connection Manager sucht nach verfügbaren Cliettes dieses Typs. Wenn keine verfügbar sind, stellt Connection Manager die Anforderung in eine Warteschlange und verarbeitet sie, wenn der richtige Cliette-Typ verfügbar ist.
6. Wenn eine Cliette zur Verfügung steht, teilt Connection Manager Net.Data mit, wie mit der Cliette kommuniziert werden muss.
7. Net.Data fordert die Cliette auf, die Funktion zu verarbeiten.
8. Dieser Prozess wird ab Schritt 3 wiederholt, bis die Verarbeitung des Net.Data-Makros abgeschlossen ist.
9. Alle Cliettes werden freigegeben.

Wenn eine Cliette in der Initialisierungsdatei angegeben ist, Connection Manager jedoch nicht aktiv ist, lädt Net.Data die DLL und verarbeitet das Makro. Wenn Sie eine API verwenden, empfangen Sie wahrscheinlich Fehler, und Sie müssen Connection Manager starten.

Net.Data-Caching

Durch Caching werden die Antwortzeiten für den Anwendungsbenutzer verbessert. Net.Data speichert Ergebnisse einer Anforderung an den Webserver lokal, bis die Informationen aktualisiert werden, damit sie rasch abgerufen werden können. In diesem Kapitel werden die Konzepte, Funktionen und Einschränkungen vom Net.Data-Caching beschrieben.

- „Das Cache-Protokoll" auf Seite 230

Informationen zum Caching von Webseiten

Viele Softwarekomponenten führen Caching für Webanwendungen aus. Es folgen einige Beispiele von Caching-Anwendungen:

- Ein Web-Browser sichert Webseiten und zugehörige Objekte wie Abbilder und Audiodateien sowie Java-Applets lokal im Hauptspeicher oder auf Platte, um Netzwerkzeit zu verringern, wenn der Benutzer wiederholt auf die gleichen Seiten zugreift.
- Ein Web-Proxyserver-Cache sichert Webseiten und zugehörige Objekte auf einem lokalen Server in der Nähe einer Benutzergruppe, um die Netzwerkzugriffszeit auf ferne Webserver zu verringern. Dadurch kann zum Beispiel die Anzahl der Abrufe angeforderter Objekte durch die Webserver gesenkt werden. Ein Web-Proxyserver-Cache ermöglicht zudem, dass von mehreren Benutzern häufig aufgerufene Seiten effektiver benutzt werden können.
- Ein Webserver speichert häufig abgerufene Seiten und zugehörige Objekte im Hauptspeicher zwischen, um die Plattenzugriffszeit zu verringern, wenn Benutzer wiederholt die gleichen Seiten abrufen.
- Ein Datenbankverwaltungssystem speichert Datenelemente, die in der Regel auf Platte gespeichert sind, im Hauptspeicher zwischen, um die Plattenzugriffszeit zu verringern, wenn Benutzer wiederholt die gleichen Datenelemente abrufen.

Alle diese Komponenten führen Ihr Caching unabhängig voneinander aus, das Gesamtergebnis sind jedoch verbesserte Antwortzeiten für Benutzer. Die Webkomponenten (Browser, Proxyserver und Webserver) berücksichtigen in der Regel beim Ermitteln, wann ein zwischengespeichertes Element aktualisiert werden muss, verschiedene Optionen, wie zum Beispiel:

- Die Browser- und Server-Konfigurationsoptionen
- Den Inhalt der von den Webseiten zurückgegebenen HTTP-Kopfzeilen und zugehörige Informationen vom Webserver, vor allem zum Ablaufdatum

Informationen zum Net.Data-Caching

Net.Data stellt eine eigene Caching-Funktion für häufig abgerufene Seiten und zugehörige, durch Net.Data-Makros generierte Datenelemente bereit. Durch Bereitstellen einer Seite aus dem Net.Data-Cache wird die zur Ausführung eines Net.Data-Makros und zum Zugriff auf eine Datenbank erforderliche Zeit bei der Erstellung der Seite verringert.

Sie können einen Cache-Manager pro Server verwenden. **Empfehlung:** Verwenden Sie einen Cache-Manager für viele Exemplare von Net.Data und mehrere Caches pro Cache-Manager.

Abb. 26 zeigt, dass Net.Data einen Cache-Manager zum Verwalten des Caching der HTML-Ausgabe von einem Makro verwendet. Zu dieser Ausgabe können Daten aus einer Datenbank gehören.

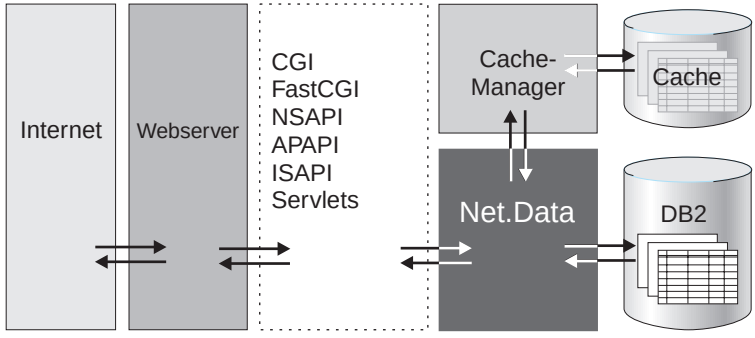


Abbildung 26. Net.Data-Caching

Terminologie für Net.Data-Caching

In der Net.Data-Dokumentation werden die folgenden Begriffe zum Beschreiben von Net.Data-Caching verwendet.

Cache Eine Art von Speicher, der die Daten enthält, auf die zuletzt zugegriffen wurde. Damit wird der nachfolgende Zugriff auf die gleichen Daten beschleunigt. Im Cache wird oft eine lokale Kopie der häufig verwendeten Daten gespeichert, auf die über ein Netzwerk zugegriffen werden kann. In Net.Data ist der Cache der lokale Hauptspeicher, der von Net.Data generierte HTML-Webseiten zur Wiederverwendung durch das Net.Data-Makro enthält. Da die Seiten im Cache zwischengespeichert werden, muss Net.Data die Informationen im Cache nicht erneut generieren. Jeder Cache wird durch den Cache-Manager verwaltet, der für mehrere Caches verwendet werden kann und zudem mehrere Exemplare von Net.Data bedienen kann.

Cache-ID
Eine Zeichenfolge, die einen bestimmten Cache angibt.

Cache-Manager
Das Programm, das Caching für eine Maschine verwaltet. Es kann mehrere Caches verwalten.

Konfigurationsdatei für Cache-Manager

Die Datei, die die von Net.Data verwendeten Einstellungen zum Ermitteln der Einstellungen für Protokollieren, Ablaufverfolgung, Cache-Größe und andere Optionen enthält. Sie enthält Einstellungen für einen Cache-Manager und alle von einem bestimmten Cache-Manager verwalteten Cache-Dateien. Der Name der mit Net.Data gelieferten Datei lautet `cachemgr.cnf`.

Konzepte des Net.Data-Caching

Je nachdem, wie viele HTTP-Server sich auf Ihrem System befinden und ob jeder HTTP-Server (unter Verwendung separater Net.Data-Konfigurationsdateien) eine eigene Kopie von Net.Data ausführt, können alle Kopien von Net.Data einem bzw. mehreren Cache-Managern zugeordnet sein. Ein Cache-Manager kann eine Anzahl von Caches im Hauptspeicher unterstützen, wobei jeder Cache über eine Cache-Kennung, die sogenannte *Cache-ID*, verfügt. Abb. 27 zeigt einen Cache-Manager, der mit mehreren Makros arbeitet und zwei Caches verwaltet.

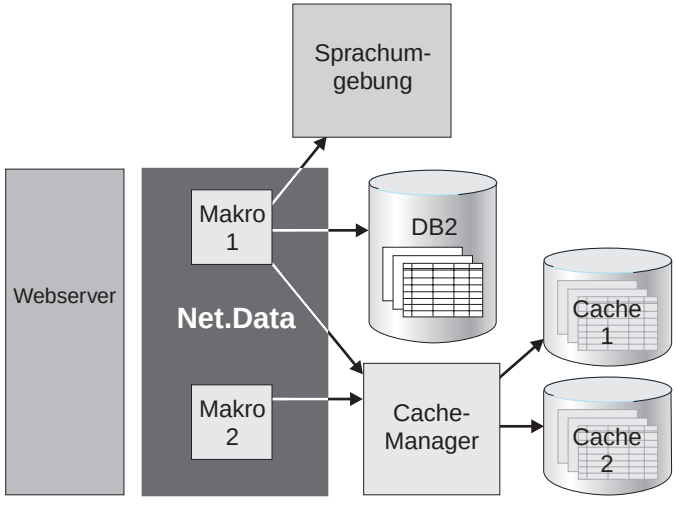


Abbildung 27. Cache-Manager arbeitet mit mehreren Makros und Caches

Sie können eine beliebige Anzahl Elemente, sogenannte *zwischengespeicherte Seiten*, in einen Cache stellen. Jede zwischengespeicherte Seite weist eine eindeutige Kennung auf, zum Beispiel eine URL-Adresse (URL - Uniform Resource Locator). Eine Seite ist ein Teil einer oder eine vollständige HTML-Seite.

Empfängt Net.Data eine Anforderung nach zwischengespeicherten Daten (zum Beispiel von der integrierten Funktion DTW_CACHE_PAGE), werden die folgenden Schritte ausgeführt:

1. Net.Data stellt die Verbindung zum Cache-Manager her.
2. Net.Data prüft, ob sich die Daten im Cache befinden.
   - Sind die Daten vorhanden und gültig, fordert Net.Data die Seiten vom Cache-Manager an, sendet diese an den Browser und beendet die Ausführung des Makros.
   - Befinden sich die Daten nicht im Cache, fährt Net.Data mit der Verarbeitung des Makros fort, sendet anschließend die generierte HTML-Seite an den Web-Browser und an den Cache-Manager, wo sie zwischengespeichert wird.
3. Net.Data trennt die Verbindung zum Cache-Manager.

Der Cache-Manager speichert die HTML-Ausgabe zwischen, wenn das Makro die Verarbeitung erfolgreich abgeschlossen hat. Dadurch wird sichergestellt, dass nur erfolgreich generierte Webseiten zwischengespeichert werden. Die Daten werden erst dann zwischengespeichert, nachdem sie an den Browser gesendet wurden. Die Daten, die der Benutzer dann sieht, sind diese zwischengespeicherten Daten.

Wenn Net.Data einen Fehler feststellt oder das Makro vorzeitig beendet wird, führt der Cache-Manager folgende Funktionen aus:

- Zurückweisen von Teilseiten oder fehlerhaften Seiten
- Aufbewahren vorhandener Seiten im Cache

Einschränkungen des Net.Data-Caching

Net.Data-Caching hat die folgenden Einschränkungen:

Sicherheit
: Der Cache-Manager bietet keine Sicherheitsfunktion. Wenn z. B. ein Datenbankbenutzer ein Makro ausführt und eine Seite mit Datenbankergebnissen zwischenspeichert, kann ein anderer Datenbankbenutzer die zwischengespeicherte Seite abrufen.

Direktanforderung
: Ein Direktanforderungsaufruf von Net.Data kann Net.Data-Caching nicht verwenden.

Schnittstellen des Net.Data-Caching

Net.Data stellt eine Gruppe flexibler Schnittstellen bereit, mit denen Sie das Caching für Ihre Anwendung konfigurieren und definieren können. Die verschiedenen Optionen zur Verwendung der Net.Data-Caching-Funktionen werden in Tabelle 14 erläutert. Außerdem finden Sie Informationen dazu, wo diese Funktionen beschrieben werden.

Tabelle 14. Net.Data-Cache-Schnittstellen

Schnittstelle	Beschreibung	Weitere Informationen
Konfigurationsoptionen für den Cache-Manager	Sie können in der Zeilengruppe für den Cache-Manager in der Konfigurationsdatei für den Cache-Manager eine Anzahl von Optionen wie Protokollieren und Ablaufverfolgung angeben.	„Definieren des Cache-Managers" auf Seite 213
Cache-Konfigurationsoptionen	In einem einzigen Exemplar des Net.Data-Cache-Managers können Sie eine Anzahl von Caches zum Aufbewahren der zwischengespeicherten Elemente definieren. Jeder Cache verfügt über eine eigene Gruppe von Kenndaten, wie Größe und Speicherposition sowie die Cache-ID. Diese Kenndaten werden in der Zeilengruppe für den Cache in der Konfigurationsdatei für den Cache-Manager definiert. Jede Zeilengruppe wird durch die Cache-ID angegeben.	„Definieren eines Cache" auf Seite 215
Net.Data-Initialisierungsoptionen	Wenn Net.Data und der entsprechende Cache-Manager auf separaten Systemen ausgeführt werden, dann geben Sie das Cache-Manager-System und die Anschlussnummer in der Net.Data-Initialisierungsdatei an.	„Konfigurationsvariablen für den Cache-Manager" auf Seite 15

Tabelle 14. Net.Data-Cache-Schnittstellen (Forts.)

Schnittstelle	Beschreibung	Weitere Informationen
Integrierte Net.Data-Cache-Funktionen	Sie können den Inhalt eines Net.Data-Cache mit den integrierten Net.Data-Cache-Funktionen bearbeiten. Geben Sie die Cache-ID in der entsprechenden Makrofunktion an, um den Cache mit den geeignetsten Kenndaten auszuwählen.	Siehe das Kapitel zu integrierten Funktionen im Handbuch *Net.Data Reference*.

Planen für den Cache-Manager

Berücksichtigen Sie beim Planen der Verwendung von Net.Data-Cache-Funktionen Folgendes:

- Die vom Caching profitierenden Seiten und die erzielten Leistungsverbesserungen
- Zeitpunkt der Zwischenspeicherung der Elemente
- Zeitpunkt der Aktualisierung der Elemente im Cache und Wahl der Aktualisierungsmethoden

Führen Sie die folgenden Schritte aus, um Net.Data-Caching zu verwenden. Dazu müssen Sie wissen, wie Sie das Caching einsetzen wollen.

Empfehlung: Es wird dringend empfohlen, vor dem Einsatz einer wichtigen Anwendung, die Caching verwendet, Ihre Anwendung zu planen und mit Hilfe eines Prototyps zu testen.

- Installieren Sie Net.Data und die zugehörige Caching-Funktion.
- Konfigurieren Sie den Cache-Manager. Siehe „Konfigurieren des Cache-Managers und der Net.Data-Caches" auf Seite 212.
- Legen Sie fest, wie Sie die Net.Data-Anwendung einsetzen wollen.
- Überprüfen Sie die verschiedenen Net.Data-Cache-Protokolle, um zu ermitteln, ob Verbesserungen in der Verwendung des Cache und seiner Konfiguration vorgenommen werden müssen.

Cache-Fehler

Der Cache-Manager speichert keine Webseiten zwischen, wenn Net.Data einen internen Fehler feststellt, durch den das Makro vor Beendigung der Verarbeitung beendet wird. Der Cache-Manager speichert keine Seiten zwischen, die unvollständig sind oder Net.Data-Fehler enthalten. Zu diesen Fehlerarten gehören Makrosyntaxfehler und SQL-Fehler.

Fehlerhafte Seiten werden in folgenden Fällen zwischengespeichert:

- Net.Data stellt einen Fehler fest, muss das Makro jedoch aufgrund einer Konfigurationsvariablen CONTINUE in einem Nachrichtenblock weiter ausführen und wird normal beendet.
- Außerhalb des Net.Data-Fehlerbestimmungsbereichs treten Fehler auf, zum Beispiel während der ROLLBACK-Operation einer Datenbank.

Cache-Kennungen

Sie müssen zwei Arten von Kennungen einplanen, wenn Sie Caching für Ihre Anwendung entwerfen.

- **Kennung für einen Cache:** Diese Kennung ist die Cache-ID und gibt den Namen in der Zeilengruppe der Konfigurationsdatei an, der den Cache definiert. Sie können verschiedene Vorgehensweisen zum Klassifizieren und Benennen Ihrer Caches verwenden. Sie könnten den Cache zum Beispiel nach der Anwendung benennen. Sie können einen Cache für jede Ihrer Net.Data-Anwendungen festlegen und jedem Cache einen Namen geben, der vom zugehörigen Net.Data-Makro abgeleitet ist.
- **Kennung für die zwischengespeicherte Seite:** Diese Kennung ist die ID der zwischengespeicherten Seite und gibt den Namen der zwischenzuspeichernden Seite an. Die ID der zwischengespeicherten Seite kann eine beliebige Zeichenfolge, wie zum Beispiel eine URL-Adresse, sein. Diese Kennung wird mit der integrierten Funktion DTW_CACHE_PAGE() angegeben. Informationen zur Syntax und Beispiele finden Sie im Kapitel zu den integrierten Funktionen im Handbuch *Net.Data Reference*.

Konfigurieren des Cache-Managers und der Net.Data-Caches

Der Cache-Manager verwaltet mindestens einen Cache in Ihrem System. Jeder dieser Caches weist den Inhalt dynamisch erstellter HTML-Seiten auf. Konfigurieren Sie den Cache-Manager und die einzelnen Caches, indem Sie die Schlüsselwortwerte in der Konfigurationsdatei für den Cache-Manager `cachemgr.cnf` aktualisieren.

Die Konfigurationsdatei für den Cache-Manager enthält zwei Arten von Zeilengruppen, und zwar die Zeilengruppe für den Cache-Manager und die Zeilengruppe für die Cache-Definition. In den folgenden Schritten wird beschrieben, wie Sie diese beiden Arten von Zeilengruppen für Ihre Anwendung anpassen.

Definieren des Cache-Managers

Definieren Sie die Zeilengruppe für den Cache-Manager, indem Sie Werte für die zulässigen Schlüsselwörter angeben. Alle Schlüsselwörter sind optional, d. h., Sie brauchen sie nur anzugeben, wenn Sie den Standardwert nicht übernehmen wollen.

Gehen Sie wie folgt vor, um den Cache-Manager zu definieren:

1. Geben Sie den Namen der Protokolldatei für den Cache-Manager an. Das Protokoll zeigt die Aktivität aller Transaktionen für alle Caches an und wird zur Fehlerbehebung und Problemanalyse bereitgestellt.

   Nachrichten werden standardmäßig über die Konsole angezeigt.

   Syntax:

   `log=`*`path`*

   Dabei ist *path* der Pfad und Dateiname der Cache-Datei.

   Hinweis: Geben Sie eine Protokolldatei für jeden einzelnen Cache mit dem Schlüsselwort **tran-log** aus der Zeilengruppe für die Cache-Definition an.

2. Geben Sie die TCP/IP-Anschlussnummer an, die vom Cache-Manager für eingehende Anforderungen verwendet wird. Diese Anschlussnummer wird nur für das Ansteuern des Cache-Managers von einer fernen Maschine aus verwendet.

   Dieser Wert muss mit der Anschlussnummer übereinstimmen, die durch die Konfigurationsvariable DTW_CACHE_PORT in der Net.Data-Initialisierungsdatei angegebenen ist. Der Standardwert wird auf folgende Art ermittelt:

   a. Der Cache-Manager überprüft den Pfad */etc/services* auf den Wert, der dem Namen `ibm-cachemgrd` zugeordnet ist. Wenn dieser Wert gefunden wird, verwendet der Cache-Manager den Wert. Wenn er nicht gefunden wird, verwendet er die nächste Methode.

   b. Der Cache-Manager verwendet den Standardanschluss 7175.

   Syntax:

   `port=`*`port_number`*

   Dabei ist *port_number* eine eindeutige TCP/IP-Anschlussnummer.

3. Geben Sie die maximale Zeitspanne in Sekunden an, die der Cache-Manager einen anstehenden Lesevorgang aktiv lassen soll. Wenn diese Dauer überschritten wird, beendet der Cache-Manager die Verbindung.

   Der Standardwert ist 30 Sekunden.

   Syntax:

   `connection-timeout=`*`seconds`*

Dabei ist *seconds* die Anzahl der Sekunden, die ein anstehender Lesevorgang aktiv sein soll.

4. Geben Sie an, ob Nachrichten protokolliert werden sollen.

   Der Standardwert ist **no** bzw. **off**.

   Syntax:

   ```
   logging=yes|on|no|off
   ```

   Dabei gilt Folgendes:

   yes | on
   Gibt an, dass Protokollierung erforderlich ist.

   no | off
   Gibt an, dass keine Protokollierung ausgeführt werden soll.

5. Geben Sie an, ob Protokollumlauf verwendet werden soll.

   Der Standardwert ist **no**. Wenn **yes** angegeben wird, wird das aktuelle Protokoll geschlossen, wenn die maximale Größe erreicht wird (siehe **log-size** weiter unten), der Datei wird der Dateityp .old zugewiesen, und es wird ein neues Protokoll geöffnet. Es wird nur eine Generation der Protokolldatei verwaltet (vorhandene OLD-Dateien werden überschrieben).

   Syntax:

   ```
   wrap-log=yes|no
   ```

   Dabei gilt Folgendes:

   yes Gibt an, dass Protokollumlauf verwendet werden soll.

   no Gibt an, dass Protokollumlauf nicht verwendet werden soll.

6. Geben Sie die maximale Größe in Byte an, bis zu der ein Protokoll anwachsen kann, wenn Protokollumlauf aktiviert ist.

   Der Standardwert ist 64000.

   Syntax:

   ```
   log-size=bytes
   ```

   Dabei ist *bytes* die Anzahl Byte der maximalen Größe.

7. Geben Sie die Stufe der Nachrichten an, die in das Protokoll geschrieben werden sollen. Diese Werte werden aktiviert, wenn sie in die Liste *trace_flag_definitions* aufgenommen werden. Für sie gibt es keine Einstellungen.

   Standardmäßig werden nur die Nachrichten beim Start und Stopp des Cache-Managers protokolliert.

   Syntax:

   ```
   trace-flags=trace_flag_definitions
   ```

Dabei gilt Folgendes:

D_ALL
Aktiviert alle Ablaufverfolgungsmarkierungen.

D_NONE
Inaktiviert alle Ablaufverfolgungsmarkierungen.

Beispiel: Ablaufverfolgungsmarkierung, die die Aktivierung aller Ablaufverfolgungsmarkierungen angibt:

```
trace-flags=D_ALL
```

Zeilengruppenbeispiel: Eine gültige Zeilengruppe des Konfigurationsmanagers:

```
cache-manager {
port = 7175
connection-timeout = 60
logging = off
log = /local/netdata/cachemgr/logs/cachemgr.log
wrap-log = yes
log-size = 32KB
}
```

Definieren eines Cache

Definieren Sie die Zeilengruppe für die Cache-Definition, indem Sie Werte für die zulässigen Schlüsselwörter angeben. Die meisten Schlüsselwörter sind optional und müssen nur angegeben werden, wenn Sie den Standardwert nicht verwenden wollen.

Gehen Sie wie folgt vor, um einen Cache zu definieren:

1. Geben Sie den Namen des Pfads und Verzeichnisses an, in dem Cache-Seiten gespeichert werden sollen. Beim Systemstart muss das Dateisystem mit diesem Verzeichnis mindestens so groß wie der Wert von **fssize** sein (siehe weiter unten). Ansonsten wird der Cache nicht gestartet. Dieser Wert kann als ein absoluter Pfadname oder als ein relativer Pfadname angegeben werden, der dem Pfad entspricht, in dem der Cache-Manager gestartet wurde.

   Erforderlich.

   Syntax:

   ```
   root=path_name
   ```

   Dabei gilt Folgendes:

   path_name
   Ist der absolute oder relative Name des Pfads und Verzeichnisses, in dem die Cache-Seiten gespeichert werden.

2. Geben Sie an, ob der aktuelle Cache aktiv sein soll, wenn der Cache-Manager gestartet wird.

   Nicht erforderlich. Der Standardwert ist **yes**. Wenn **no** gesetzt wird, wird der Cache im Cache-Manager definiert, jedoch nicht aktiviert. Er kann später mit dem Befehl **cacheadm** aktiviert werden.

   Syntax:

   ```
   caching=yes|no
   ```

   Dabei gilt Folgendes:

   yes Gibt an, dass der Cache aktiv sein soll, wenn der Cache-Manager gestartet wird.

   no Gibt an, dass der Cache nicht aktiv sein soll, wenn der Cache-Manager gestartet wird.

3. Geben Sie den maximalen Speicherbereich an, der im Dateisystem durch Seiten im aktuellen Cache belegt werden soll. Wenn der maximale Speicherbereich überschritten wird, löscht der Cache-Manager angefangen bei der ältesten Seite genügend Seiten, um den vom Cache belegten Gesamtspeicherbereich entsprechend zu begrenzen. Sie können das automatische Löschen von Einträgen inaktivieren, indem Sie diesen Wert erhöhen. Wenn jedoch der physische Dateisystemspeicherbereich überschritten wird, schlagen Versuche, dem Cache neue Seiten hinzuzufügen, fehl.

   Nicht erforderlich. Der Standardwert ist 0 (kein Caching auf dem Datenträger).

   Syntax:

   ```
   fssize=nnB|nnKB|nnM
   ```

   Dabei gilt Folgendes:

   *nn***B** Ist die Anzahl Byte, zum Beispiel 5000B.

   *nn***KB** Ist die Anzahl Kilobyte, zum Beispiel 640KB.

   *nn***MB** Ist die Anzahl Megabyte, zum Beispiel 30MB.

4. Geben Sie den maximalen Speicherbereich an, der von allen Seiten in diesem Cache belegt werden soll. Wenn der maximale Speicherbereich überschritten wird, löscht der Cache-Manager angefangen bei der ältesten Seite genügend Seiten, um den vom Cache belegten Gesamtspeicherbereich entsprechend zu begrenzen. Sie können das automatische Löschen von Seiten inaktivieren, indem Sie diesen Wert erhöhen. Wenn jedoch der Prozess **cachemgrd** zu viel Hauptspeicher in Anspruch nimmt, wird er eventuell vom Betriebssystem beendet.

   Nicht erforderlich. Der Standardwert ist 1MB.

Syntax:

```
mem-size=nnB|nnKB|nnMB
```

Dabei gilt Folgendes:

*nn***B** Ist die Anzahl Byte, zum Beispiel 5000B.

*nn***KB** Ist die Anzahl Kilobyte, zum Beispiel 640KB.

*nn***MB** Ist die Anzahl Megabyte, zum Beispiel 30MB.

5. Geben Sie an, wie lange eine Seite maximal im Cache gespeichert werden soll. Wenn dieser Wert überschritten wird, markiert der Cache-Manager die Seite als abgelaufen, löscht sie jedoch nicht, außer wenn die Grenzwerte für **fssize** (bei Zwischenspeicherung auf Platte) bzw. **memsize** (bei Zwischenspeicherung im Hauptspeicher) erreicht werden. Der Cache-Manager löscht als abgelaufen markierte Seiten vor allen anderen Seiten, wenn die Grenzwerte für **memsize** bzw. **fssize** erreicht werden. Sie können die Überprüfung der Gültigkeitsdauer (**lifetime**) mit dem Schlüsselwort **check_expiration** inaktivieren.

   Erforderlich: Nein. Der Standardwert ist 5 Minuten.

   Syntax:

   ```
   lifetime=time_length
   ```

   Dabei gilt Folgendes:

   *nn***S** Ist die Anzahl Sekunden, zum Beispiel 600S.

   *nn***M** Ist die Anzahl Minuten, zum Beispiel 20M.

   *nn***H** Ist die Anzahl Stunden, zum Beispiel 30H.

6. Geben Sie an, ob Cache-Seiten als abgelaufen markiert und eine Überprüfung der Gültigkeitsdauer ausgeführt werden soll.

   Nicht erforderlich. Der Standardwert ist **yes** mit einer Standardgültigkeitsdauer von 60 Sekunden. Dieser Wert kann auch auf eine Zeitspanne gesetzt werden, indem Sie den Wert **yes** angeben und eine maximale Zeitspanne für ein im Cache gespeichertes Element festlegen. Wenn **no** gesetzt wird, werden Cache-Seiten nie als abgelaufen markiert, und es wird keine Überprüfung der Gültigkeitsdauer ausgeführt.

   Syntax:

   ```
   check-expiration=yes|nnS|nnM|nnH|no
   ```

   Dabei gilt Folgendes:

   yes Gibt an, dass der Cache-Manager eine Überprüfung der Gültigkeitsdauer ausführt und dass Cache-Seiten als abgelaufen markiert werden.

   *nn***S** Ist die Anzahl Sekunden, zum Beispiel 600S.

   *nn***M** Ist die Anzahl Minuten, zum Beispiel 20M.

*nn***H** Ist die Anzahl Stunden, zum Beispiel 30H.

no Gibt an, dass der Cache-Manager keine Überprüfung der Gültigkeitsdauer ausführt und dass Cache-Seiten nicht als abgelaufen markiert werden.

7. Geben Sie den maximalen Speicherbereich an, den eine zwischengespeicherte Seite im Hauptspeicher-Cache belegen kann. Wenn eine Seite für den Hauptspeicher zu groß ist, wird der Datei-Cache überprüft. Wenn genügend Speicherbereich vorhanden ist, speichert der Cache-Manager die Cache-Seite im Datei-Cache. Wenn die Seite nicht in den Datei-Cache passt, schlägt der Caching-Versuch fehl. Wenn die Seite kleiner als der Wert für **datum_memory_limit (cacheobj-memory-limit)** ist, der Cache jedoch nicht über genügend Speicherbereich verfügt, werden die ältesten Cache-Seiten aus dem Hauptspeicher-Cache gelöscht, um Speicher für die neue Seite freizugeben.

   Nicht erforderlich. Der Standardwert ist 1KB.

   Syntax:

   ```
   datum-memory-limit (cacheobj-memory-limit)=nnB|nnKB|nnMB
   ```

   Dabei gilt Folgendes:

   *nn***B** Ist die Anzahl Byte, zum Beispiel 5000B.

   *nn***KB** Ist die Anzahl Kilobyte, zum Beispiel 640KB.

   *nn***MB** Ist die Anzahl Megabyte, zum Beispiel 30MB.

8. Geben Sie den maximalen Speicherbereich an, den eine zwischengespeicherte Seite im Datei-Cache belegen kann. Wenn die Seite kleiner als der Wert für **datum_disk_limit** ist, im Datei-Cache jedoch kein Speicherbereich frei ist, werden die ältesten Cache-Seiten aus dem Datei-Cache gelöscht, um Speicher für die neue Seite freizugeben.

   Nicht erforderlich. Der Standardwert ist 1KB.

   Syntax:

   ```
   datum-disk-limit (cacheobj-space-limit)=nnB|nnKB|nnMB
   ```

   Dabei gilt Folgendes:

   *nn***B** Ist die Anzahl Byte, zum Beispiel 5000B.

   *nn***KB** Ist die Anzahl Kilobyte, zum Beispiel 640KB.

   *nn***MB** Ist die Anzahl Megabyte, zum Beispiel 30MB.

9. Geben Sie die Zeitspanne zwischen der Erstellung von Datensätzen mit Statistikdaten an. Wenn 0 gesetzt wird, werden keine Datensätze mit Statistikdaten geschrieben.

   Nicht erforderlich. Der Standardwert ist 0 (keine Statistikdaten).

Syntax:

```
stat-interval = nnS|nnM|nnH
```

Dabei gilt Folgendes:

*nn***S** Ist die Anzahl Sekunden, zum Beispiel 600S.

*nn***M** Ist die Anzahl Minuten, zum Beispiel 1M.

*nn***H** Ist die Anzahl Stunden, zum Beispiel 3H.

10. Geben Sie den Namen des Pfads und der Datei an, die zum Protokollieren von Statistikdaten für den aktuellen Cache verwendet werden.

    Erforderlich, wenn der Wert für **stat-interval** größer als 0 ist.

    Syntax:

    ```
    stat-files=filename
    ```

    Dabei ist *filename* der Pfad und Name der Statistikprotokollierungsdatei.

11. Geben Sie an, ob Zähler für Statistikdaten bei jedem Schreibvorgang von Statistikdaten in die Protokolldatei auf 0 zurückgesetzt werden sollen.

    Nicht erforderlich. Der Standardwert ist **yes**.

    Syntax:

    ```
    reset-stat-counters=yes|no
    ```

    Dabei gilt Folgendes:

    yes Setzt die Zähler für Statistikdaten zurück.

    no Setzt die Zähler für Statistikdaten nicht zurück.

12. Definieren Sie den Pfad und Dateinamen zum Speichern des Transaktionsprotokolls für jeden Cache. Transaktionsprotokolldateien für den Cache unterscheiden sich von Protokolldateien des Cache-Managers, mit denen die Gesamtaktivität des Cache-Managers protokolliert wird.

    Erforderlich. Wenn dieses Schlüsselwort nicht angegeben wird, wird für den Cache kein Transaktionsprotokoll erstellt.

    Syntax:

    ```
    tran-log=filename
    ```

    Dabei ist *filename* der Pfad und Name der Transaktionsprotokolle für jeden Cache.

13. Geben Sie an, ob die Transaktionsprotokollierung für den Cache beim ersten Start des Cache-Managers aktiviert werden soll. Dieser Parameter wird ignoriert, außer wenn über den Parameter **tran-log** eine gültige Transaktionsprotokolldatei angegeben wird. Sie können die Transaktionsprotokollierung bei aktivem Cache-Manager-Dämon mit dem Befehl **cacheadm** aktivieren, wenn in der Konfigurationsdatei für den Cache-Manager ein gültiger Wert für **tran-log** angegeben wurde.

    Nicht erforderlich. Der Standardwert ist **no**.

    Syntax:

    ```
    tran-logging=yes|on|no|off
    ```

    Dabei gilt Folgendes:

    yes|on
    Gibt an, dass Protokollierung erforderlich ist.

    no|off
    Gibt an, dass keine Protokollierung ausgeführt werden soll.

14. Geben Sie an, ob Transaktionsprotokollumlauf verwendet werden soll.

    Nicht erforderlich. Der Standardwert ist **yes**. Wenn **yes** angegeben wird, wird das aktuelle Protokoll geschlossen, wenn die maximale Größe erreicht wird (siehe **tran-log-size**), der Datei wird der Dateityp .old zugewiesen, und es wird ein neues Protokoll geöffnet. Es wird nur eine Generation des Protokolls verwaltet (vorhandene OLD-Dateien werden überschrieben).

    Syntax:

    ```
    wrap-tran-log=yes|no
    ```

    Dabei gilt Folgendes:

    yes Gibt an, dass Protokollumlauf verwendet werden soll.

    no Gibt an, dass Protokollumlauf nicht verwendet werden soll.

15. Geben Sie die maximale Größe in Byte an, bis zu der ein Transaktionsprotokoll anwachsen kann, wenn **wrap-tran-log** angegeben ist.

    Nicht erforderlich. Der Standardwert ist 64000.

    Syntax:

    ```
    tran-log-size=bytes
    ```

    Dabei ist *bytes* die Anzahl Byte der maximalen Größe.

Zeilengruppenbeispiel: Eine gültige Zeilengruppe einer Cache-Definition für einen Cache:

```
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
}
```

Starten und Stoppen des Cache-Managers

In den folgenden Abschnitten wird beschrieben, wie Sie den Cache-Manager starten und stoppen können.

- „Starten des Cache-Managers"
- „Stoppen des Cache-Managers" auf Seite 222

Starten des Cache-Managers

Der Cache-Manager-Dämon wird mit dem Befehl `cachemgrd` gestartet.

Syntax:

Parameter:

cachemgrd
Das Befehlsschlüsselwort

config_file
Gibt den Namen der Datei an, in der der Cache-Manager und die einzelnen vom Cache-Manager verwalteten Caches definiert sind. Die mit Net.Data gelieferte Konfigurationsdatei ist `cachemgr.cnf`.

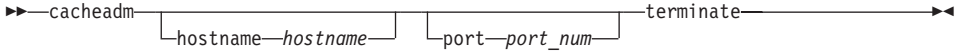
Beispiel:

```
cachemgrd -c myconfig.cfg
```

Stoppen des Cache-Managers

Der Cache-Manager wird mit dem Befehl `cacheadm` gestoppt.

Syntax:



Parameter:

cacheadm
Das Befehlsschlüsselwort

hostname
Gibt den Namen der Maschine an, auf der der Cache aktiv ist, wenn er sich von der Maschine unterscheidet, auf der der Befehl `cacheadm` abgesetzt wird.

port_num
Gibt die Cache-Anschlussnummer an, wenn sich die Nummer vom Standardwert (7175) unterscheidet.

terminate
Gibt an, dass der Cache-Manager gestoppt werden soll.

Beispiel:

```
cacheadm hostname host1 port 7178 terminate
```

Caching von Webseiten

Sie können eine Webseite mit der integrierten Funktion DTW_CACHE_PAGE zwischenspeichern. Wenn Net.Data die Funktion DTW_CACHE_PAGE im Makro erkennt, steuert es den Cache-Manager an und fängt mit dem Sichern der HTML-Ausgabe für das Makro im Hauptspeicher an. Nach der erfolgreichen Verarbeitung eines Makros durch Net.Data wird die HTML-Ausgabe an den Browser gesendet, und der Cache-Manager speichert die Ausgabe in einer Transaktion, wie in Abb. 28 gezeigt.

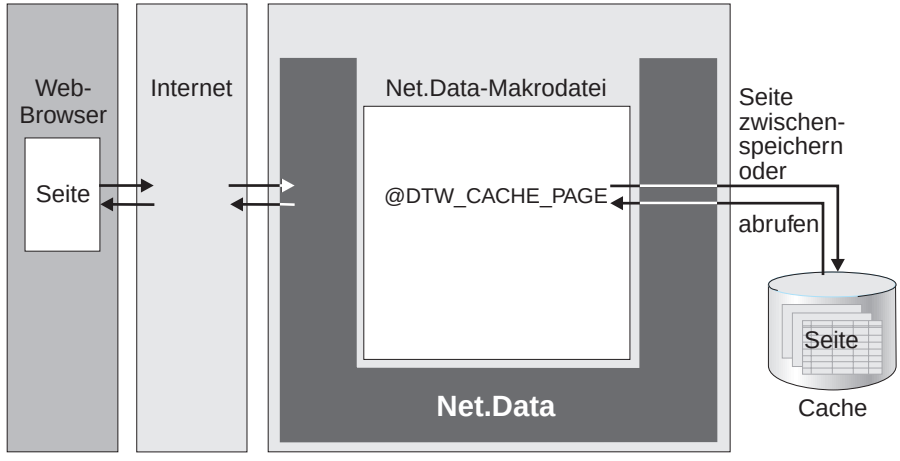


Abbildung 28. Funktion DTW_CACHE_PAGE leitet Caching ein

Caching einer Seite

Geben Sie an, dass von Net.Data generierte Seiten mit der integrierten Net.Data-Funktion DTW_CACHE_PAGE() in den Cache geschrieben werden sollen.

Die Funktion DTW_CACHE_PAGE() speichert die gesamte Ausgabe des Makros nach der Funktionsanweisung zwischen, nachdem sie ermittelt hat, dass die Seite nicht bereits im Cache vorhanden oder abgelaufen ist. Wenn die Seite nicht im Cache vorhanden oder älter als die angegebene Gültigkeitsdauer ist, sendet Net.Data die Ausgabe an den Browser zurück, generiert neue Ausgabeseiten von der Makroausführung und speichert die Seite im Cache.

Wenn der Cache-Manager die zwischengespeicherte Seite findet und sie weiterhin gültig ist, zeigt er den Cache-Inhalt an, und Net.Data verlässt das Makro. Durch dieses Verhalten wird sichergestellt, dass keine unnötige Verarbeitung ausgeführt wird, nachdem die Webseite aus dem Cache abgerufen wurde.

Hinweis zur Leistung: Stellen Sie DTW_CACHE_PAGE() als die erste bzw. eine der ersten Anweisungen in das Makro, um den Aufwand bei der Ausführung des Makros zu minimieren.

Gehen Sie wie folgt vor, um eine Seite zwischenzuspeichern:

1. Fügen Sie im HTML- oder XML-Block eines Makros vor der HTML-Codierung die folgende Funktionsanweisung ein:

   ```
   @DTW_CACHE_PAGE("cache_id", cache_page_id, "age", status)
   ```

   Geben Sie mit dieser Funktion an, dass Net.Data die gesamte HTML-Ausgabe vom Makro nach dieser Anweisung zwischenspeichern soll. Platzieren Sie diese Anweisung an den Anfang im Makro, wenn Sie die gesamte HTML-Ausgabe zwischenspeichern wollen.

   Parameter:

   cache_id
   : Eine Zeichenfolge, die den Cache angibt, in dem die Seite gespeichert wird. Sie können Cache-IDs Makros bzw. Makrogruppen zuordnen.

   cache_page_id
   : Eine Zeichenfolge mit einer Kennung, mit der die zwischengespeicherte Seite in einer nachfolgenden Cache-Anforderung über @DTW_CACHE_PAGE lokalisiert wird, zum Beispiel die URL-Adresse der Seite.

   age
   : Eine Zeichenfolgevariable mit einer Zeitspanne in Sekunden, die angibt, wann eine Seite als abgelaufen betrachtet wird. Wenn sich die angeforderte Seite länger im Cache befindet als der Wert von *age* angibt, führt Net.Data das Makro aus, generiert die Seite erneut und speichert die generierte Seite zwischen, wodurch die abgelaufene Seite ersetzt wird. Wenn sich die angeforderte Seite kürzer oder genau so lang im Cache befindet wie der Wert von *age* angibt, ruft Net.Data die Seite aus dem Cache ab und sendet sie an den Browser. In diesem Fall beendet Net.Data die Makroausführung sofort.

   status
   : Eine von Net.Data zurückgegebene Zeichenfolgevariable, die angibt, ob die Seite erfolgreich zwischengespeichert wurde oder nicht.

Beispiel:

```
%HTML(cache_example) {
 %IF (customer == "Joe Smith")
 @DTW_CACHE_PAGE("mymacro.dtw", "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>
 %}
```

Erweitertes Caching: Dynamisches Ermitteln der Notwendigkeit zur Zwischenspeicherung

Die Funktion DTW_CACHE_PAGE() leitet Caching von ihrer Position im Makro an ein. In der Regel stellen Sie die Funktion an den Anfang des Makros, um die Leistung zu steigern und sicherzustellen, dass die gesamte HTML-Ausgabe zwischengespeichert wird. Bei Anwendungen mit erweitertem Caching können Sie die Funktion DTW_CACHE_PAGE() in die HTML-Ausgabeabschnitte stellen, wenn Sie festlegen müssen, dass das Zwischenspeichern zu einem bestimmten Zeitpunkt während der Verarbeitung erfolgen soll anstatt am Anfang des Makros. Diese Entscheidung kann beispielsweise davon abhängig sein, wie viele Zeilen von einer Abfrage oder von einem Funktionsaufruf zurückgegeben werden.

Beispiel: Platzieren der Funktion im HTML- oder XML-Block, weil die Entscheidung, Daten zwischenzuspeichern, von der erwarteten Größe der HTML-Ausgabe abhängt

```
% 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()
 %IF (ALL_ROWS > "100")
 @DTW_CACHE_PAGE("mymacro.dtw", "http://www.mypage.org", "-1", status)
%ENDIF

@all_customers()
</body>
</html>
%}
```

In diesem Beispiel wird die Seite basierend auf der erwarteten Größe der HTML-Ausgabe zwischengespeichert bzw. abgerufen. HTML-Ausgabeseiten werden nur dann zwischengespeichert, wenn die Datenbanktabelle mehr als 100 Zeilen enthält. Net.Data sendet den Text im OUTPUT-Block, also `This is the customer list`, nach der Ausführung des Makros immer an den Browser. Der Text wird nie zwischengespeichert. Die Zeilen nach dem Funktionsaufruf `@count_rows()` werden zwischengespeichert bzw. abgerufen, wenn die Bedingungen des IF-Blocks erfüllt sind. Beide Abschnitte zusammen bilden eine vollständige Net.Data-Ausgabeseite.

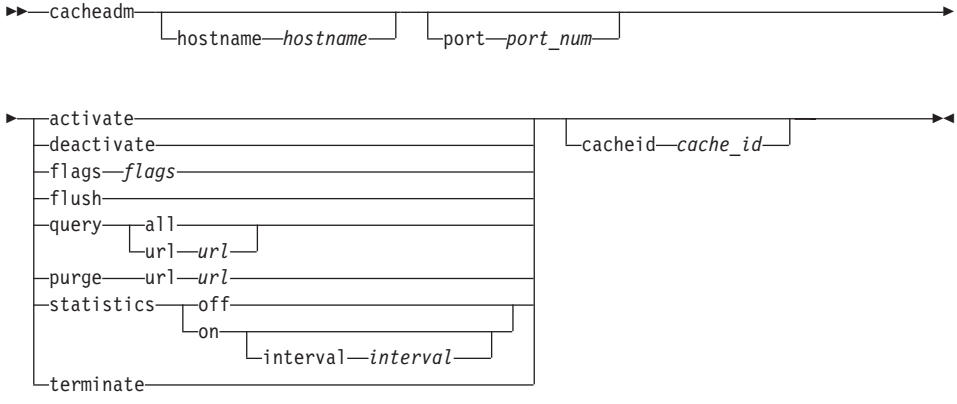
Der Befehl CACHEADM

Verwenden Sie den Befehl CACHEADM für die folgenden Aufgaben:

- Stoppen des Cache-Managers
- Leeren eines bestimmten Cache
- Abfragen eines bestimmten Cache
- Aktivieren bzw. Inaktivieren der Protokollierung
- Protokollierungsmarkierungen
- Starten und Stoppen der Sammlung von Statistikdaten

Alle Parameter können auf die kleinste eindeutige Zeichengruppe gekürzt werden.

Syntax:



Parameter:

activate
Aktiviert einen angegebenen Cache. Wenn der Cache bereits aktiv ist, führt der Cache-Manager keine Aktionen aus.

cache_id
Eine Zeichenfolgevariable, die den Cache angibt, in dem die Seite gespeichert ist, zum Beispiel: `cache1`.

deactivate
Inaktiviert einen angegebenen Cache. Wenn der Cache bereits inaktiv ist, führt der Cache-Manager keine Aktionen aus. Alle anstehenden Operationen werden beendet, und es werden keine neuen akzeptiert. Wenn die letzte Operation beendet ist, markiert der Cache-Manager den Cache als inaktiv.

flags
Gibt an, ob die aufgelisteten Markierungen ein- oder ausgeschaltet werden sollen.

D_ALL
Schaltet alle Ablaufverfolgungsmarkierungen ein.

D_NONE
Schaltet alle Ablaufverfolgungsmarkierungen aus.

flush
Leert einen Cache, der durch den Parameter *cache_id* angegeben wird, der für diesen Parameter erforderlich ist. Dieser Parameter löscht bedingungslos alle Elemente aus dem angegebenen Cache.

hostname
Gibt den Namen der Maschine an, auf der der Cache aktiv ist, wenn er sich von der Maschine unterscheidet, auf der der Befehl `cacheadm` abgesetzt wird. Zum Beispiel: `myhost`.

port_num
Gibt die Cache-Anschlussnummer an, wenn sich die Nummer vom Standardwert (7175) unterscheidet. Diese Nummer muss im System eindeutig sein.

purge
Gibt eine bestimmte, aus dem Cache zu löschende Seite an. Wenn *url* angegeben wird, löscht der Cache-Manager die Seite mit einem mit *url* übereinstimmenden Schlüssel. Wenn eine Abhängigkeit definiert ist, löscht der Cache-Manager alle Elemente mit der zugehörigen Abhängigkeit und schreibt ihre Schlüssel in den standardmäßig verwendeten Ausgabedatenstrom `stdout`.

query
Gibt je nach den angegebenen Parametern die folgenden Caching-Informationen zurück:

- Gibt Informationen zu einem Cache zurück, wenn nur die Cache-ID angegeben wird.

- Gibt Informationen zu einer bestimmten zwischengespeicherten Seite zurück, wenn *url* angegeben wird.
- Gibt Informationen zu allen Seiten zurück, wenn `all` angegeben wird.

  Andere Programme verwenden die Option `all` zum Formatieren oder Interpretieren der Ergebnisse. Jede Zeile enthält die folgenden Informationen:

  – Seitenschlüssel
  – Seitenalter
  – Seitenlänge
  – Seitenerstellungsdatum
  – Seitenablaufdatum
  – Datum des letzten Verweises auf die Seite

  Alle Daten liegen im ganzzahligen Standardzeitformat von UNIX vor.

  Hinweis zur Leistung: Die Option `cache query all` kann die Leistung beeinträchtigen und sollte daher mit Bedacht eingesetzt werden.

statistics
Aktiviert bzw. inaktiviert die Protokollierung für die Sammlung von Statistikdaten für einen bestimmten Cache und erfordert den Parameter *cache_id*. Wenn mit dem auf `on` gesetzten Parameter `statistics` ein Intervall angegeben wird, setzt Net.Data das Intervall zwischen Aktualisierungen erstmals oder erneut auf die angegebene Anzahl Sekunden.

terminate
Gibt an, dass der Cache-Manager gestoppt werden soll.

tranlogging
Aktiviert bzw. inaktiviert die Transaktionsprotokollierung für einen bestimmten Cache und erfordert den Parameter *cache_id*. Dieser Parameter wird nur dann wirksam, wenn in der Konfigurationsdatei für den Cache-Manager über den Parameter `tran-log` ein gültiges Transaktionsprotokoll für den Cache angegeben wird.

url Die URL-Adresse (URL - Universal Relative Location), die die Speicherposition der Datei auf dem Webserver angibt, zum Beispiel: `http://www.ibm.com/mydir/page1`.

Das Cache-Protokoll

Mehrere Arten von Statistikdaten für interne Operationen werden aufbewahrt und optional in das Cache-Protokoll geschrieben. Sie können angeben, dass ein separates Protokoll für jeden Cache verwaltet werden soll oder dass alle Statistikdaten in das gleiche Protokoll geschrieben werden. In diesem Abschnitt werden die folgenden Themen zum Cache-Protokoll erläutert:

- „Konfigurieren des Protokolls"
- „Format des Cache-Protokolls"

Konfigurieren des Protokolls

Sie müssen die Konfigurationsdatei für den Cache-Manager konfigurieren, um Statistikdaten zu protokollieren.

Gehen Sie wie folgt vor, um das Protokoll zu konfigurieren:

Geben Sie die Schlüsselwörter `stat-files` und `stat-interval` in der Cache-Zeilengruppe der Konfigurationsdatei für den Cache-Manager an.

Sie können die Einstellungen für die Statistikdaten ändern, ohne den Cache-Manager stoppen, rekonfigurieren und erneut starten zu müssen.

Gehen Sie wie folgt vor, um die Einstellungen zum Sammeln von Statistikdaten zu ändern:

Geben Sie den Befehl `cacheadm statistics` an. Beachten Sie jedoch, dass über den Befehl `cacheadm statistics` vorgenommene Änderungen nicht gesichert werden, wenn der Cache-Manager erneut gestartet wird.

Format des Cache-Protokolls

Das Statistikdatenprotokoll ist eine unverschlüsselte ASCII-Datei, die von Tabellenkalkulations- und Datenbankprogrammen verarbeitet und importiert werden kann. Es werden drei Arten von Datensätzen geschrieben:

- Initialisierungsdatensätze dokumentieren den Beginn der Sammlung von Statistikdaten für einen bestimmten Cache. Diese Datensätze haben folgendes Format:

  mm/dd/yy hh:mm:ss id `Initialization: interval` *n* `seconds`

  Dabei gilt Folgendes:

mm/dd/yy	Monat, Tag und Jahr des Beginns der Sammlung von Statistikdaten
hh:mm:ss	Stunde, Minute und Sekunde des Beginns der Sammlung von Statistikdaten
id	Name des dem Datensatz zugeordneten Cache
n	Sammlungsintervall

- Beendigungsdatensätze dokumentieren die Beendigung der Sammlung von Statistikdaten für einen bestimmten Cache. Diese Datensätze haben folgendes Format:

  ```
  mm/dd/yy hh:mm:ss id Termination
  ```

  Dabei gilt Folgendes:

mm/dd/yy	Monat, Tag und Jahr des Endes der Sammlung von Statistikdaten
hh:mm:ss	Stunde, Minute und Sekunde des Endes der Sammlung von Statistikdaten
id	Name des dem Datensatz zugeordneten Cache

- Datensätze mit Statistikdaten sind eine durch Leerzeichen begrenzte Gruppe von Nummern, die die Aktivität im Cache anzeigen. Diese Datensätze haben folgendes Format:

  ```
  mm/dd/yy hh:mm:ss id statistics
  ```

  Dabei gilt Folgendes:

mm/dd/yy	Monat, Tag und Jahr der Erstellung der Sammlung von Statistikdaten
hh:mm:ss	Stunde, Minute und Sekunde der Erstellung der Sammlung von Statistikdaten
id	Name des dem Datensatz zugeordneten Cache
<*statistics*>	Liste der durch Leerzeichen begrenzten und für diesen Cache gesammelten Statistikdaten wie in Tabelle 15 dargestellt:

Tabelle 15. Liste der Statistikdaten

Feld-nummer	Inhalt	Beschreibung	Zähler auf Null zurückgesetzt
1	Lese-operationen	Anzahl der Leseoperationen für den Cache	Ja
2	Schreib-operationen	Anzahl der Schreiboperationen für den Cache	Ja
3	Schließ-operationen	Anzahl der Schließoperationen für Objekte im Cache	Ja
4	Öffnungs- und Lese-operationen	Anzahl der Öffnungs- und Lese-operationen für Objekte im Cache	Ja

Tabelle 15. Liste der Statistikdaten (Forts.)

Feld-nummer	Inhalt	Beschreibung	Zähler auf Null zurückgesetzt
5	Öffnungs- und Schreib-operationen	Anzahl der Öffnungs- und Schreib-operationen für Objekte im Cache	Ja
6	Öffnungs-, Schreib- und Abfrage-operationen	Anzahl der Öffnungs-, Schreib- und Abfrageoperationen für Objekte im Cache	Ja
7	Erfolgreiche Lese-operationen	Anzahl der erfolgreichen Leseoperationen für Objekte im Cache	Ja
8	Erfolgreiche Schreib-operationen	Anzahl der erfolgreichen Schreib-operationen für Objekte im Cache	Ja
9	Erfolgreiche Schreib- und Abfrage-operationen	Anzahl der erfolgreichen Schreib- und Abfrageoperationen für Objekte im Cache	Ja
10	Initialisie-rungen	Anzahl der mit dem Cache neu erstellten Sitzungen	Ja
11	Beendigungen	Anzahl der mit dem Cache beendeten Sitzungen	Ja
12	Lösch-operationen	Anzahl der aus diesem Cache gelöschten Objekte	Nein
13	Speicher-belegung	Von Objekten im Hauptspeicherbereich des Cache belegter Speicher	Nein
14	Platten-belegung	Von Objekten im Plattenbereich des Cache belegter Speicher	Nein
15	Verfügbarer Hauptspeicher	Für Objekte im Hauptspeicherbereich des Cache noch verfügbarer Speicherbereich	Nein
16	Verfügbarer Platten-speicherplatz	Für Objekte im Plattenbereich des Cache noch verfügbarer Plattenspeicherplatz	Nein
17	Anz. Haupt-speicherobj.	Anzahl der Objekte im Hauptspeicher-bereich des Cache	Nein
18	Anzahl Datei-objekte	Anzahl der Objekte im Plattenbereich des Cache	Nein
19	Anzahl Sitzungen	Anzahl der momentan für den Cache aktiven Sitzungen	Nein

Festlegen der Fehlerprotokollstufe

Net.Data stellt ein Fehlerprotokoll bereit, mit dem Sie Fehler bzw. Leistungsprobleme auf Ihrem Net.Data-System überwachen können.

Wenn Sie das Net.Data-Fehlerprotokoll verwenden und viele Nachrichten in die Fehlerprotokolle geschrieben werden, kann die Systemleistung möglicherweise beeinträchtigt werden. Zum Beispiel übergibt Net.Data jedesmal, wenn ein Benutzer auf ein Makro zugreift, das Net.Data nicht finden kann, eine Nachricht als Ausgabe an das Fehlerprotokoll.

Prüfen Sie die in einem Net.Data-Makro festgelegte Protokollstufe mit dem Schlüsselwort DTW_LOG_LEVEL, um die Leistungseinbuße zu verringern. Wenn die Protokollstufe auf WARNING gesetzt ist, erwägen Sie, ob Sie sie auf ERROR (geringe Leistungssteigerung) bzw. auf OFF (hohe Leistungssteigerung) herabsetzen.

Optimieren der Sprachumgebungen

In den folgenden Abschnitten werden Methoden beschrieben, mit denen Sie die Leistung optimieren können, wenn Sie die von Net.Data bereitgestellten Sprachumgebungen verwenden.

- „REXX-Sprachumgebung"
- „SQL-Sprachumgebung" auf Seite 234
- „SYSTEM- und Perl-Sprachumgebungen" auf Seite 235

REXX-Sprachumgebung

Beachten Sie die folgenden Hinweise, um die Leistung Ihrer Net.Data-Anwendung zu optimieren:

- Kombinieren Sie Ihre REXX-Programme, wo möglich. Die Verwendung einer geringeren Anzahl von größeren Programmen bietet eine bessere Leistung als die Verwendung einer größeren Anzahl von kleineren Programmen, weil der REXX-Interpreter bei jedem Aufruf einer REXX-Sprachumgebungsfunktion im Makro initialisiert wird.
- Speichern Sie das REXX-Programm in einer externen Datei, statt es inline in das Net.Data-Makro einzufügen.
- Verweisen Sie bei externen REXX-Programmen in der Befehlszeile in der %EXEC-Anweisung auf die globalen Variablen.
- Übergeben Sie Eingabeparameter direkt an ein REXX-Programm, indem Sie globale Net.Data-Variablen definieren und auf die Variablen verweisen. Rufen Sie bei internen REXX-Programmen die globalen Variablen direkt in Ihrer REXX-Quelle auf.
- Erwägen Sie die Verwendung von MACRO_FUNCTION-Blöcken anstelle von REXX-Programmen, um den Systemaufwand für das Starten des REXX-Interpreters zu vermeiden.

SQL-Sprachumgebung

In diesem Abschnitt werden Leistungstechniken für die Datenbank und die SQL-Sprachumgebung beschrieben. Besuchen Sie folgende Website, um mehr über DB2-Leistungsüberlegungen zu erfahren:
http://review.ibm.com/software/data/db2/performance

Datenbankverfahren

Die folgende Zusammenfassung nennt einige der einfachsten Datenbankverfahren, mit denen der Zugriff auf die Datenbank verbessert werden kann:

- Aktivieren Sie die Datenbank. Durch Absetzen des Befehls `db2 activate database` *`datenbankname`* werden Verbindungen zur Datenbank wesentlich schneller hergestellt. Weitere Informationen zum DB2-Befehl zum Aktivieren von Datenbanken finden Sie im Handbuch *DB2 Systemverwaltung*.
- Vermeiden Sie numerische Umwandlungen. Wenn ein Spaltenwert und ein Literalwert verglichen werden, versuchen Sie, die gleichen Datentypen und Attribute anzugeben. DB2 verwendet keinen Index für die benannte Spalte, wenn der Literalwert eine höhere Genauigkeit hat als die Spalte. Wenn die beiden zu vergleichenden Elemente unterschiedliche Datentypen aufweisen, muss DB2 einen der beiden Werte umwandeln, was zu Ungenauigkeiten führen kann (aufgrund der beschränkten Genauigkeit der Maschine).

  EDUCLVL ist beispielsweise ein ganzzahliges Halbwort (SMALLINT). Geben Sie Folgendes an:

  ```
  ... WHERE EDUCLVL < 11 AND EDUCLVL >= 2
  ```

  Und nicht:

  ```
  ... WHERE EDUCLVL < 1.1E1 AND EDUCLVL > 1.3
  ```

- Vermeiden Sie das Auffüllen von Zeichenfolgen. Versuchen Sie, die gleiche Datenlänge zu verwenden, wenn Sie einen Zeichenfolgespaltenwert mit fester Länge mit einem Literalwert vergleichen. DB2 verwendet keinen Index, wenn der Literalwert länger ist als die Spaltenlänge.

  EMPNO beispielsweise ist CHAR(6) und DEPTNO CHAR(3). Geben Sie Folgendes an:

  ```
  ... WHERE EMPNO > '000300' AND  DEPTNO < 'E20'
  ```

  Und nicht:

  ```
  ... WHERE EMPNO > '000300 ' AND DEPTNO < 'E20 '
  ```

- Vermeiden Sie die Verwendung von LIKE-Mustern, die mit % oder _ anfangen. Das Prozentzeichen (%) und das Unterstreichungszeichen (_) geben bei Verwendung im Muster eines LIKE-Vergleichselements eine Zeichenfolge an, die ähnlich ist wie der Spaltenwert von Zeilen, die Sie auswählen möchten. Bei Verwendung zur Angabe von Zeichen in der Mitte oder am Ende einer Zeichenfolge können LIKE-Muster Indizes nutzen. Beispiel:

  ```
  ... WHERE LASTNAME LIKE 'J%SON%'
  ```

Bei Verwendung am Anfang einer Zeichenfolge können LIKE-Muster DB2 jedoch daran hindern, Indizes zu verwenden, die in der Spalte LASTNAME definiert sein könnten, um die Anzahl der gelesenen Zeilen zu beschränken. Beispiel:

```
... WHERE LASTNAME LIKE '%SON'
```

Vermeiden Sie die Verwendung dieser Symbole am Anfang von Zeichenfolgen, vor allem, wenn Sie auf eine besonders große Tabelle zugreifen.

Verfahren für die SQL-Sprachumgebung

- Wenn eine Ergebnismenge eine große Anzahl Zeilen enthält, können Sie eine Untergruppe der Ergebnismenge angeben, die mit START_ROW_NUM und RPT_MAX_ROWS an den Browser zurückgegeben wird. START_ROW-_NUM gibt an, bei welcher Zeile die zurückgegebene Untergruppe beginnen soll, und RPT_MAX_ROWS gibt die Anzahl der Zeilen an, die an die Seite zurückgegeben werden sollen. START_ROW_NUM kann anschließend in einer Verbindung (Link) zum Anzeigen der nächsten Ergebnisseite verwendet werden.

  Beachten Sie, dass Net.Data die Abfrage für jede Seite erneut absetzt, da die Cursorposition im Verlauf der Anforderungen nicht beibehalten wird.
- Erwägen Sie die Verwendung einer gespeicherten Prozedur, um komplexe Datenbank-Tasks auszuführen. Die Verwendung von eingebettetem SQL und das Verständnis der Struktur von Ergebnismengen verringern den Systemaufwand, den Net.Data erzeugt, um Ergebnisse dynamisch zu beschreiben. Weitere Informationen zu den Nachteilen der Leistung bei der Verwendung von gespeicherten Prozeduren finden Sie im Handbuch *DB2 Systemverwaltung*.
- Wenn Sie SQL-Anweisungen haben, bei denen die einzigen veränderlichen Informationen die Eingabewerte in der Klausel WHERE sind, denken Sie daran, die Vorteile der Funktion DTW_USE_DB2_PREPARE_CACHE von Net.Data auszunutzen. Setzen Sie diesen Wert in der Initialisierungsdatei auf "YES" oder in einzelnen Makros, wenn Sie ihn nicht global anwenden wollen. Diese Einstellung lässt Net.Data Host-Variablen für die Eingabewerte verwenden, um DB2 zu helfen, Anweisungen schneller vorzubereiten.

SYSTEM- und Perl-Sprachumgebungen

Übergeben Sie Eingabeparameter direkt an das Programm, das die SYSTEM- oder Perl-Sprachumgebung aufruft. Definieren Sie dazu globale Net.Data-Variablen, und verweisen Sie darauf. Verweisen Sie bei externen Programmen und Perl-Scripts in der Befehlszeile in der %EXEC-Anweisung auf die Variablen. Verweisen Sie bei internen Perl-Scripts direkt in der Perl-Quelle auf die Variablen. Erwägen Sie auch die Verwendung von MACRO_FUNCTION-Blöcken anstelle von Perl-Scripts, um den Systemaufwand für das Starten des Perl-Interpreters zu vermeiden.

Kapitel 8. Net.Data-Protokollierung

Net.Data stellt mehrere Protokolle für die Überwachung der Net.Data-Leistung und Fehlerbehebung bereit. Es gibt unter anderem folgende Net.Data-Protokolle:

Net.Data-Fehlernachrichtenprotokoll
Protokoll mit allen Net.Data-Fehlernachrichten

Direktverbindungsprotokoll
Protokoll mit Direktverbindungsfehlernachrichten und der Kommunikation zwischen Net.Data, der Direktverbindung und der DB2-Datenbank. Protokollierung ist für die Sprachumgebungen DTW_SQL und DTW_ODBC bei DB2-Datenbanken verfügbar.

Net.Data-Trace-Protokoll
Protokoll mit allen Net.Data-Trace-Nachrichten

In den folgenden Abschnitten wird Net.Data-Protokollierung beschrieben:

- „Protokollieren von Net.Data-Fehlernachrichten"
- „Protokollieren von Cliette-Nachrichten und Fehlernachrichten bei Direktverbindung" auf Seite 240
- „Net.Data-Trace-Protokoll" auf Seite 245

Protokollieren von Net.Data-Fehlernachrichten

Net.Data schreibt Fehlernachrichten in die Net.Data-Fehlerprotokolldatei namens `netdata.log`. Die maximale Größe des Fehlerprotokolls wird von Net-.Data auf 500 KB begrenzt. Dies entspricht ungefähr 3000 Protokolleinträgen.

Sie können die Fehlerprotokolldatei bzw. archivierte Kopien in bestimmten Abständen durchsuchen, um zu ermitteln, ob im Net.Data-System Probleme aufgetreten sind.

Gehen Sie wie folgt vor, um das Net.Data-Fehlerprotokoll zu aktivieren:

- Legen Sie die Net.Data-Konfigurationsvariable zur Protokollierung DTW_LOG_DIR wie folgt fest:

  ```
  DTW_LOG_DIR path
  ```

  Dabei ist *path* das Verzeichnis, in dem die Fehlerprotokolldatei gespeichert werden soll.

- Legen Sie die Net.Data-Protokollierungsvariable DTW_LOG_LEVEL im Makro wie folgt fest:

  ```
  @DTW_ASSIGN(DTW_LOG_LEVEL, "level")
  ```

  Dabei ist *level* die Protokollstufe. Folgende Werte sind gültig:

  off Net.Data protokolliert Fehler nicht. Dies ist der Standardwert.

  error Net.Data protokolliert Fehlernachrichten.

In diesem Abschnitt werden die folgenden, die Protokollierung betreffenden Themen erörtert:

- „Planen für das Net.Data-Fehlerprotokoll"
- „Steuern der Net.Data-Protokollstufe" auf Seite 239
- „Arten nicht protokollierter Net.Data-Fehlernachrichten" auf Seite 239
- „Größe und Archivierung der Net.Data-Fehlerprotokolldatei" auf Seite 239
- „Net.Data-Fehlerprotokollformat" auf Seite 240

Planen für das Net.Data-Fehlerprotokoll

Zur Protokollierung von Fehlern müssen Sie die folgenden Punkte einplanen:

- Bestimmen des Plattenspeicherplatzes:

  Wenn Sie die Verwendung von Fehlerprotokollierung wünschen, müssen Sie zusätzlichen Plattenspeicherplatz für die Fehlerprotokolle freigeben.

- Konfigurieren von Net.Data:

  Wenn Fehler für das gesamte Net.Data-System protokolliert werden sollen, legen Sie die Konfigurationsvariable DTW_LOG_DIR in Ihrer Net.Data-Initialisierungsdatei fest.

  Diese Variable ist für die Fehlerprotokollierung erforderlich, selbst wenn Sie die Variable DTW_LOG_LEVEL in Ihrem Makro auf ERROR (Fehler) oder WARNING (Warnung) gesetzt haben. Informationen zum Aktualisieren der Initialisierungsdatei finden Sie in „DTW_LOG_DIR und DTW_LOG_LEVEL: Fehlerprotokollvariablen" auf Seite 19.

- Schreiben von Net.Data-Makros:

  Legen Sie die Protokollstufe mit dem Schlüsselwort DTW_LOG_LEVEL in Ihrem Makro fest.

- Ausführen von Net.Data:

  Wenn Sie die Fehlerprotokollierung verwenden, können Sie die Fehlerprotokolle und Archivdateien auf Fehler in Ihrem Net.Data-System überprüfen.

- Optimierung:

  Berücksichtigen Sie, dass sich die Protokollierung nachteilig auf die Leistung auswirken kann. Informationen zur Leistung finden Sie in „Festlegen der Fehlerprotokollstufe" auf Seite 233.

Steuern der Net.Data-Protokollstufe

Sie können die Protokollstufe mit der Variablen DTW_LOG_LEVEL angeben. Definieren Sie dieses Schlüsselwort im Net.Data-Makro. Für diese Variable gibt es drei Einstellungen:

off Net.Data protokolliert Fehler nicht. Dies ist der Standardwert.

error Net.Data protokolliert Fehlernachrichten.

warning
Net.Data protokolliert sowohl Warnungen als auch Fehlernachrichten.

Arten nicht protokollierter Net.Data-Fehlernachrichten

Net.Data protokolliert die Fehler, die explizit von einem MESSAGE-Abschnitt im Makro gehandhabt werden, nicht.

Größe und Archivierung der Net.Data-Fehlerprotokolldatei

Die Protokolldatei kann maximal 500 KB groß sein. Bei dieser Größe enthält sie ungefähr 3000 Protokolleinträge.

Wenn die Protokolldatei die maximale Größe erreicht, wird sie unter dem Namen netdata.log*MMMDDYYYY_nn* archiviert.

Dabei gilt Folgendes:

MMM Der Monat (Jan-Dec)

DD Das Datum

YYYY Das Jahr

nn Eine Zahl zwischen 01 und 99, die jede Archivdatei eindeutig für einen bestimmten Tag kennzeichnet.

Die Protokollierung wird in der Originaldatei fortgesetzt.

Net.Data-Fehlerprotokollformat

Protokolldateieinträge haben folgendes Format:
`[DD/MMM/YYYY:HH:MM:SS][MACRO][BLOCK][PID#][TID#]error_message`

Parameter:

DD Das Datum

MMM Der Monat (Jan-Dec)

YYYY Das Jahr

HH Die Stunde (00-23)

MM Die Minute (00-59)

SS Die Anzahl Sekunden (00-59)

MACRO
Das Makro, das die Fehlernachricht generierte

BLOCK
Der Name des HTML-Blocks, der die Fehlernachricht generierte

PID# Die Nummer der Prozess-ID des Prozesses, der die Fehlernachricht generierte. Diese ID ist notwendig, weil mehrere Net.Data-Prozesse in die Protokolldatei schreiben können.

TID# Die Nummer der Thread-ID des Threads, der die Fehlernachricht generierte. Diese ID ist notwendig, weil mehrere Threads vom gleichen Net.Data-Prozess in die Protokolldatei schreiben können.

error_message
Der Text der Fehlernachricht

Protokollieren von Cliette-Nachrichten und Fehlernachrichten bei Direktverbindung

Eine Direktverbindung zeichnet Nachrichten und die Kommunikation zwischen der Direktverbindung, Net.Data und der DB2-Datenbank in der Direktverbindungsprotokolldatei auf. Die maximale Größe des Protokolls wird von Net.Data auf 1 MB begrenzt. Dies entspricht ungefähr 1200 Protokolleinträgen.

Sie können die Protokolldatei bzw. archivierte Kopien in bestimmten Abständen durchsuchen, um zu ermitteln, ob bei Ihren Cliettes oder der DB2-Datenbank Probleme aufgetreten sind.

Gehen Sie wie folgt vor, um das Direktverbindungsprotokoll zu aktivieren:

Starten Sie Connection Manager wie folgt mit dem Attribut `-l`:

```
dtwcm -l [level]
```

Dabei ist *level* die Protokollstufe. Folgende Werte sind gültig:

normal
: Die Direktverbindung protokolliert alle Cliette-Aktivitäten, zugehörige DB2-SQL-Anweisungen und DB2-Statusnachrichten sowie Direktverbindungsfehlernachrichten.

minimal
: Die Direktverbindung protokolliert nur wesentliche Informationen wie Datenbankabfragen und die Anzahl der Zeilen in der Ergebnismenge.

In diesem Abschnitt werden die folgenden, die Protokollierung betreffenden Themen erörtert:

Planen für das Direktverbindungsprotokoll

Zur Protokollierung von Nachrichten müssen Sie die folgenden Punkte einplanen:

- Bestimmen des Plattenspeicherplatzes:

  Wenn Sie die Verwendung von Fehlerprotokollierung wünschen, müssen Sie zusätzlichen Plattenspeicherplatz für die Protokolldateien freigeben.

- Ausführung von Connection Manager:

  Sie aktivieren die Protokollierung durch Eingabe eines Attributs im Befehl `dtwcm`. Informationen zur Syntax finden Sie in „Protokollieren von Cliette-Nachrichten und Fehlernachrichten bei Direktverbindung" auf Seite 240.

  Wenn Sie Protokollierung verwenden, können Sie die Fehlerprotokolle und Archivdateien in Ihren Cliettes auf Fehler überprüfen.

- Optimierung:

  Berücksichtigen Sie, dass sich die Protokollierung nachteilig auf die Leistung auswirken kann. Informationen zu die Leistung betreffenden Punkten finden Sie in „Festlegen der Fehlerprotokollstufe" auf Seite 233 und „Steuern der Direktverbindungsprotokollstufe" auf Seite 242.

Steuern der Direktverbindungsprotokollstufe

Sie können die Protokollstufe im Befehl `dtwcm` beim Aufrufen von Connection Manager angeben. Für das Attribut `-l` des Befehls `dtwcm` gibt es zwei Einstellungen:

normal
Die Direktverbindung protokolliert alle Cliette-Aktivitäten, zugehörige DB2-SQL-Anweisungen und DB2-Statusnachrichten sowie Direktverbindungsfehlernachrichten.

minimal
Die Direktverbindung protokolliert nur wesentliche Nachrichten. Diese Option generiert weniger Nachrichten im Protokoll.

Arten nicht protokollierter Direktverbindungsnachrichten

Die Direktverbindung protokolliert Net.Data-Fehler oder Fehler, die explizit von einem MESSAGE-Abschnitt im Makro gehandhabt werden, nicht.

Namen der Direktverbindungsprotokolldateien

Die Direktverbindung erstellt eine Protokolldatei für Connection Manager und für jede Cliette. In der folgenden Liste werden die Namensformate beschrieben:

Connection Manager-Datei

Format:

`conman-`*`process_id-DDMMMYYYYHHMMSS`*`.log`

Parameter:

process_id
Die Kennung des Connection Manager-Prozesses

DD
Das Datum

MMM
Der Monat (Jan-Dec)

YYYY
Das Jahr

HH
Die Stunde (24-Stundenformat)

MM
Die Minuten

SS Die Sekunden

Beispiel:

```
conman-513-01Feb1999095639.log
```

Cliette-Datei

Format:

`cliett-`*`process_id-DDMMMYYYYHHMMSS`*`.log`

Parameter:

process_id
Die Kennung des Cliette-Threads

DD
Das Datum

MMM
Der Monat (Jan-Dec)

YYYY
Das Jahr

HH
Die Stunde (24-Stundenformat)

MM
Die Minuten

SS Die Sekunden

Beispiel:

```
cliett-592-01Feb1999095647.log
```

Größe und Archivierung der Direktverbindungsprotokolldatei

Die Protokolldatei kann maximal 1 MB groß sein. Bei dieser Größe enthält sie ungefähr 6000 Protokolleinträge. Wenn die Protokolldatei die maximale Größe erreicht, schließt der Prozess die Originalprotokolldatei, erstellt eine neue Protokolldatei und setzt die Protokollierung in der neuen Datei fort.

Protokolldateien werden im gleichen Verzeichnis gespeichert wie `dtwcm` und `dtwcdb2`.

Direktverbindungsprotokollformat

Protokolldateieinträge haben folgendes Format:

```
--process_type-DD/MMM/YYYY:HH:MM:SS-PID#--
message_text
```

Parameter:

process_type
Entweder `dtwcm` oder `cliet` in Abhängigkeit davon, ob Connection Manager oder eine Cliette die Nachricht protokolliert hat

DD Das Datum

MMM Der Monat (Jan-Dec)

YYYY Das Jahr

HH Die Stunde (00-23)

MM Die Minute (00-59)

SS Die Anzahl Sekunden (00-59)

PID# Die Nummer der Prozess-ID des Prozesses, der die Nachricht generierte

message_text
Der Text der Nachricht

Beispiel 1: Ein Connection Manager-Protokolleintrag

```
--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.
...
```

Beispiel 2: Ein Cliette-Protokolleintrag

```
--cliet-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 ...
```

Net.Data-Trace-Protokoll

Net.Data stellt Trace-Daten über die Ausführung Ihres Makros bereit, die im Trace-Protokoll aufgezeichnet werden. Sie können angeben, wo das Trace-Protokoll gespeichert und welche Trace-Stufe aufgezeichnet wird. Mit Hilfe der IBM Trace-Informationen verfügen Sie über Informationen, wenn Sie mit Ihrem IBM Ansprechpartner arbeiten. Eine Liste mit Net.Data-Trace-Nachrichten finden Sie in *Net.Data Nachrichten und Codes*.

Konfigurieren von Net.Data für die Trace-Funktion

Zum Konfigurieren von Net.Data für die Trace-Funktion müssen Sie Konfigurationsvariablen festlegen, mit denen Sie angeben, wo das Trace-Protokoll gespeichert wird und welche Stufe der Trace-Daten Net.Data erfassen soll.

- „Festlegen des Trace-Protokoll-Verzeichnisses"
- „Festlegen der Trace-Protokollstufe"

Festlegen des Trace-Protokoll-Verzeichnisses

Der Name des Trace-Protokolls ist `netdata.trace.pid`. Dabei ist `pid` die ID des Prozesses, der die Anforderung bearbeitet. Mit der Konfigurationsvariablen DTW_TRACE_LOG_DIR können Sie das Verzeichnis angeben, in dem die Trace-Datei gespeichert wird.

Anmerkung: Net.Data beschränkt die Protokolldateigröße auf 50 MB. Wenn die Protokolldateigröße 50 MB enthält, wird die Datei unter dem Namen `net.data.trace.pid.0` archiviert.

Syntax:

```
DTW_TRACE_LOG_DIR [=] full_directory_path
```

Beispiel:

```
DTW_TRACE_LOG_DIR /usr/lpp/internet/server_root/logs
```

Festlegen der Trace-Protokollstufe

Durch Setzen des Werts für die Konfigurationsvariable DTW_TRACE_LOG_LEVEL bestimmen Sie die Trace-Stufe, die Net.Data protokolliert.

Syntax:

```
DTW_TRACE_LOG_LEVEL [=] OFF|APPLICATION|SERVICE
```

Dabei gilt Folgendes:

OFF Gibt an, dass keine Trace-Daten im Trace-Protokoll erfasst werden. Dies ist der Standardwert.

APPLICATION
Net.Data schreibt Trace-Nachrichten der Anwendungsstufe in das Trace-Protokoll.

SERVICE
Net.Data schreibt *alle* Trace-Nachrichten in das Trace-Protokoll.

Beispiel:

```
DTW_TRACE_LOG_LEVEL SERVICE
```

Format des Trace-Protokolls

Das Format eines Trace-Protokolleintrags sieht wie folgt aus:

```
[DD/MMM/YYYY:HH:MM:SS][macro][PID#][TID#][UID] trace_message
```

Dabei gilt Folgendes:

DD/MMM/YYYY:HH:MM:SS
Eine Zeitmarke, die angibt, wann der Trace-Eintrag erstellt wurde

macro Der Name des Makros, das die Trace-Nachricht generiert hat

PID# Die Prozess-ID des Prozesses, der die Trace-Nachricht generiert hat

TID# Die ID des Thread, der die Trace-Nachricht generiert hat

UID Die ID des Benutzers, der die Trace-Nachricht generiert hat

trace_message
Der Text der Trace-Nachricht

Zugriffsrechte

Die Benutzer-ID, unter der Net.Data ausgeführt wird, müssen über die folgenden Berechtigungen verfügen, um Trace-Nachrichten in die Trace-Protokolldatei schreiben zu können:

- Schreibberechtigung für das in der Konfigurationsvariablen DTW_TRACE_LOG_DIR angegebene Protokollverzeichnis
- Ausführungsberechtigung für alle Verzeichnisse im Pfad, einschließlich des Protokollverzeichnisses

Anhang A. Literaturübersicht

Net.Data Technical Library

Die Net.Data Technical Library auf der Net.Data-Website ist unter folgender Adresse verfügbar:

`http://www.ibm.com/software/data/net.data/library.html`

Dokument	Beschreibung
• *Net.Data Administration and Programming Guide for OS/390* • *Net.Data Verwaltung und Programmierung für OS/2, Windows NT und UNIX* • *Net.Data Verwaltung und Programmierung für OS/400*	Enthält Informationen zu Konzepten und Tasks für das Installieren, Konfigurieren und Aufrufen von Net.Data. Außerdem wird das Schreiben von Net.Data-Makros, die Verwendung von Net.Data-Leistungsverfahren und Net.Data-Sprachumgebungen, die Verwaltung von Verbindungen und die Verwendung von Net.Data-Protokoll- und Trace-Funktionen für die Fehlerbehebung und Leistungsverbesserung beschrieben.
Net.Data Reference	Beschreibt die Net.Data-Makrosprache, Variablen und integrierte Funktionen.
Net.Data Language Environment Interface Reference	Beschreibt die Net.Data-Sprachumgebungsschnittstelle.
Net.Data Nachrichten und Codes	Listet die Net.Data-Fehlernachrichten und -Rückkehrcodes auf.

Anhang B. Net.Data für AIX

Ausführliche Informationen zu AIX finden Sie in der Informationsdatei (README), die zusammen mit Net.Data geliefert wird. Diese Datei enthält die folgenden Informationen:

- Anforderungen
- Installation
- Konfiguration
- Entfernen der Installation

Laden gemeinsam benutzter Bibliotheken für Sprachumgebungen

Wenn Sie auf einer AIX-Plattform eine Sprachumgebung erstellen, müssen Sie gemeinsam benutzte Bibliotheken laden. Unter AIX ist die Sprachumgebung erforderlich, damit eine von Net.Data aufgerufene Routine bereitgestellt wird, welche die Adressen der Schnittstellenroutinen der Sprachumgebung, zum Beispiel dtw_initialize() und dtw_execute(), zurückgibt.

Net.Data verwendet die Struktur dtw_fp, um Zeiger auf die Schnittstellenroutinen der Sprachumgebung aus einer Sprachumgebung in AIX abzurufen. Diese Struktur hat folgendes Format:

```
typedef struct dtw_fp {
    int (* dtw_initialize_fp)();   /* dtw_initialize function pointer */
    int (* dtw_execute_fp)();      /* dtw_execute function pointer    */
    int (* dtw_cleanup_fp)();      /* dtw_cleanup function pointer    */
} dtw_fp_t;
```

Diese Struktur wird von Net.Data als Parameter in der Routine dtw_getFp() an die Sprachumgebung übermittelt, wenn die gemeinsam benutzte Bibliothek geladen wird.

Die Struktur dtw_fp ist der einzige übermittelte Parameter. Diese Struktur enthält für jede unterstützte Schnittstelle ein Feld, das von der Sprachumgebung eingestellt wird. Wenn die Sprachumgebung die angegebene Schnittstelle bereitstellt, stellt sie dieses Feld auf den Funktionszeiger der betreffenden Schnittstelle ein. Wird die angegebene Schnittstelle nicht bereitgestellt, wird das Feld auf NULL gesetzt. Die Routine dtw_getFp() in der Programmschablone zeigt eine korrekte Implementierung dieser Routine an.

Damit Net.Data den Zeiger auf diese Routine findet, wenn die gemeinsam benutzte Bibliothek geladen wird, muss die Routine dtw_getFp in der Exportdatei der gemeinsam benutzten Bibliothek als erster Eintrag angegeben sein.

Nachfolgend finden Sie ein Beispiel für eine Exportdatei einer Bibliothek namens dtwsampshr.o, die alle verfügbaren Schnittstellenroutinen für Sprachumgebungen unterstützt:

```
#!dtwsampshr.o
dtw_getFp
dtw_initialize
dtw_execute
dtw_cleanup
```

Leistungsverbesserung in der REXX-Umgebung

Wenn Sie die REXX-Sprachumgebung auf Ihrem AIX-System oft aufrufen, sollten Sie die Umgebungsvariable RXQUEUE_OWNER_PID auf 0 stellen. Makros, die viele Aufrufe an die REXX-Sprachumgebung ausführen, können viele Prozesse erstellen, die die Systemressourcen aufbrauchen.

Sie haben drei Möglichkeiten, die Umgebungsvariable einzustellen:

- Im Makro mit Hilfe der integrierten Funktion DTW_SETENV:

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

- In der AIX-Systemumgebungsdatei:

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

  Diese Methode beeinflusst das Verhalten von REXX auf der gesamten Maschine.

- In der Umgebungsdatei des HTTP-Webservers. Das nachfolgende Beispiel gilt für Domino Go Webserver. Für dieses Produkt müssen Sie die folgende Anweisung einfügen:

  ```
  InheritEnv RXQUEUE_OWNER_PID = 0
  ```

  Diese Methode beeinflusst das Verhalten von REXX auf dem Webserver.

Überlegungen zu Landessprachen

Net.Data wird mit der gleichen Codepage ausgeführt wie der Webserver. Damit Net.Data die korrekte Codepage für Ihre länderspezifischen Angaben verwendet, muss der Webserver die korrekte Codepage verwenden. Wenn Sie z. B. mit einem IBM Internet Connection Server arbeiten und eine koreanische Codepage verwenden wollen, stoppen Sie den Server, und starten Sie ihn mit den koreanischen länderspezifischen Angaben erneut:

```
stopsrc httpd
startsrc -s httpd -e "LC_ALL=ko_KR"
```

Wenn Ihr Makro UTF-8-Zeichen enthält oder wenn Sie die Verbindung zu einer DB2-Datenbank mit Unicode-Daten herstellen, setzen Sie die Konfigurationsvariable DTW_UNICODE in der Initialisierungsdatei auf UTF8. Net.Data unterstützt derzeit UTF-8-Zeichen im Makro, allerdings nicht UTF-16. Net-.Data kann jedoch DB2-Datenbankdaten verarbeiten, die in UTF-8 oder UCS-2 codiert sind. Die Net.Data-Ausgabe liegt immer in UTF-8 vor.

Tipp zur Leistungsoptimierung: Wenn Ihre Umgebung auf länderspezifische Angaben mit Doppelbytezeichen ausgelegt ist, die integrierten Zeichenfolge- oder Arbeitsfunktionen jedoch immer Einzelbytezeichenfolgen verarbeiten, setzen Sie DTW_MBMODE auf NO, um sich unnötige Umwandlungen zu ersparen.

Anhang C. Net.Data-Assistenten

Die Net.Data-Assistenten bieten Ihnen eine schnelle und einfache Methode, benutzerdefinierte Net.Data-Anwendungen zu erstellen. Wählen Sie einfach einen Assistent aus, beantworten Sie einige Fragen, und schon erstellt Net.Data Ihre benutzerdefinierte Anwendung.

Net.Data enthält die folgenden SmartGuides, die Sie verwenden und dabei das Erstellen von Makros und die Anwendung der Net.Data-Funktionen erlernen können:

Drilldown
Dieser SmartGuide verwendet Ihre vorhandenen Datenbanktabellen und erstellt eine webfähige Drilldown-Anwendung, die es Ihnen ermöglicht, in unterschiedlichen Detaillierungsgraden auf Ihre Daten zuzugreifen. Optional kann der SmartGuide **Drilldown** eine Verbindung zu Ihrer Datenbank herstellen und Ihre Datenbankinformationen erfassen. Sie können bis zu fünf Webberichte nach Ihren Wünschen definieren. Das generierte Makro verwendet in Ihrer Datenbank gespeicherte Primär- und Fremdschlüsselinformationen, um Ihre Webberichte automatisch zu verbinden.

Gespeicherte Prozedur
Dieser SmartGuide stellt eine Verbindung zu Ihrer Datenbank her und ruft eine Liste aller gespeicherten Prozeduren ab, die in Ihrer Datenbank registriert sind. Wählen Sie eine gespeicherte Prozedur aus, und der SmartGuide generiert ein Net.Data-Makro, mit dem Sie Ihre gespeicherte Prozedur aufrufen können. Nun können Sie dieses generierte Makro ändern oder es in Ihre vorhandenen Net.Data-Anwendungen integrieren.

Kontakte
Dieser SmartGuide generiert ein webfähiges Adressbuch, in dem Sie Namen, Adressen, Rufnummern und andere wichtige Informationen speichern können. Zu diesem Adressbuch gehört eine Suchfunktion, mit der Sie schnell auf Ihre Kontakte zugreifen können. Die generierte Kontaktanwendung kann sowohl einen Kurz- als auch einen Gesamtbericht enthalten. Außerdem können Sie einen benutzerdefinierten Bericht hinzufügen.

Auf der Net.Data-Website unter `http://www.ibm.com/software/data/net.data` finden Sie die neueste Version des Net.Data-SmartGuide-Pakets.

In diesem Anhang werden die folgenden Themen behandelt:
- „Vorbereitung"
- „Ausführen der Assistenten"

Vorbereitung

Für die Ausführung der SmartGuides und das Generieren der Net.Data-Makros muss die folgende Software installiert sein:
- Java Development Kit (JDK) oder Java Runtime Environment (JRE) 1.1.x
- Net.Data Version 2 oder höher
- IBM Universal Database (UDB) 5.0 oder höher
- REXX (für den SmartGuide **Drilldown** erforderlich)

Ausführen der Assistenten

Die Net.Data-Assistenten werden von der Befehlszeile aus gestartet. Sie befinden sich in der Datei `NetDataSmartGuides.jar`.

Gehen Sie wie folgt vor, um einen Assistent mit Java Development Kit (JDK) zu starten:

1. Fügen Sie die folgende Zeile Ihrer Umgebungsvariablen CLASSPATH hinzu.

   [Path]`NetDataSmartGuides.jar`

   Dabei steht *[Path]* für den optionalen Pfad zur Datei `NetDataSmartGuides.jar`.
2. Wenn Sie die Funktion zum Verbinden der Datenbank der Assistenten **Drilldown** und **Gespeicherte Prozedur** verwenden wollen, müssen Sie der Umgebungsvariablen CLASSPATH den UDB-JDBC-Treiber hinzufügen.

   Für die Betriebssysteme Windows NT und OS/2 muss der Umgebungsvariablen CLASSPATH die folgende Zeile hinzugefügt werden:

   [Path]`NetDataSmartGuides.jar;`*[UDBInstallationPath]*`\java\db2java.zip`

   Für UNIX-Betriebssysteme muss der Umgebungsvariablen CLASSPATH die folgende Zeile hinzugefügt werden:

   [Path]`NetDataSmartGuides.jar:`*[UDBInstallationPath]*`\java\db2java.zip`

   Dabei steht *[Path]* für den optionalen Pfad zur Datei NetDataSmartGuides.jar und *[UDBInstallationPath]* für den Pfad zu Ihrer UDB-Installation, zum Beispiel C:\SQLLIB.
3. Starten Sie den Assistenten.
   - Geben Sie für UNIX-Betriebssysteme den folgenden Befehl ein:

     ```
     java TaskGuide LaunchPad.class
     ```

- Führen Sie für die Betriebssysteme Windows NT und OS/2 die folgende Datei aus:

  ```
  SmartGuides.cmd
  ```

Nun wird eine Klickstartleiste geöffnet, in der die verfügbaren Assistenten, wie in Abb. 29 dargestellt, aufgelistet werden.

Abbildung 29. Die Assistentenklickstartleiste

4. Klicken Sie den Namen des gewünschten Assistenten an.

Gehen Sie wie folgt vor, um einen Assistenten mit Java Runtime Environment (JRE) zu starten:

1. Wählen Sie unter Windows NT **Start->Programme->Net.Data->SmartGuides** aus, wodurch die Stapeldatei SMARTGUIDES.BAT ausgeführt wird. Geben Sie auf anderen Betriebssystemen den folgenden Befehl ein, um die Assistenten auszuführen:

   ```
   jre -cp [Path]NetDataSmartGuides.jar TaskGuide LaunchPad.class
   ```

   Dabei steht *[Path]* für den optionalen Pfad zur Datei `NetDataSmartGuides.jar`.

   Nun wird eine Klickstartleiste geöffnet, in der die verfügbaren Assistenten, wie in Abb. 29 dargestellt, aufgelistet werden.

2. Wenn Sie die Funktion zum Verbinden der Datenbank der Assistenten **Drilldown** und **Gespeicherte Prozedur** verwenden wollen, müssen Sie den folgendem Befehl eingeben:

   Für die Betriebssysteme Windows NT und OS/2:

   ```
   jre -cp [Path]NetDataSmartGuides.jar;[UDBInstallationPath]
     \java\db2java.zip TaskGuide LaunchPad.class
   ```

   Für UNIX-Betriebssysteme:

   ```
   jre -cp [Path]NetDataSmartGuides.jar:[UDBInstallationPfad]
     \java\db2java.zip TaskGuide LaunchPad.class
   ```

   Dabei steht *[Path]* für den optionalen Pfad zur Datei NetDataSmartGuides-.jar und *[UDBInstallationPath]* für den Pfad zu Ihrer UDB-Installation, zum Beispiel C:\SQLLIB.
3. Klicken Sie den Namen des gewünschten Assistenten an.

Anhang D. Erstellen von SQL-Anweisungen mit Net.Data SQL Assist

Net.Data SQL Assist ist ein Java-gestütztes Erstellungsprogramm für SQL-Anweisungen, das eine benutzerfreundliche grafische Benutzerschnittstelle bereitstellt, die Sie durch die einzelnen Schritte beim Erstellen von SQL-Anweisungen führt. Mit Net.Data SQL Assist können Sie die folgenden Funktionen ausführen:

- Erstellen von SELECT-, INSERT-, UPDATE- und DELETE-Anweisungen (einschließlich SELECT DISTINCT)
- Erstellen von Mehrfachbedingungen mit Hilfe von Suchfunktionen, AND oder OR und typenabhängigen Eingabefeldern
- Definieren von Tabellenverknüpfungen (innere, rechte erweiterte und linke erweiterte)
- Auswählen der anzuzeigenden Spalten
- Auswählen der Sortierreihenfolge
- Eingeben von benutzerdefinierten Variablen zur Verwendung in Bedingungen, Werten und Sortiervorgängen

Nach dem Erstellen der SQL-Anweisung können Sie die folgenden Funktionen ausführen:

- Sichern der SQL-Anweisung als Datei
- Generieren und Sichern eines Makros mit der SQL-Anweisung
- Kopieren der SQL-Anweisung bzw. des Makros in die Zwischenablage

In diesem Anhang werden die folgenden Themen behandelt:

Vorbereitung

Zur Ausführung von Net.Data SQL Assist muss folgende Software installiert sein:

- Java Development Kit (JDK) oder Java Runtime Environment (JRE) 1.1.x
- Eine JDBC-fähige Datenbank

Weitere Einzelangaben zum Zugreifen auf die Datenquelle über JDBC und zum Start anderer eventuell erforderlicher Server finden Sie in der Dokumentation zu Ihrer Datenbank. Wenn Sie z. B. auf eine Datenquelle von DB2 UDB Version 5.0 fern zugreifen wollen, muss auf dem Datenbank-Server der JDBC-Server (db2jstrt) aktiv sein.

Ausführen von Net.Data SQL Assist

Net.Data SQL Assist wird von der Befehlszeile aus gestartet und ist in der Datei {*inst_dir*}/assist/NetDataAssist.jar enthalten.

Gehen Sie wie folgt vor, um Net.Data Assist mit Java Development Kit (JDK) zu starten:

Geben Sie den folgenden Befehl ein, um Net.Data Assist zu starten:

```
java -classpath %CLASSPATH%;{inst_dir}/assist/NetDataAssist.jar NetDataAssist
```

Gehen Sie wie folgt vor, um Net.Data Assist mit Java Runtime Environment (JRE) zu starten:

Geben Sie den folgenden Befehl ein, um Net.Data Assist zu starten:

```
jre -cp %CLASSPATH%;{inst_dir}/assist/NetDataAssist.jar NetDataAssist
```

Klicken Sie den Knopf **Weiter** an, um durch die Fenster zum Anmelden, Erstellen einer SQL-Anweisung und Generieren eines Net.Data-Makros zu navigieren.

Anhang E. Verwenden von Plug-Ins für NetObjects Fusion (NOF) mit Net.Data-Servlets

Net.Data stellt ein Plug-In für NetObjects Fusion für die Net.Data-Servlets bereit.

Sie können NetObjects Fusion (NOF) zur Integration Ihrer Net.Data-Makros verwenden. NOF bietet bessere Integration in Ihre Websiteverwaltung und eine benutzerfreundliche grafische Benutzerschnittstelle.

In diesem Anhang werden die folgenden Themen behandelt:

Informationen zum Plug-In für NetObjects Fusion

Das Plug-In NetDataServlet.NFX verwendet die Net.Data-Servlets. Dieses NOF-Plug-In unterstützt den Aufruf eines vorhandenen Net.Data-Makros bzw. einer einzelnen Net.Data-Funktion. Das Plug-In generiert HTML zum Aufrufen von Net.Data als Servlet oder SSI (Server-Side-Include). Wenn der Webserver Net.Data aufruft, wird das Net.Data-Makro bzw. die Net.Data-Funktion ausgeführt. Betten Sie mit dem Net.Data-Servlet-Plug-In ein beliebiges Makro- und Funktions-Servlet in eine über NOF verwaltete Website ein. Dies wird in Abb. 30 auf Seite 260 beschrieben. Das Plug-In stellt viele der grundlegenden Makro- bzw. Funktionsparameter bereit, die standardmäßig so eingestellt sind, dass das Erstellen eines Makros automatisch erfolgt. Sie müssen NOF und die Plug-In-Dateien installieren und konfigurieren, um das Plug-In verwenden zu können.

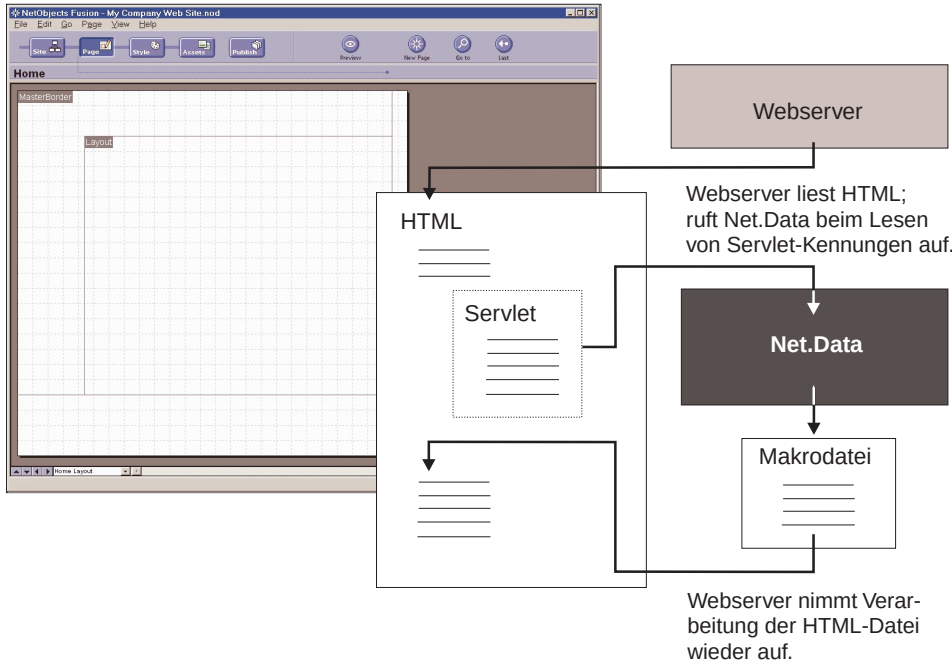


Abbildung 30. Net.Data-Servlets-Plug-Ins

Installieren des Plug-Ins für NetObjects Fusion

Hardware- und Softwarevoraussetzungen:

Für die NOF-Plug-Ins ist NetObjects Fusion Version 2.0 oder höher erforderlich.

Installation: Kopieren Sie <inst_dir>\fx\NetDataServlet.nfx und <inst_dir>\fx\NetDataServlet.gif in das Verzeichnis <NetObjects fusion>\components\.

Konfigurieren des Net.Data-Plug-Ins für NetObjects Fusion

Sie können die Merkmale des Servlets, mit dem Sie arbeiten, mit NOF ändern.

1. Öffnen Sie NetObjects Fusion.
2. Wählen Sie in der Funktionspalette von NetObjects Fusion (NOF) den Knopf **NetObjects Components** aus: Die Plug-In-Knöpfe werden am unteren Rand der Funktionspalette angezeigt.

3. Wählen Sie aus diesen sechs Knöpfen in der Funktionspalette den Knopf **NetObjects Components** aus:
4. Markieren Sie auf der Arbeitsoberfläche von NOF den Bereich, in den Sie das ausgewählte Plug-In stellen wollen. Hier werden die Ergebnisse des Servlets angezeigt. Das Fenster **Installed Components** wird geöffnet. Es wird eine Liste mit Plug-Ins angezeigt, aus denen Sie auswählen können. Wenn sich das Servlet-Plug-In nicht in der Liste befindet, geben Sie mit Hilfe der Felder für Pfad und Dateiname den folgenden Plug-In-Dateinamen zur Verwendung mit dem Makro- oder Funktions-Servlet an: `NetDataServlet.NFX`.
5. Wählen Sie das Servlet-Plug-In in der Liste aus, und klicken Sie den Druckknopf **OK** an.

   Das Plug-In wird zu einem Objekt auf der Arbeitsoberfläche von NOF.

Ändern der Plug-In-Merkmale

Sie können die Makro- und Funktions-Servlets mit dem Net.Data-Servlet-Plug-In ändern.

Gehen Sie wie folgt vor, um das Net.Data-Servlet mit NOF zu ändern:

1. Markieren Sie auf der Arbeitsoberfläche von NOF den Bereich, in den Sie das ausgewählte Plug-In stellen wollen. Hier werden die Ergebnisse des Servlets angezeigt. Das Fenster **Installed Components** wird geöffnet. Es wird eine Liste mit Plug-Ins angezeigt, aus denen Sie auswählen können. Wenn sich das Servlet-Plug-In nicht in der Liste befindet, geben Sie mit Hilfe der Felder für Pfad und Dateiname den folgenden Plug-In-Dateinamen zur Verwendung mit dem Makro- oder Funktions-Servlet an: `NetDataServlet.NFX`.
2. Wählen Sie das Net.Data-Servlet-Plug-In in der Liste aus, und klicken Sie den Druckknopf **OK** an.

   Das Plug-In wird zu einem Objekt auf der Arbeitsoberfläche von NOF.
3. Wählen Sie die Merkmale des Net.Data-Servlet-Plug-Ins aus, und passen Sie sie an:
   a. Wählen Sie das Net.Data-Servlet-Plug-In auf der Arbeitsoberfläche von NOF aus. Die NOF-Palette **Properties** wird geöffnet und zeigt die Plug-In-Merkmale wie in Abb. 31 auf Seite 262 an.

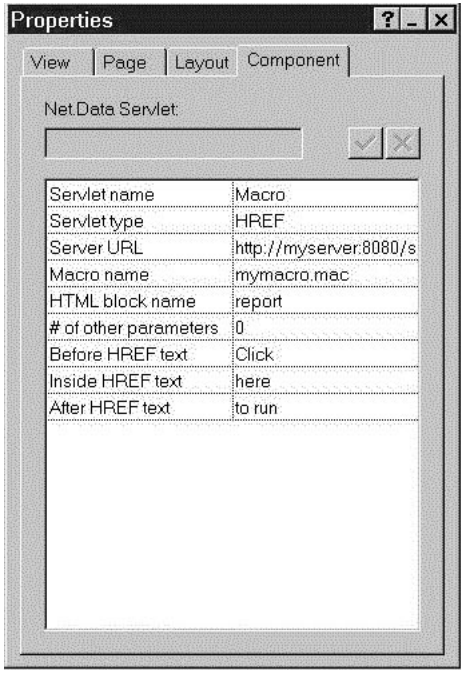


Abbildung 31. Die Net.Data-Servlet-Palette "Properties"

Merkmale für die Net.Data-Servlets:

Sie können die folgenden Merkmale anpassen:

Servlet name (Servlet-Name)
Wählen Sie den Namen des gewünschten Servlets aus: Function oder Macro. Je nach ausgewähltem Servlet-Namen werden verschiedene Merkmale angezeigt.

Servlet type (Servlet-Art)
Wählen Sie die gewünschte Servlet-Art aus: SSI, HREF oder FORM Submit Button. Je nach ausgewählter Servlet-Art werden verschiedene Merkmale angezeigt.

Submit label (Übergabebezeichnung)
Wenn Sie die Art FORM Submit Button auswählen, müssen Sie den Text für die Übergabebezeichnung angeben. Ansonsten wird dieses Merkmal nicht angezeigt.

Server URL (Server-URL-Adresse)
Wenn Sie die Servlet-Art HREF auswählen, müssen Sie die Server-URL-Adresse für den Servlet-fähigen Webserver angeben. Wenn Sie SSI auswählen, wird dieses Merkmal nicht angezeigt.

Macro name (Makroname)
Wenn Sie den Makro-Servlet-Namen auswählen, müssen Sie den Namen eines vorhandenen auszuführenden Net.Data-Makros angeben. Ansonsten wird dieses Merkmal nicht angezeigt.

HTML block name (HTML-Blockname)
Wenn Sie den Makro-Servlet-Namen auswählen, müssen Sie den Namen des HTML-Blocks im auszuführenden Net.Data-Makro angeben. Ansonsten wird dieses Merkmal nicht angezeigt.

Function type (Funktionsart)
Wenn Sie den Funktions-Servlet-Namen auswählen, müssen Sie die auszuführende Funktionsart auswählen: Function oder SQL. Ansonsten wird dieses Merkmal nicht angezeigt.

Language Env (Sprachumgebung)
Wenn Sie den Funktions-Servlet-Namen auswählen, müssen Sie die zu verwendende Net.Data-Sprachumgebung angeben. Ansonsten wird dieses Merkmal nicht angezeigt.

Statement (Anweisung)
Wenn Sie den Funktions-Servlet-Namen auswählen, müssen Sie die auszuführende Anweisung angeben. Ansonsten wird dieses Merkmal nicht angezeigt.

Database (Datenbank)
Wenn Sie den Funktions-Servlet-Namen auswählen, müssen Sie den Namen der zu verwendenden Datenbank angeben. Ansonsten wird dieses Merkmal nicht angezeigt.

# of other parameters (Anzahl anderer Parameter)
Geben Sie die Anzahl anderer, an Net.Data zu übermittelnder Parameter an (Maximum: 25). Geben Sie für jeden Parameter seinen Namen und Wert ein (optional).

Before HREF text (Text vor HREF)
Wenn Sie ein Servlet der Art HREF auswählen, geben Sie den Text an, der vor dem Text vor dem HTML-Befehl `<a href>` angezeigt werden soll. Ansonsten wird dieses Merkmal nicht angezeigt (optional).

Inside HREF text (Text innerhalb HREF)
Wenn Sie ein Servlet der Art HREF auswählen, geben Sie den Text an, der innerhalb des HTML-Befehls `<a href>` angezeigt werden soll. Ansonsten wird dieses Merkmal nicht angezeigt (optional).

After HREF text (Text nach HREF)
Wenn Sie ein Servlet der Art HREF auswählen, geben Sie den Text an, der nach dem Text nach dem HTML-Befehl `<a href>` angezeigt werden soll. Ansonsten wird dieses Merkmal nicht angezeigt (optional).

SQL Reminder! (SQL-Hinweis!)
Wenn Sie ein Servlet der Art HREF auswählen und eine Funktion der Art SQL angeben, wird eine Nachricht angezeigt, die Sie daran erinnert, dass die HREF-SQL-Anweisung für jedes Leerzeichen ( ) ein Pluszeichen (+) enthalten muss. Dieser Text kann nicht geändert werden, und er wird auch nach dem Veröffentlichen der Webseite nicht angezeigt. Ansonsten wird dieses Merkmal nicht angezeigt.

b. Klicken Sie nach dem Definieren der Merkmale für Ihre Seite den Druckknopf **Publish** an, um die Webseiten mit dem NOF-Plug-In für Net.Data-Servlets zu erstellen und zu veröffentlichen.

Anmerkung: Wenn Sie die Servlet-Art SSI auswählen, muss die Dateierweiterung Ihrer Webseite `.shtml` sein. Sie können dies als den Standardwert für Ihre Webseite von der NOF-Palette **Properties** einstellen, indem Sie die Notizbuchindexzunge **Page** auswählen, den Druckknopf **Custom names** anklicken und im Feld **Extension Type** `.shtml` eingeben.

Veröffentlichen von Servlets mit dem NOF-Plug-In

Klicken Sie nach dem Einstellen der Merkmale für Ihre Seite den Druckknopf **Publish** an, um die Webseiten mit dem Plug-In zu erstellen und zu veröffentlichen.

Anhang F. Net.Data-Beispielmakro

Diese Beispielmakroanwendung zeigt eine Liste von Mitarbeiternamen an, aus der der Anwendungsbenutzer zusätzliche Informationen zu einem bestimmten Mitarbeiter abrufen kann, indem er den Namen des Mitarbeiters in der Liste auswählt. Das Makro verwendet die SQL-Sprachumgebung, um die Tabelle EMPLOYEE nach den Mitarbeiternamen und Informationen zu einem bestimmten Mitarbeiter abzufragen.

Das Makro verwendet eine Kopfdatei, die den DEFINE-Block für das Makro enthält.

Abb. 32 auf Seite 267 zeigt das Beispielmakro. Abb. 33 auf Seite 267 zeigt die Kopfdatei.

```
%{********************** Sample Macro ****************************
*   FileName = sqlsamp1.dtw                                       *
*   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 EMPLOYEE
%MESSAGE {
    -204:  {<p><b>ERROR -204: Table EMPLOYEE not found. </b> </p>
           %} : exit
    +default: "WARNING $(RETURN_CODE)" : continue
    -default: "Unexpected ERROR $(RETURN_CODE)" : exit
%}

%REPORT {
<select name="emp_name">
   %ROW{
<option>$(V1)</option>
```

```
%}
</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 EMPLOYEE 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. </p>
<hr />
<form method="post" action="report">
@queryDB()
<input type="submit" value="Select Employee" />
</form>
<hr />
</body>
</html>
%}
```

```
%{*************************************************************************
*  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></p>
<hr /><a href="input">Return to previous page</a>
</body>
</html>
%}

%{     End of Net.Data macro 1 %}
```

Abbildung 32. Beispielmakro

```
==========================================================================
%{************************** Include File *********************************
*   FileName = sqlsamp1.hti                                               *
*   Description:                                                          *
*     This include file provides global DEFINEs for the sqlsamp1.dtw      *
*     Net.Data macro.                                                     *
*************************************************************************%}
%define {
    emp_name   =""
    reposition = sign
 exampleTitle = "Sample Macro"
    %}

%{     End of include file  %}
```

Abbildung 33. Kopfdatei

Bemerkungen

Die vorliegenden Informationen wurden für Produkte und Services entwickelt, die auf dem deutschen Markt angeboten werden. Möglicherweise bietet IBM die in dieser Dokumentation beschriebenen Produkte, Services oder Funktionen in anderen Ländern nicht an. Informationen über die gegenwärtig im jeweiligen Land verfügbaren Produkte und Services sind beim IBM Ansprechpartner erhältlich. Hinweise auf IBM Lizenzprogramme oder andere IBM Produkte bedeuten nicht, dass nur Programme, Produkte oder Dienstleistungen von IBM verwendet werden können. Anstelle der Produkte, Programme oder Dienstleistungen können auch andere ihnen äquivalente Produkte, Programme oder Dienstleistungen verwendet werden, solange diese keine gewerblichen oder andere Schutzrechte verletzen. Die Verantwortung für den Betrieb von Fremdprodukten, Fremdprogrammen und Fremdservices liegt beim Kunden.

Für in diesem Handbuch beschriebene Erzeugnisse und Verfahren kann es IBM Patente oder Patentanmeldungen geben. Mit der Auslieferung dieses Handbuchs ist keine Lizenzierung dieser Patente verbunden. Lizenzanforderungen sind schriftlich an IBM Europe, Director of Licensing, 92066 Paris La Defense Cedex, France, zu richten. Anfragen an obige Adresse müssen auf Englisch formuliert werden. Hinweise auf IBM Lizenzprogramme oder andere IBM Produkte bedeuten nicht, dass nur Programme, Produkte oder Dienstleistungen von IBM verwendet werden können.

Anfragen an obige Adresse müssen auf Englisch formuliert werden.

Trotz sorgfältiger Bearbeitung können technische Ungenauigkeiten oder Druckfehler in dieser Veröffentlichung nicht ausgeschlossen werden. Die Angaben in diesem Handbuch werden in regelmäßigen Zeitabständen aktualisiert. Die Änderungen werden in Überarbeitungen oder in Technical News Letters (TNLs) bekannt gegeben. IBM kann jederzeit Verbesserungen und/oder Änderungen an den in dieser Veröffentlichung beschriebenen Produkten und/oder Programmen vornehmen.

Verweise in diesen Informationen auf Websites anderer Anbieter dienen lediglich als Benutzerinformationen und stellen keinerlei Billigung des Inhalts dieser Websites dar. Das über diese Websites verfügbare Material ist nicht Bestandteil des Materials für dieses IBM Produkt. Die Verwendung dieser Websites geschieht auf eigene Verantwortung.

Lizenznehmer des Programms, die Informationen zu diesem Produkt wünschen mit der Zielsetzung: (i) den Austausch von Informationen zwischen unabhängigen, erstellten Programmen und anderen Programmen (einschließlich des vorliegenden Programms) sowie (ii) die gemeinsame Nutzung der ausgetauschten Informationen zu ermöglichen, wenden sich an folgende Adresse:

IBM Corporation
W92/H3
555 Bailey Avenue
P.O. Box 49023
San Jose, CA 95161-9023
U.S.A.

Anfragen an obige Adresse müssen auf Englisch formuliert werden.

Die Bereitstellung dieser Informationen kann unter Umständen von bestimmten Bedingungen - in einigen Fällen auch von der Zahlung einer Gebühr - abhängig sein.

Die Lieferung des im Handbuch aufgeführten Lizenzprogramms sowie des zugehörigen Lizenzmaterials erfolgt im Rahmen der Allgemeinen Geschäftsbedingungen der IBM, der Internationalen Nutzungsbedingungen der IBM für Programmpakete oder einer äquivalenten Vereinbarung.

Informationen über Produkte anderer Hersteller als IBM wurden von den Herstellern dieser Produkte zur Verfügung gestellt, bzw. aus von ihnen veröffentlichten Ankündigungen oder anderen öffentlich zugänglichen Quellen entnommen. IBM hat diese Produkte nicht getestet und übernimmt im Hinblick auf Produkte anderer Hersteller keine Verantwortung für einwandfreie Funktion, Kompatibilität oder andere Ansprüche.

Die oben genannten Erklärungen bezüglich der Produktstrategien und Absichtserklärungen von IBM stellen die gegenwärtige Absicht der IBM dar, unterliegen Änderungen oder können zurückgenommen werden, und repräsentieren nur die Ziele der IBM.

Diese Veröffentlichung dient nur zu Planungszwecken. Die in dieser Veröffentlichung enthaltenen Informationen können geändert werden, bevor die beschriebenen Produkte verfügbar sind.

Diese Veröffentlichung enthält Beispiele für Daten und Berichte des alltäglichen Geschäftsablaufes. Diese Beispiele enthalten Namen von Personen, Firmen, Marken oder Produkten. Alle diese Namen sind frei erfunden; Ähnlichkeiten mit tatsächlichen Namen und Adressen sind rein zufällig.

Marken

Folgende Namen sind in gewissen Ländern Marken der IBM Corporation:

AIX
AS/400
DB2
DB2 Universal Database
DRDA
DataJoiner
IBM
IMS
Sprachumgebung
MVS/ESA
Net.Data
OS/2
OS/390
OS/400
OpenEdition

Folgende Namen sind Marken anderer Unternehmen:

Java und alle auf Java basierenden Marken und Logos sind in gewissen Ländern Marken von Sun Microsystems, Inc.

UNIX ist in gewissen Ländern eine eingetragene Marke und wird ausschließlich von der X/Open Company Limited lizenziert.

Lotus und Domino Go Webserver sind in gewissen Ländern Marken der Lotus Development Corporation.

Microsoft, Windows, Windows NT und das Logo von Windows sind in gewissen Ländern Marken oder eingetragene Marken der Microsoft Corporation.

Index

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

Z

Antwort

IBM Net.Data für OS/2, Windows NT und UNIX
Verwaltung und Programmierung
Version 7

Anregungen zur Verbesserung und Ergänzung dieser Veröffentlichung nehmen wir gerne entgegen. Bitte informieren Sie uns über Fehler, ungenaue Darstellungen oder andere Mängel.

Zur Klärung technischer Fragen sowie zu Liefermöglichkeiten und Preisen wenden Sie sich bitte entweder an Ihre IBM Geschäftsstelle, Ihren IBM Geschäftspartner oder Ihren Händler.

Unsere Telefonauskunft ″HALLO IBM″ (Telefonnr.: 01803/31 32 33) steht Ihnen ebenfalls zur Klärung allgemeiner Fragen zur Verfügung.

Kommentare:

Danke für Ihre Bemühungen.

Sie können ihre Kommentare betr. dieser Veröffentlichung wie folgt senden:
- Als Brief an die Postanschrift auf der Rückseite dieses Formulars
- Als E-Mail an die folgende Adresse: comment@tcvm.vnet.ibm.com

Name	Adresse
Firma oder Organisation	
Rufnummer	E-Mail-Adresse

Antwort

IBM Deutschland GmbH
SW TSC Germany

70548 Stuttgart