Net.Data

Verwaltung und Programmierung für OS/2, Windows NT und UNIX

Version 6 Release 1

Net.Data

Verwaltung und Programmierung für OS/2, Windows NT und UNIX

Version 6 Release 1

Anmerkung

Vor Verwendung dieser Informationen und des darin beschriebenen Produkts sollten Sie die allgemeinen Informationen in Anhang G, „Bemerkungen“ auf Seite 257 lesen.

Diese Veröffentlichung ist eine Übersetzung des Handbuchs
Net.Data Administration and Programming Guide for OS/2, Windows NT and UNIX,

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 NLS Center
Kst. 2877
Juni 1999

Inhaltsverzeichnis

Vorwort

Vielen Dank, daß Sie sich für Net.Data Version 6.1, das IBM Entwicklungs-Tool zum Erstellen dynamischer Web-Pages, entschieden haben! Mit Hilfe von Net.Data können Sie schnell Web-Seiten 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 von IBM können Sie unter Verwendung von Daten aus relationalen und nichtrelationalen Datenbankverwaltungssystemen (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 Web-Pages erstellen.

Net.Data ist ein Makroumwandler, der als Middleware auf einer Web-Server-Maschine ausgeführt wird. Sie können Net.Data-Anwendungsprogramme (sogenannte *Makros*) schreiben, die von Net.Data interpretiert und für die Erstellung dynamischer Web-Pages 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 Web-Server gesendet, der die Anforderung zur Ausführung an Net.Data weiterleitet. Net.Data lokalisiert das Makro, führt es aus und erstellt eine Web-Seite, die basierend auf den von Ihnen definierten Funktionen angepaßt wird. Diese Funktionen können folgende Aktionen ausführen:

- Einbinden von Geschäftslogik in Perl-Prozeduren, C- und C++-Anwendungen bzw. REXX-Programme
- Zugreifen auf Datenbanken wie DB2
- Zugreifen auf andere Datenquellen wie unstrukturierte Textdateien

Net.Data gibt diese Web-Seite an den Web-Server 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 Industriestandard entsprechende Schnittstelle für Interaktion zwischen einem Browser und einem Web-Server, und CGI ist eine dem Industriestandard entsprechende Schnittstelle für den Web-Server-Aufruf von Gateway-Anwendungen wie Net.Data. Diese Schnittstellen ermöglichen Ihnen die Auswahl Ihres bevorzugten Browsers bzw. Web-Servers zur Verwendung mit Net.Data. Net.Data unterstützt außerdem eine Vielzahl von Web-Server-APIs (Application Programming Interfaces - Anwendungsprogrammierschnittstellen) für verbesserte Leistung.

Die Net.Data-Produktfamilie bietet ein ähnliches Leistungsspektrum unter OS/400, OS/390, Windows NT, AIX, OS/2, HP-UX, Sun Solaris, Linux und SCO (Santa Cruz Operating System). Net.Data unterstützt außerdem FastCGI und die wesentlichen Web-Server-APIs für viele Betriebssysteme.

Ein grafisches Verwaltungs-Tool (Administration Tool) hilft Ihnen bei der Verwaltung von Net.Data-Konfigurationseinstellungen für die Betriebssysteme AIX, Windows NT und OS/2. Administration 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 SmartGuides 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 Web-Entwicklungs-Tools von NetObjects Fusion zum Erstellen komplexer Anwendungen mit dynamischen Daten relationaler Datenquellen. Net.Data-SmartGuides stellen ein grafisches Tool bereit, das Sie durch das Erstellen grundlegender Net.Data-Makros führt.

Neue Funktionen

In den folgenden Abschnitten werden die neuen Erweiterungen für Net.Data beschrieben.

Neue Funktionen im FixPak (Version 2.0.7)

Net.Data Version 2.0.7 wurde wie folgt erweitert:

- Fähigkeit, DB2 File Manager und den Datentyp DATALINK zu verwenden
- Unterstützung für neue Net.Data-Funktionen zur Tabellenverarbeitung
- Konfigurationsvariablen für die Sicherheit: DTW_DIRECT_REQUEST und DTW_SHOWSQL sowie neue Richtlinien zur Verbesserung der Sicherheit Ihrer Anwendungen

Neue Funktionen in Version 6 Release 1

Net.Data für OS/2, Windows NT und UNIX weist in Version 6 Release 1 die folgenden Funktionen auf:

- Die Leistung wurde unter anderem wie folgt erweitert:
  - Unterstützung für FastCGI auf dem Betriebssystem Solaris
- Die Sprachumgebungen (Language Environment) wurden unter anderem wie folgt erweitert:
  - Unterstützung für den Datentyp DATALINK in der SQL-Sprachumgebung unter Windows NT
  - Unterstützung für die IMS-Web-Sprachumgebung auf dem Betriebssystem Solaris
  - Unterstützung für große Objekte (LOBs) in der ODBC-Sprachumgebung
  - Unterstützung für neue LOB-Typen: MIDI-Audiodateien (.mid), AIFF-Audiodateien (.aif), Basisaudiodateien (.au), RA-Audiodateien (Real Audio), Portable Document Format-Dateien (.pdf) und WAV-Dateien (.wav)
  - Vereinfachte Pflege des Verzeichnisses für temporäre große Objekte (`tmplobs`) mit dem Dämon `cleanup`
- Die Makrosprache wurde unter anderem wie folgt erweitert:
  - Unterstützung für Millisekunden in der Funktion DTW_TIME
  - Unterstützung für REPORT-Blöcke in MACRO_FUNCTION-Blöcken
  - Unterstützung für mehrere REPORT-Blöcke in FUNCTION- und MACRO_FUNCTION-Blöcken
  - Unterstützung für das dynamische Erstellen von Variablenverweisen
  - Fähigkeit, das Attribut VALUE im Element OPTION von DTW_SELECT() festzulegen
  - Unterstützung für das Hash-Zeichen (#) in Variablennamen
- Fähigkeit, Direktverbindungsfehler und -aktivitäten zu protokollieren
- Unterstützung von Net.Data auf dem Betriebssystem Linux
- Unterstützung für Unicode-Zeichen in Makrodateien und DB2-Datenbanken mit der Konfigurationsvariablen DTW_UNICODE

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 über das IMS-Web 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.software.ibm.com/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 Web-Server funktioniert, einfache SQL-Anweisungen verstehen und HTML-Befehle, einschließlich HTML-Formularbefehlen, 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.

Einführung

Die meisten Web-Seiten im Internet sind statische Web-Seiten, d. h., sie ändern sich nur, wenn sie von Ihnen editiert werden. Sollen dynamische Daten und Anwendungen (z. B. aktuelle Verkaufsstatistiken) ins Web gestellt werden, schreiben Web-Site-Entwickler in der Regel Programme, die auf dem Web-Server als Middleware ausgeführt werden, um dynamisch Web-Seiten zu erstellen. Das Schreiben solcher Programme ist allerdings nicht ganz einfach.

Net.Data vereinfacht das Schreiben interaktiver Web-Anwendungen durch *Makros*.

In diesem Kapitel wird Net.Data beschrieben und werden die Gründe zur Verwendung für Ihre Web-Anwendung 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-Makrosprachkonstrukten, HTML-Befehlen, JavaScript und Anweisungen einer Sprachumgebung wie SQL und Perl. 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 Web-Server-Programmen, wodurch das Hinzufügen von dynamischen Daten zu statischen Web-Seiten 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 illustriert die Beziehung zwischen Net.Data, dem Web-Server und unterstützten Daten sowie Programmiersprachumgebungen.

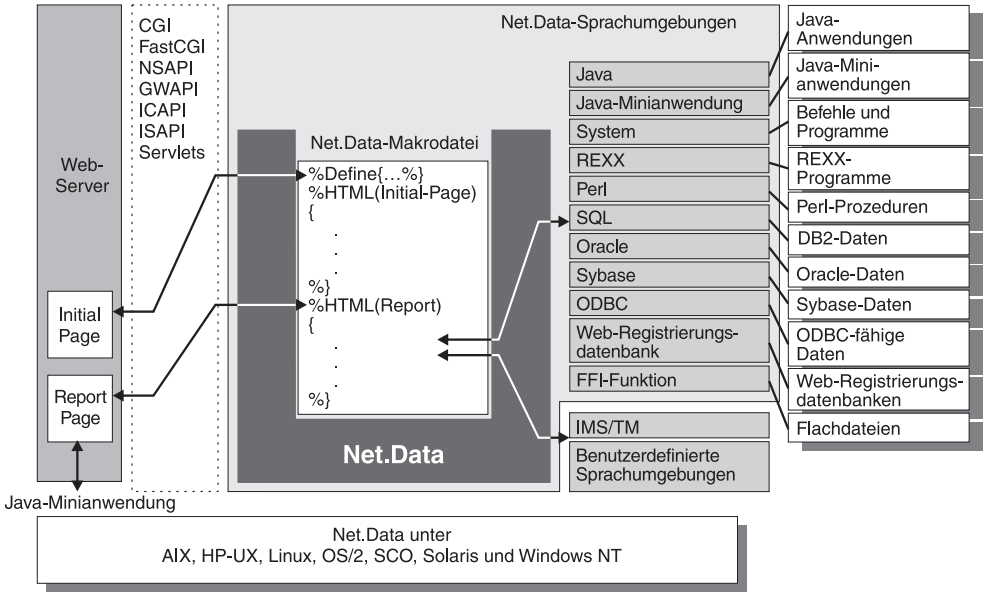


Abbildung 1. Die Beziehung zwischen Net.Data, dem Web-Server und unterstützten Daten sowie Programmquellen

Der Web-Server ruft Net.Data als eine CGI-, FastCGI- oder Web-Server-API auf, indem Net.Data als DLL oder gemeinsam benutzte Bibliothek aufgerufen wird, wenn eine URL-Adresse empfangen wird, die Net.Data-Services anfordert. 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 Web-Seite an den Web-Server 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 Web-Pages, weil die Verwendung der Makrosprache einfacher ist als das Schreiben eigener Web-Server-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 das 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. Die Net.Data-Web-Anwendungsumgebung bietet die folgenden Funktionen:
- In Ihren Web-Anwendungen 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 Web-Pages 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 einer Perl-Prozedur erfordern, braucht kein Makro erstellt zu werden. Diese Anforderungen können direkt in der URL-Adresse angegeben werden, die vom Browser an den Web-Server übergeben wird.

Freies Format

Die Net.Data-Makrosprache weist nur einige wenige Regeln zum Programmformat auf. Diese Unkompliziertheit ermöglicht Programmierern 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 104 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 im Handbuch *Net.Data Reference*.

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 Net.Data konfigurieren und Ihre Konfiguration für den Web-Server ändern. Folgende Konfigurationsaufgaben müssen ausgeführt werden:

- Anpassen der Net.Data-Initialisierungsdatei (INI)
- Konfigurieren von Net.Data zur Verwendung mit FastCGI oder einer der Unterstützungs-Web-Server-APIs (wahlfrei)
- Anpassen der Konfigurations- und Umgebungsvariablendateien für den Web-Server
- Konfigurieren des Cache-Managers (wahlfrei)
- Konfigurieren der Direktverbindung (wahlfrei)
- Definieren der Net.Data-Sprachumgebungen
- Angeben von Zugriffsrechten

Sie können Net.Data mit den folgenden Tools konfigurieren:

- Texteditor

  Bearbeiten Sie die Initialisierungsdatei und die Konfigurationsdatei für Direktverbindungen auf allen Betriebssystemen mit einem Texteditor. Mit einem Texteditor können Sie ferner die Konfigurationsdateien von Web-Servern 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.

A - Kann mit Administration Tool oder manuell konfiguriert werden. **M** - Kann nur manuell konfiguriert werden.'
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|SCO**
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	N/V	N/V	N/V
Aktivieren der Fehlerprotokollierung	A	A	A	M
Konfigurieren des Web-Servers 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 Web-Server bieten Verwaltungs-Tools, mit denen Sie den Web-Server konfigurieren können.

In diesem Kapitel wird beschrieben, wie Sie Net.Data konfigurieren und Ihre Konfiguration des Web-Servers zur Verwendung mit Net.Data ändern können. Außerdem wird beschrieben, wie Sie wahlfreie Komponenten konfigurieren können.

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 DBCS-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 Ausgabeparameterwerte 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-Prozeduren.

Die Net.Data-Initialisierungsdatei (db2www.ini) befindet sich im Dokumentverzeichnis des Web-Servers. Weitere Informationen finden Sie in der Informationsdatei (README) für Ihr Betriebssystem.

Hinweis zu Berechtigungen: Stellen Sie sicher, daß der Web-Server Zugriffsrechte für diese Datei hat. Weitere Informationen hierzu finden Sie in „Erteilen von Zugriffsrechten für Dateien, auf die Net.Data zugreift“ auf Seite 62.

Informationen zu den Net.Data-Konfigurationsdateien für wahlfreie Komponenten

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

Die Konfigurationsdatei für Direktverbindungen

Eine Direktverbindung bietet Verbindungsverwaltung auf den Betriebssystemen Windows NT, OS/2, AIX und Sun Solaris, um die Leistung durch Beseitigen des Systemaufwands beim Start zu steigern. Die Konfigurationsdatei für die Net.Data-Direktverbindung enthält Informationen zu einer oder mehreren benannten Cliettes. Eine Cliette ist ein Langzeitprozeß, der eine Verbindung zu einer Datenbank verwaltet, oder eine Java-Anwendung, 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, eindeutige Anschlüsse 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. Unter AIX können Sie das Kennwort verschlüsseln.

Hinweis zu Berechtigungen: Stellen Sie sicher, daß die Benutzer-ID, mit der Connection Manager gestartet wird, Zugriffsrechte für diese Datei hat. Weitere Informationen hierzu finden Sie in „Erteilen von Zugriffsrechten für Dateien, auf die Net.Data zugreift“ auf Seite 62.

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 200 beschrieben. Das Konfigurieren des Cache-Managers wird in „Konfigurieren des Cache-Managers und der Net.Data-Caches“ auf Seite 208 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 200.

Hinweis zu Berechtigungen: Stellen Sie sicher, daß die Benutzer-ID, mit welcher der Cache-Manager gestartet wird, Zugriffsrechte für diese Datei hat. Weitere Informationen hierzu finden Sie in „Erteilen von Zugriffsrechten für Dateien, auf die Net.Data zugreift“ auf Seite 62.

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 wahlfrei Anschlußnummer- und Einheitennamenvariablen angeben. Diese Werte müssen mit den Werten in der Konfigurationsdatei für den Cache-Manager übereinstimmen, sofern verwendet.
Konfigurationsdatei für Direktverbindungen	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 wahlfrei Anschlußnummer- und Einheitennamenvariablen 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 10 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 auf Seite 11 zeigt einen Ausschnitt der Konfigurationsdatei für Direktverbindungen, die die Cliette-Definitionen für DTW_SQL:SAMPLE und DTW_SQL:CELDIAL enthält.

```
CONNECTION_MANAGER{
    MAIN_PORT=7128
    ADMIN_PORT1=7131
    ADMIN_PORT2=7133
    ENCRYPTION=OFF
    }

    ######################################################################
    # 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:SAMPLE{
    MIN_PROCESS=1
    MAX_PROCESS=3
    START_PRIVATE_PORT=7100
    START_PUBLIC_PORT=7300
    EXEC_NAME=dtwcdb2.exe
    DATABASE=SAMPLE
    LOGIN=USER1
    PASSWORD=HAMLET
    }

    CLIETTE DTW_SQL:CELDIAL{
    MIN_PROCESS=1
    MAX_PROCESS=5
    START_PRIVATE_PORT=7500
    START_PUBLIC_PORT=7700
    EXEC_NAME=dtwcdb2.exe
    DATABASE=CELDIAL
    LOGIN=USER2
    PASSWORD=OPHELIA
    }
```

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 12
- „Pfadkonfigurationsanweisungen“ auf Seite 19
- „Umgebungskonfigurationsanweisungen“ auf Seite 23

Die in Abb. 5 auf Seite 12 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. (Eine Anweisung ENVIRONMENT wird in mehreren Zeilen wiedergegeben, um die Lesbarkeit zu verbessern.) 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  MACRO_PATH c:\DB2WWW\Macro
8  HTML_PATH  c:\www\html
9  INCLUDE_PATH c:\db2www\Macro
10 EXEC_PATH c:\db2www\Macro
11 FFI_PATH c:\pub\ffi;pub\ffi\data
12 ENVIRONMENT (DTW_SQL)          [DLL path]   [Parameter list]
13 ENVIRONMENT (DTW_SYB)          [DLL path]   [Parameter list]
14 ENVIRONMENT (DTW_ORA)          [DLL path]   [Parameter list]
15 ENVIRONMENT (DTW_ODBC)         [DLL path]   [Parameter list]
16 ENVIRONMENT (DTW_DEFAULT)      [DLL path]   [Parameter list]
17 ENVIRONMENT (DTW_APPLET)       [DLL path]   [Parameter list]
18 ENVIRONMENT (DTW_REXX)         [DLL path]   [Parameter list]
19 ENVIRONMENT (DTW_PERL)         [DLL path]   [Parameter list]
20 ENVIRONMENT (DTW_SYSTEM)       [DLL path]   [Parameter list]
21 ENVIRONMENT (DTW_FILE)         [DLL path]   [Parameter list]
22 ENVIRONMENT (DTW_WEBREG)       [DLL path]   [Parameter list]
23 ENVIRONMENT (DTW_JAVAPPS)      [DLL path]   [Parameter list]
24 ENVIRONMENT (HWS_LE)           [DLL path]   [Parameter list]
```

- Die Zeilen 1 - 6 definieren Konfigurationsvariablen.
- Die Zeilen 7 - 11 definieren Pfade zu Dateien, die zum Verarbeiten des Makros erforderlich sind.
- Die Zeilen 12 - 24 definieren die verfügbaren ENVIRONMENT-Anweisungen.

Abbildung 5. Die Net.Data-Initialisierungsdatei

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

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 Web-Seite. 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 wahlfrei, 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

Es werden zwei wahlfreie 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 200.

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_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 67.

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.

DTW_LOG_DIR: Variable für Speicherposition des Fehlerprotokolls

Gibt das Verzeichnis an, in dem die Fehlerprotokolle gespeichert sind. Wenn im Makro über die Variable DTW_LOG_LEVEL das Protokollieren aktiviert ist, werden die Protokolldateien in dem Verzeichnis gespeichert, das in der Pfadanweisung der Variablen DTW_LOG_DIR angegeben ist.

Der Standardwert ist `\inst_dir\logs\netdata.logs`. Informationen zum Protokollieren von Fehlernachrichten mit Net.Data und zur Variablen DTW_LOG_LEVEL finden Sie in „Protokollieren von Net.Data-Fehlernachrichten“ auf Seite 229.

Voraussetzung: Die Variable DTW_LOG_DIR muß für Net.Data definiert werden, um Dateien protokollieren zu können. Wenn diese Variable nicht definiert wird, wird keine Protokollierung durchgeführt, selbst wenn DTW_LOG_LEVEL im Makro auf ERROR bzw. WARNING gesetzt ist.

Syntax:

```
DTW_LOG_DIR \inst_dir\path
```

Beispiel: Konfiguration der Initialisierungsdatei

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

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 DBCS-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 DBCS gemischt mit SBCS	Unterstützt nur SBCS
UTF8	Unterstützt UTF-8	Unterstützt UTF-8

Syntax:

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

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 Domänennamens-Server 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 UTF-16-Daten enthält. Sie werden in UTF-8 umgesetzt.

Wird DTW_UNICODE auf UTF8 gesetzt, wird Net.Data zur Ausführung in einer Unicode-Umgebung angewiesen. Dies bedeutet, daß Net.Data vom UTF-8-Format für die Makrodateidaten, die Formulardateneingabe vom Browser und die von der DB2-Datenbank kommenden Daten ausgeht. Durch das Festlegen dieser Variable wird Net.Data mitgeteilt, daß die Eingabe und Ausgabe für Makros in UTF-8 erfolgt. Net.Data generiert folglich Web-Pages in UTF-8.

Diese Konfigurationsvariable 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. 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 6 veranschaulicht, wie die Einstellungen dieser beiden Variablen festlegen, wie integrierte Funktionen Zeichenfolgen verarbeiten:

Tabelle 6. Beziehung zwischen den Einstellungen von DTW_UNICODE und DTW_MBMODE

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

Syntax:

```
DTW_UNICODE NO|UTF8|UTF16
```

Dabei gilt folgendes:

NO Gibt die Verwendung des Werts für die Variable DTW_MBMODE an. Tabelle 6 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_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.

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. Dieses Verhalten war die Standardeinstellung vor Net.Data Version 2. Wenn Sie diesen Wert angeben, brauchen Sie Makrodateien nicht zu ändern.

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:

- „MACRO_PATH“
- „EXEC_PATH“ auf Seite 20
- „INCLUDE_PATH“ auf Seite 21
- „FFI_PATH“ auf Seite 22
- „HTML_PATH“ auf Seite 23

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 mehrere allgemeine Richtlinien. Ausnahmen werden in den Beschreibungen der einzelnen Pfadanweisungen angemerkt.

- Jedes angegebene Verzeichnis wird durch ein Semikolon (;) beendet.
- Schrägstriche (/) und umgekehrte Schrägstriche (\) werden auf gleiche Weise behandelt.
- 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 Web-Anwendungen 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.

MACRO_PATH

Die Konfigurationsanweisung MACRO_PATH 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.d2w` angefordert:

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

Syntax:

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

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

Net.Data fügt den Pfad `/macro/sqlm.d2w` an die Pfade in der Konfigurationsanweisung MACRO_PATH von links nach rechts an, bis das Net.Data-Makro gefunden

wird bzw. alle Pfade durchsucht worden sind. Informationen zum Aufrufen von Net.Data-Makros finden Sie in „Aufrufen von Net.Data“ auf Seite 73.

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

Net.Data-Initialisierungsdatei:

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

HTML-Programmverbindung (Link):

```
<A HREF="http://server/cgi-bin/db2www/query.d2w/input">Submit
another query.</A>
```

Wenn die Datei `query.d2w` im Verzeichnis `/u/user1/macros` gefunden wird, ist der vollständig qualifizierte Pfad `/u/user1/macros/query.d2w`.

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`

EXEC_PATH

Die Konfigurationsanweisung EXEC_PATH gibt ein oder mehrere Verzeichnisse an, die 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.

INCLUDE_PATH

Die Konfigurationsanweisung INCLUDE_PATH gibt ein oder mehrere Verzeichnisse an, die 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
```

Hinweis: Wenn Sie HTML-Dateien von einem lokalen Web-Server verwenden, verwenden Sie das Konstrukt INCLUDE_URL, so wie im Beispiel für den lokalen Web-Server für INCLUDE_URL im Handbuch *Net.Data Reference* gezeigt. Bei Verwendung der gezeigten Syntax brauchen Sie INCLUDE_PATH nicht zu aktualisieren, um Verzeichnisse anzugeben, die dem Web-Server bereits bekannt sind.

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.

FFI_PATH

Die Konfigurationsanweisung FFI_PATH gibt ein oder mehrere Verzeichnisse an, in denen 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*.

HTML_PATH

Die Konfigurationsanweisung HTML_PATH gibt an, in welches Verzeichnis Net.Data große Objekte (LOBs) schreibt. Für diese Pfadanweisung ist nur ein Verzeichnispfad zulässig.

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 Anweisung HTML_PATH in der Initialisierungsdatei.

Net.Data-Initialisierungsdatei:

```
HTML_PATH /db2/lobs
```

Wenn eine Abfrage ein LOB zurückgibt, sichert Net.Data es in dem in der in der Konfigurationsanweisung HTML_PATH angegebenen Verzeichnis.

Hinweis zur Leistung: 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 150.

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 „Verwenden der Sprachumgebungen“ auf Seite 143 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 ent-

weder der Variablen mit der Funktion DTW_ASSIGN() einen Wert zuordnen oder die Variable in einem DEFINE-Abschnitt definieren.

Wichtig: Wird eine Makrovariable 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, SCO und LINUX wird der Name der gemeinsam benutzten Bibliothek mit der Erweiterung *.so* angegeben.

  – Unter OS/2 und Windows NT wird der DLL-Name ohne die 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. Wenn eine Funktion ihre Ausgabeparameter ändert, behalten die Parameter ihre geänderten Werte nach der Beendigung der Funktion bei. Die folgende Liste gibt an, welche Variablen die Anweisungen ENVIRONMENT übergeben können:

DTW_SQL: TRANSACTION_SCOPE, LOCATION, DB2SSID, DB2PLAN

DTW_ODBC: TRANSACTION_SCOPE, LOCATION

- *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 wahlfrei und können nur für Sprachumgebungen von Datenbanken und Java-Anwendungen und für benutzerdefinierte Sprachumgebungen verwendet werden, für die Cliettes geschrieben wurden.

  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 59.

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

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

Einrichten der 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 IMS-Web-Sprachumgebung“
- „Definieren der Java-Sprachumgebung“ auf Seite 27
- „Definieren der Oracle-Sprachumgebung“ auf Seite 28
- „Definieren der Sybase-Sprachumgebung“ auf Seite 31

Definieren der IMS-Web-Sprachumgebung

Sie müssen die folgenden Schritte ausführen, um die IMS-Web-Sprachumgebung verwenden zu können:

1. Installieren Sie die IMS Web Runtime-Komponente auf dem Web-Server, auf dem Net.Data ausgeführt wird. Informationen zum Installieren und Verwenden der IMS Web Runtime-Komponente finden Sie im Handbuch *IMS Web User's Guide* unter folgender Adresse:

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

2. Erstellen Sie die Transaktions-DLL.

   a. Generieren Sie mit IMS Web Studio den C++-Code, die Makefile (DTW*proj*.mak) und Net.Data-Makrodateien (DTW*proj*.d2w) für Ihre Transaktion.

b. Wenn Sie Net.Data auf einem Betriebssystem ausführen, das sich von dem Betriebssystem unterscheidet, auf dem IMS Web Studio ausgeführt wird, versetzen Sie die DLL-Quellendateien auf eine IMS-Web-Entwicklungsmaschine für das Zielbetriebssystem. Wenn Sie z. B. IMS Web Studio unter Windows NT ausführen und die Zielplattform AIX oder OS/390 ist, versetzen Sie die Quelle für die Transaktions-DLL-Datei auf eine IMS-Web-Entwicklungsmaschine mit AIX bzw. OS/390.

c. Erstellen Sie die ausführbare Form der Transaktions-DLL-Datei mit der generierten Makefile.

3. Kopieren Sie die Transaktions-DLL-Datei (DTW*proj*.dll) und die Net.Data-Makrodatei (DTW*proj*.d2w) auf Ihren Web-Server.

   a. Stellen Sie das Makro in ein Verzeichnis, von dem Net.Data Makros abruft. (Weitere Informationen finden Sie in „MACRO_PATH“ auf Seite 19.)

   b. Stellen Sie die Transaktions-DLL-Datei bzw. gemeinsam benutzte Bibliothek in ein Verzeichnis, aus dem der Web-Server DLL-Dateien bzw. gemeinsam benutzte Bibliotheken abruft.

4. Verwenden Sie die Programmverbindung (Link) namens DTW*proj*.htm in der durch IMS Web Studio generierten Beispieldatei, um eine HTML-Datei in der HTML-Baumstruktur Ihres Web-Servers zu ändern.

   Mit dieser Programmverbindung (Link) können Sie dann Net.Data aufrufen und das HTML-Eingabeformular anzeigen, um die IMS-Web-Sprachumgebung aufzurufen. Die IMS-Web-Sprachumgebung ruft anschließend die IMS-Transaktions-DLL auf, die die von den IMS-Web-Runtime-DLLs geprüften Services verwendet, um die Transaktion auszuführen und ihre Ausgabe an den Web-Browser zurückzugeben.

   Die IMS-Web-Runtime-DLLs für das IMS-Web formulieren und senden eine Anforderungsnachricht über IMS TOC an OTMA. Hierdurch wird die entsprechende Transaktion in die Warteschlange eingereiht. Die Ausgabe der Transaktion wird im Anschluß von OTMA über IMS TOC an das IMS-Web zurückgegeben. Die Transaktion wird abschließend über die IMS-Web-Sprachumgebung zur Anzeige im Web-Browser wieder an Net.Data übergeben.

Definieren der Java-Sprachumgebung

Für die Java-Sprachumgebung sind einige zusätzliche Konfigurationsschritte erforderlich, bevor Sie Funktionen von einem Makro aus aufrufen können:

1. Erstellen Sie eine Stapeldatei zum Starten der Java-Anwendung, denn Net.Data kann eine Java-Anwendung nicht direkt starten. 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 eindeutige Anschlußnummern für die Cliette und den Namen der zugehörigen Stapeldatei in der Konfigurationsvariablen EXEC_NAME 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              <= Erforderlich: Der Wert muß 1 sein, weil es sich
                              bei JAVAPPS um eine Multi-Thread-Cliette handelt.
MAX_PROCESS=1              <= Erforderlich: Der Wert muß 1 sein, weil es sich
                              bei JAVAPPS um eine Multi-Thread-Cliette handelt.
START_PRIVATE_PORT=5100    <= Muß eine eindeutige Anschlußnummer sein.
START_PUBLIC_PORT=5300     <= Muß eine eindeutige Anschlußnummer sein.
EXEC_NAME=launchjv.bat     <= Der Name der Stapeldatei mit den
                              Klassenpfadanweisungen
}
```

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 Pfadanweisung DTW_JAVAPPS ENVIRONMENT in der Net.Data-Initialisierungsdatei `db2www.ini`, indem Sie der Anweisung jeden 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 der auch von Ihrem Web-Server verwendeten Sicherheitsberechtigung verwendet werden kann. Melden Sie sich dazu mit der Benutzer-ID Ihres Web-Servers 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 Web-Server 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 Web-Server verwendeten Sicherheitsberechtigung zugegriffen werden kann. Geben Sie dazu mit Hilfe des Zeilenbefehl-Tools SQL*Plus eine SQL-Anweisung SELECT mit der Berechtigung Ihres Web-Servers 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 Web-Server 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 Web-Server-Prozeß ordnungsgemäß eingestellt sind.

   - Nehmen Sie unter AIX die folgenden Zeilen in die Datei `/etc/environment` 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:

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

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

4. Testen Sie Ihre Konfiguration, indem Sie eine CGI-Shell-Prozedur ausführen, um sicherzustellen, daß von Ihrem Web-Server auf das Oracle-Exemplar zugegriffen werden kann. Siehe folgendes Beispiel:

   ```
   #! /bin/sh
   echo "content-type; text/html
   echo
   echo "< html>< pre>"
   set
   echo "</pre>< 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)
%}
```

Definieren der Sybase-Sprachumgebung

Gehen Sie wie folgt vor, um von Net.Data auf Sybase zuzugreifen:

1. Stellen Sie wie folgt sicher, daß die erforderlichen Sybase-Komponenten installiert sind und ordnungsgemäß funktionieren:

   a. Installieren Sie Sybase Open Client auf der Maschine, auf der Net.Data installiert ist, sofern dieses Produkt nicht bereits installiert ist. Weitere Informationen hierzu finden Sie in der Dokumentation zu Sybase Open Client.

   b. Prüfen Sie, ob die Sybase-Funktion *ping* mit der auch von Ihrem Web-Server verwendeten Sicherheitsberechtigung verwendet werden kann. Melden Sie sich dazu mit der Benutzer-ID Ihres Web-Servers an, und geben Sie folgendes ein:

      ```
      ping sybase-instance-name
      ```

      Dabei gilt folgendes: *sybase-instance-name* ist der Name des Sybase-Systems, auf das Ihre Net.Data-Makros zugreifen.

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

   c. Prüfen Sie, ob auf die Sybase-Tabellen mit der auch vom Web-Server verwendeten Sicherheitsberechtigung zugegriffen werden kann. Geben Sie dazu mit Hilfe des Zeilenbefehl-Tools ISQL eine SQL-Anweisung SELECT mit der Berechtigung Ihres Web-Servers ein, um auf eine Sybase-Tabelle zuzugreifen. Beispiel:

      ```
      SELECT * FROM tablename
      ```

      Sie sind eventuell nicht in der Lage, den Tabellenzugriff unter Windows NT zu prüfen, wenn Ihr Web-Server 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 Sybase-Konfiguration.

2. Stellen Sie sicher, daß die Sybase-Umgebungsvariablen in Ihrem Web-Server-Prozeß ordnungsgemäß eingestellt sind.

   - Nehmen Sie unter AIX die folgenden Zeilen in die Datei `/etc/environment` auf:

     ```
     DSQUERY=sybase-instance-name
     SYBASE=sybase-runtime-library-directory
     ```

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

     ```
     DSQUERY=sybase-instance-name
     SYBASE=sybase-runtime-library-directory
     ```

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

3. Testen Sie die Verbindung von Net.Data zu Sybase. Geben Sie in Ihrem Net.Data-Makro die entsprechenden Werte in den Variablen LOGIN, PASSWORD und DATABASE an. Im folgenden Beispiel werden Verbindungsanweisungen in einem Makro illustriert:

   ```
   %DEFINE DATABASE=database-name
   %DEFINE LOGIN=user_ID@remote-sybase-instance-name
   %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:

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

4. Testen Sie Ihre Konfiguration, indem Sie eine CGI-Shell-Prozedur ausführen, um sicherzustellen, daß von Ihrem Web-Server auf das Sybase-Exemplar zugegriffen werden kann. Siehe folgendes Beispiel:

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

   Fehlerbehebung:

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

   - Überprüfen Sie Ihre Sybase-Konfiguration.
   - Prüfen Sie, ob die Sybase-Umgebungsvariablensyntax korrekt ist und keine Variablen fehlen.
   - Überprüfen Sie die Sybase-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 Sybase-Sprachumgebung mit Funktionen im Makro tätigen. Siehe folgendes Beispiel:

```
%function(DTW_SYB) 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 verwendete Cliettes. Informationen zur Direktverbindung finden Sie in „Verwalten von Verbindungen“ auf Seite 196.

Die in Abb. 6 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  ADMIN_PORT1=7101
 4  ADMIN_PORT2=7102
 5  }
 6
 7  CLIETTE DTW_SQL:CELDIAL{
 8  MIN_PROCESS=1
 9  MAX_PROCESS=5
10  START_PRIVATE_PORT=7200
11  START_PUBLIC_PORT=7210
12  EXEC_NAME=./dtwcdb2
13  DATABASE=CELDIAL
14  LOGIN=marshall
15  PASSWORD=stlpwd
16  }
17
18  CLIETTE DTW_JAVAPPS{
19  MIN_PROCESS=1
20  MAX_PROCESS=5
21  START_PRIVATE_PORT=7300
22  START_PUBLIC_PORT=7310
23  EXEC_NAME=./javaapp
24  }
```

- Die Zeilen 1 - 5 sind für die Konfigurationsdatei erforderlich und definieren eindeutige, mit der Direktverbindung zu verwendende Anschlußnummern.
- Die Zeilen 7 - 16 definieren alle Datenbank-Cliettes, wobei der Cliette-Name, die Anzahl der auszuführenden Prozesse, der Datenbankname, die Anschlußnummern 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. Diese zusätzlichen Werte werden in den Zeilen 13 - 15 gezeigt.
- Die Zeilen 19 - 25 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.

Gehen Sie wie folgt vor, um Anschlüsse für Direktverbindungen zu konfigurieren:

1. Öffnen Sie die Konfigurationsdatei `dtwcm.cnf` mit einem Editor.
2. Konfigurieren Sie die drei Anschlußnummern für Direktverbindungen:
   - MAIN_PORT
   - ADMIN_PORT1
   - ADMIN_PORT2

   Abb. 6 auf Seite 33 zeigt die Standardanschlußnummern. Wenn diese Nummern nicht eindeutig sind, müssen Sie sie in eindeutige Anschlußnummern ändern.
3. **Wichtig:** Stellen Sie sicher, daß der Wert von MAIN_PORT mit dem Wert von DTW_CM_PORT in der Net.Data-Initialisierungsdatei übereinstimmt.

Gehen Sie wie folgt vor, um die Datenbank-Cliettes zu konfigurieren:

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 59.

   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 wahlfrei, 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. Die Werte, die Sie verwenden, können sich auf die Leistung auswirken. Sie können sie jedoch zu einem späteren Zeitpunkt ändern.

   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. Legen Sie die Anschlußnummern fest, die auf Ihrem System für die Datenbank-Cliette verwendet werden sollen. Diese Nummern müssen eindeutig sein, um einen Konflikt mit den für den Cache-Manager und andere Anwendungen verwendeten Anschlußnummern zu vermeiden. Jede Cliette verwendet zwei Anschlüsse. Wenn Sie eine Gruppe von Anschlüssen angeben, müssen Sie den Bereich der zu verwendenden Anschlußnummern angeben. Die ersten beiden Werte sind START_PUBLIC_PORT und START_PRIVATE_PORT. Der andere ist MAX_PROCESS, der die maximale Anzahl von Cliettes angibt. Das folgende Beispiel zeigt, welche Anschlußnummern verwendet werden sollen.

   ```
   START_PUBLIC_PORT=1000
   START_PRIVATE_PORT=1010
   MAX_PROCESS=5
   ```

   Im Beispiel werden die folgenden Anschlüsse verwendet:

1000	1010
1001	1011
1002	1012
1003	1013
1004	1014

   Ein häufiger Fehler ist die überlappende Verwendung der Anschlußnummern durch zwei Cliette-Gruppen oder Überlappung mit den Anschlußnummern des Cache-Managers. Wenden Sie sich an Ihren Systemadministrator, um sicherzustellen, daß die zu verwendenden Anschlußnummern verfügbar sind. Die Informationsdatei (README) für Ihr Betriebssystem enthält allgemeine Richtlinien zu den Anschlüssen, die für Ihr Betriebssystem gültig sind.

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

   ```
   EXEC_NAME=./dtwcxxx
   ```

   Dabei gilt folgendes: *xxx* ist die Kennung der Datenbankart. Die Namen der gültigen ausführbaren Dateien finden Sie in Tabelle 7 auf Seite 36:

Tabelle 7. Namen ausführbarer Cliette-Dateien

Cliette-Beschreibung	Cliette-Typ	Namen		Plattformverfügbarkeit					
		UNIX	Windows NT oder OS/2	AIX	NT	OS/2	HP	SUN	SCO
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
Sybase-Prozeß-Cliette	DTW_SYB	dtwcsyb	dtwcsyb.exe	J	J	N	N	N	N
Oracle-Prozeß-Cliette	DTW_ORA	dtwcora	dtwcora.exe	J	J	N	N	N	N

5. 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.

6. Wahlfrei: Ändern Sie die Standardwerte für die Variablen LOGIN und PASSWORD, 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 33 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.

Gehen Sie wie folgt vor, um die Java-Anwendungs-Cliettes zu konfigurieren:

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 im Anschluß daran simultane Prozesse ankommen, startet Connection Manager weitere Cliettes. Bei Bedarf wird jeweils eine Cliette hinzugefügt, bis der für MAX_PROCESS angegebene Wert erreicht ist. Die Werte, die Sie verwenden, können sich auf die Leistung auswirken. Sie können sie jedoch zu einem späteren Zeitpunkt ändern.

   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.

3. Legen Sie die Anschlußnummern fest, die auf Ihrem System für die Datenbank-Cliette verwendet werden sollen. Diese Nummern müssen eindeutig sein, um einen Konflikt mit den für den Cache-Manager und andere Anwendungen verwendeten Anschlußnummern zu vermeiden. Jede Cliette verwendet zwei Anschlüsse. Wenn Sie eine Gruppe von Anschlüssen angeben, müssen Sie den Bereich der zu verwendenden Anschlußnummern angeben. Die ersten beiden Werte sind START_PUBLIC_PORT und START_PRIVATE_PORT. Der andere ist MAX_PROCESS, der die maximale Anzahl von Cliettes angibt. Das folgende Beispiel zeigt, welche Anschlußnummern verwendet werden sollen.

   ```
   START_PUBLIC_PORT=1000
   START_PRIVATE_PORT=1010
   MAX_PROCESS=5
   ```

Im Beispiel werden die folgenden Anschlüsse verwendet:

1000	1010
1001	1011
1002	1012
1003	1013
1004	1014

Ein häufiger Fehler ist die überlappende Verwendung der Anschlußnummern durch zwei Cliette-Gruppen oder Überlappung mit den Anschlußnummern des Cache-Managers. Wenden Sie sich an Ihren Systemadministrator, um sicherzustellen, daß die zu verwendenden Anschlußnummern verfügbar sind. Die Informationsdatei (README) für Ihr Betriebssystem enthält allgemeine Richtlinien zu den Anschlüssen, die für Ihr Betriebssystem gültig sind.

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 DB2-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 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 einen SQL-Abschnitt 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 statt dessen mit einer DLL-Version.

Gehen Sie wie folgt vor, um Connection Manager automatisch als einen Windows NT-Service zu starten:

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.

Hinweis: 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 Web-Servers zur Verwendung mit CGI

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

Konfigurieren Sie den Web-Server 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 Web-Server 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.

Pass-Anweisungen

Pass-Anweisungen werden von Net.Data nicht verwendet. Wenn Sie Ihre URL-Adresse vereinfachen möchten, können Sie die Anweisung MACRO_PATH in einer Net.Data-Initialisierungsdatei verwenden (siehe dazu „MACRO_PATH“ auf Seite 19).

Konfigurieren von Net.Data für FastCGI

Net.Data kann als ein FastCGI-Prozeß auf Apache Web Server und Domino Go Webserver ausgeführt werden. FastCGI verbindet die Leistungsstärke anderer Web-API-Programme mit der Zuverlässigkeit von CGI. FastCGI wird auf den Betriebssystemen AIX und Sun Solaris unterstützt.

Vorbereitung:

Stellen Sie vor der Verwendung von FastCGI sicher, daß Sie die erforderlichen Produkte installiert haben:

- **Für Apache:** Laden Sie Apache Web Server 1.2.0 oder höher herunter, und installieren Sie das Produkt.
- **Für Domino Go Webserver:** Laden Sie Domino Go Webserver für AIX oder Sun von folgender Adresse herunter, und installieren Sie das Produkt:

  `http://www.ics.raleigh.ibm.com/dominowebserver`

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

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

   Für Apache Web Server:

   Aktualisieren Sie die Datei `http.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 SYBASE=sybase_path
     -initial-env DSQUERY=sybase_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 Domino Go Webserver:

   Aktualisieren Sie die Dateien `httpd.conf` und `fcgi.conf`:

   - Deklarieren Sie in der Datei `httpd.conf` den Serviceabschnitt:

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

- Deklarieren Sie die Anwendung in der Datei `fcgi.conf`:

```
Local {
Exec inst_dir
Role Responder
URL /fcgi-bin/db2www
BindPath /tmp/db2www.ibm
NumProcesses proc_num
Environ LIBPATH=libpath
Environ ORACLE_HOME=oracle_path
Environ ORACLE_SID=oracle_instance
Environ SYBASE=sybase_path
Environ DSQUERY=sybase_instance
Environ DB2INSTANCE=db2_instance
Environ RXQUEUE_OWNER_PID=REXX_perf_var
Environ LANG=locale
}
```

Parameter:

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

Für Apache:

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

Für Domino Go Webserver:

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

Role Responder
Nur für Domino Go Webserver erforderliches Schlüsselwort.

URL Nur für Domino Go Webserver erforderliches Schlüsselwort und erforderliche URL-Adresse. Die URL-Adresse zeigt auf den für die Anweisung EXEC_PATH angegebenen Pfad.

BindPath Nur für Domino Go Webserver unter AIX erforderliches Schlüsselwort und erforderliche Pfadanweisung. Der Pfad des von Net.Data und FastCGI verwendeten eindeutigen UNIX-Sockets.

proc_num Die Anzahl von Anforderungen, die gleichzeitig bearbeitet werden können. Der Standardwert ist 1, sollte jedoch je nach Ihren Anwendungsanforderungen für eine Leistungsoptimierung erhöht werden. Informationen zur Optimierung der Leistung finden Sie in „Verwenden von FastCGI“ auf Seite 195.

Für Apache:

```
-processes 7
```

Für ICS oder Domino Go Webserver:

```
NumProcesses 7
```

libpath Die Anweisungen LIBPATH (gemeinsam benutzte Bibliothek oder DLL-Datei), die in jeder Anweisung ENVIRONMENT in der Net.Data-Initialisierungsdatei deklariert sind. Für Zugriff auf DB2 muß die Anweisung LIBPATH den Pfad zum DB2-UDB-Bibliotheksverzeichnis enthalten, z. B.:

```
/usr/lpp/db2_05_00/lib
```

Für Apache:

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

Für Domino Go Webserver:

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

oracle_path

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

Für Apache:

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

Für Domino Go Webserver:

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

oracle_instance

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

Für Apache:

```
-initial-env ORACLE_SID=mvpdb2
```

Für Domino Go Webserver:

```
Environ ORACLE_SID=mvpdb2
```

sybase_path

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

Für Apache:

```
-initial-env SYBASE=/home.native/sybase/product
```

Für Domino Go Webserver:

```
Environ SYBASE=/home.native/sybase/product
```

sybase_instance

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

Für Apache:

```
-initial-env DSQUERY=SybaseAIX
```

Für Domino Go Webserver:

```
Environ DSQUERY=SybaseAIX
```

db2_instance

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

Für Apache:

```
-initial-env DB2INSTANCE=wwwinst
```

Für Domino Go Webserver:

```
Environ DB2INSTANCE=wwwinst
```

REXX_perf_var

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 Variable finden Sie in Anhang B, „Net.Data für AIX“ auf Seite 239.

Für Apache:

```
-initial-env RXQUEUE_OWNER_PID=0
```

Für Domino Go Webserver:

```
Environ RXQUEUE_OWNER_PID=0
```

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

Für Apache:

```
-initial-env LANG=De_DE
```

Für Domino Go Webserver:

```
Environ LANG=De_DE
```

2. **Für Apache:** Fügen Sie das Verzeichnis `fcgi-bin` als einen neuen Prozeduraliasnamen in der Datei „srm.conf“ hinzu: `ScripAlias /fcgi-bin/ /u/mydir/apache/fcgi-bin`
3. Stellen Sie alle Hyperlinks in statisch oder dynamisch generierten Web-Seiten 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 und Java-Beans

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

Weitere Informationen zur Installation des Web-Servers und zu den Anweisungen in der Konfigurationsdatei des Web-Servers finden Sie in Ihrer Web-Server-Dokumentation.

Konfigurieren von Net.Data zur Verwendung mit den Web-Server-APIs

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

- IBM Internet Connection Server API (ICAPI)
- Lotus Domino Go Webserver API (GWAPI)
- Microsoft Internet Server API (ISAPI)
- Netscape API (NSAPI)

Weitere Informationen zu den einzelnen APIs finden Sie in „Verwenden der Web-Server-APIs“ auf Seite 195 und in der Informationsdatei (README) für Ihre Version von Net.Data.

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

In den folgenden Abschnitten wird beschrieben, wie Sie Net.Data und den Web-Server 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 ICAPI und GWAPI zu konfigurieren:

Domino Go Webserver ist das Nachfolgeprodukt von IBM Internet Connection Secure Server. Bei einer Erweiterung entscheiden Sie sich möglicherweise für die Verwendung des neueren Produkts Domino Go Webserver. Beachten Sie, daß es sich bei GWAPI und ICAPI um das gleiche Produkt handelt, das lediglich umbenannt wurde, um anzugeben, welcher Web-Server verwendet wird.

1. Stoppen Sie den Web-Server.
2. Stellen Sie sicher, daß sich die DLL-Datei bzw. die gemeinsam benutzte Bibliothek von ICAPI bzw. GWAPI im Verzeichnis `CGI-BIN` bzw. `ICAPI-LIB` des Servers befindet.

   Spezifische Datei- und Verzeichnisnamen finden Sie in der Net.Data-Informationsdatei (README) oder im Programmverzeichnis für Ihr Betriebssystem.
3. Fügen Sie der Konfigurationsdatei des Web-Servers (`httpd.conf` oder `httpd.cnf`) eine Serviceanweisung hinzu, um die API aufzurufen.

   Beispiel:

   ```
   Service /cgi-bin/db2www*
   /usr/lpp/internet/server_root/cgi-bin/dtwicapi.o:dtw_icapi*
   ```

   Spezifische Datei- und Verzeichnisnamen finden Sie in der Net.Data-Informationsdatei (README) für Ihr Betriebssystem.
4. Starten Sie den Web-Server erneut.

ICAPI und GWAPI verfügen über die volle Kompatibilität zur Unterstützung vorhandener Anwendungen. Verwenden Sie die gleichen Methoden wie bei CGI zum Aufrufen einer URL-Adresse, eines Formulars oder einer Programmverbindung (Link) mit ICAPI bzw. GWAPI. Ein mit CGI erfolgreich ausführbares Makro wird unter Verwendung von ICAPI bzw. GWAPI ebenfalls erfolgreich ausgeführt. An diesen Makros brauchen keine Änderungen vorgenommen zu werden.

Gehen Sie wie folgt vor, um ISAPI zu konfigurieren:

1. Stoppen Sie den Web-Server.
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 Programmverbindungen (Links) auslassen. Verwenden Sie statt dessen `dtwisapi.`*`filetype`*. Beispiel: Die folgende URL-Adresse ruft Net.Data als CGI-Programm auf:

   `http://server1.stl.ibm.com/cgi-bin/db2www/test1.d2w/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.d2w/report`
4. Wenn Sie Ihr Makro `test1.d2w` im Unterverzeichnis `/order/` unter einem der in der Anweisung MACRO_PATH angegebenen Verzeichnisse oder im aktuellen Verzeichnis des Web-Servers 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.d2w/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.d2w/report`
5. Starten Sie den Web-Server erneut.

Gehen Sie wie folgt vor, um NSAPI zu konfigurieren:

1. Stoppen Sie den Web-Server.
2. Kopieren Sie die mit Net.Data gelieferte DLL-Datei für NSAPI in das Server-Verzeichnis. 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 Server-Konfigurationsdatei 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=magnum-internal/d2w exts=d2w`

4. Versetzen Sie die Net.Data-Makrodateien 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 Makrodateien gesucht werden soll.

6. Da NSAPI die CGI-Verarbeitung umgeht, können Sie den Teil `cgi-bin/db2www/` der URL-Adresse in Formularen und Programmverbindungen (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.d2w/report`

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

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

7. Starten Sie den Web-Server 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 Makrodateien enthält.

3. Ändern Sie die Programmverbindungen (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 im CGI-Modus ein Net.Data-Makro auf, das im Verzeichnis `/orders/` gespeichert ist:

   `http://server1.stl.ibm.com/cgi-bin/db2www/orders/test1.d2w/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.d2w/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 49
- „Konfigurieren von Pfadanweisungen“ auf Seite 49
- „Konfigurieren von Anschlüssen“ auf Seite 51
- „Konfigurieren von Cliettes“ auf Seite 52
- „Konfigurieren von Sprachumgebungen“ auf Seite 57
- „Definieren von Konfigurationsvariablen“ auf Seite 61

Informationen zum Einrichten von Administration Tool und zum Sicherstellen, daß die Softwarevoraussetzungen erfüllt werden, finden Sie in „Vorbereitung“.

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 Version 2 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 19 beschrieben. Abb. 7 zeigt die Seite **Pfad**.

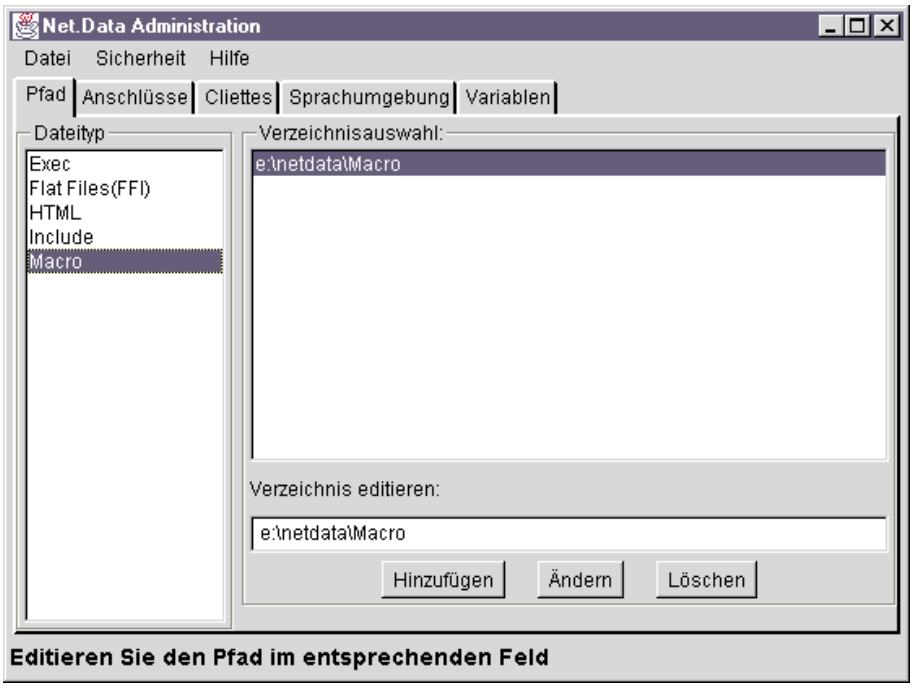


Abbildung 7. Die Seite "Pfad" des Notizbuchs "Net.Data Administration".. Auf dieser Seite können Sie Pfadanweisungen hinzufügen, ändern oder löschen.

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**.

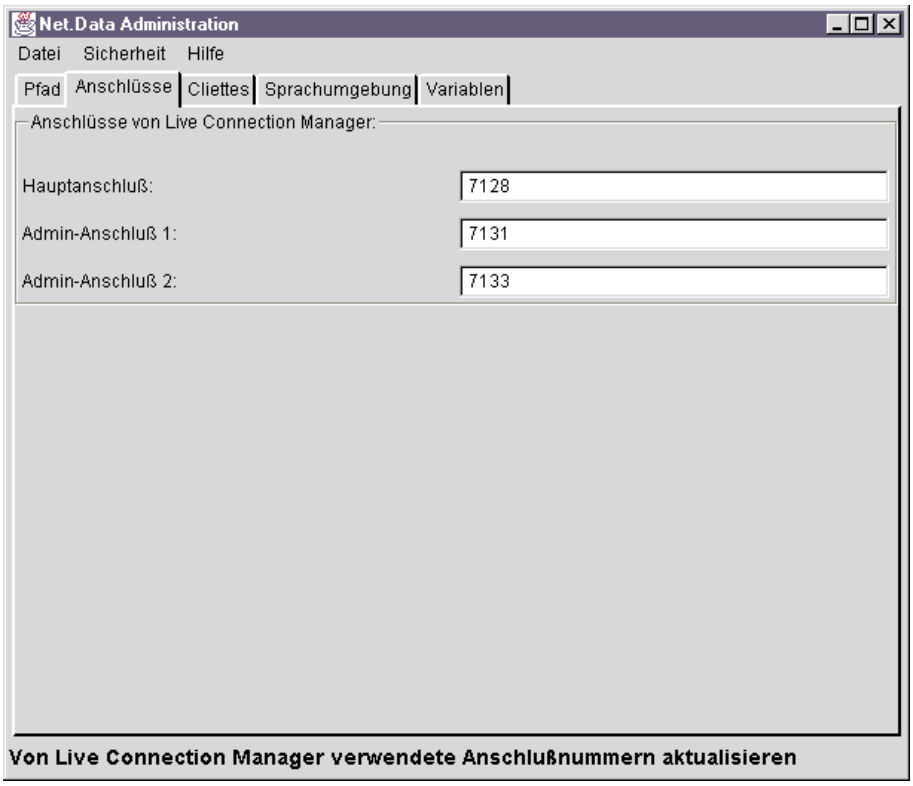


Abbildung 8. Die Seite "Anschlüsse" von Net.Data Administration Tool. Auf dieser Seite können Sie Anschlüsse angeben.

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 196. Abb. 9 zeigt die Seite **Cliettes**.

Abbildung 9. Die Seite "Cliettes" von Net.Data Administration Tool. Auf dieser Seite können Sie Cliettes hinzufügen, ändern und löschen.

Gehen Sie wie folgt vor, um eine Cliette hinzuzufügen:

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.

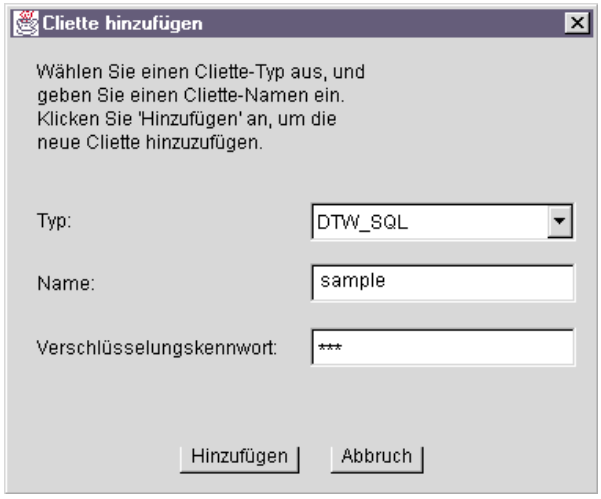


Abbildung 10. Das Fenster "Cliette hinzufügen" von Net.Data Administration Tool. Auf dieser Seite können Sie Cliettes hinzufügen.

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 33.
   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 33.

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 23 ausführlich behandelt. Abb. 11 zeigt die Seite **Sprachumgebung**.

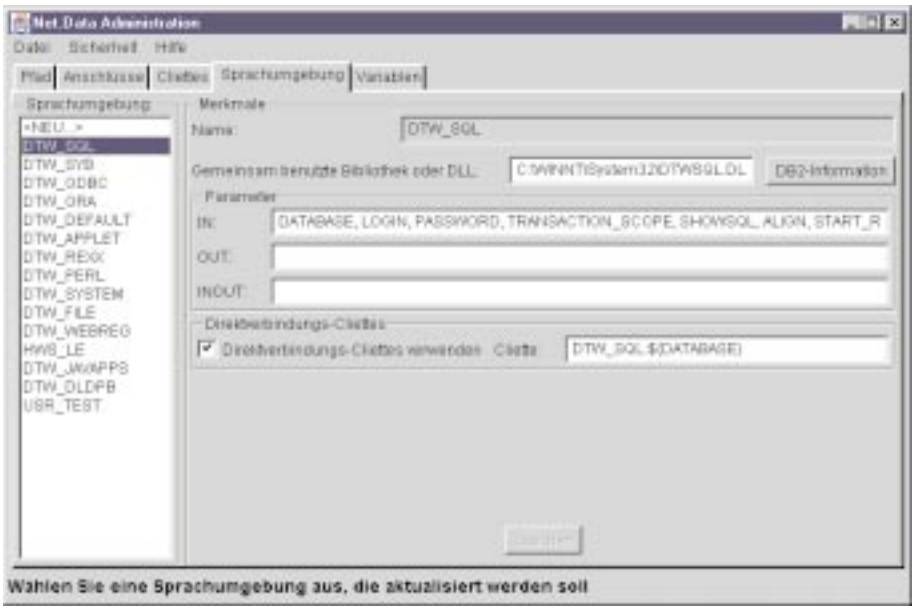


Abbildung 11. Die Seite "Sprachumgebung" von Net.Data Administration Tool. Auf dieser Seite können Sie Sprachumgebungen angeben.

Gehen Sie wie folgt vor, um eine Sprachumgebung hinzuzufügen:

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.

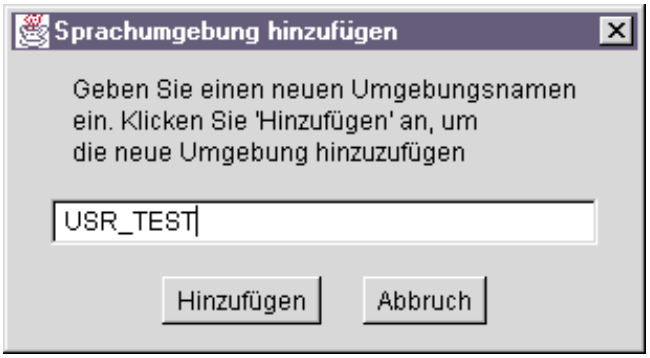


Abbildung 12. Das Fenster "Sprachumgebung hinzufügen" von Net.Data Administration Tool. Auf dieser Seite können Sie eine neue Sprachumgebung angeben.

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 auf Seite 57 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 23.
   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.

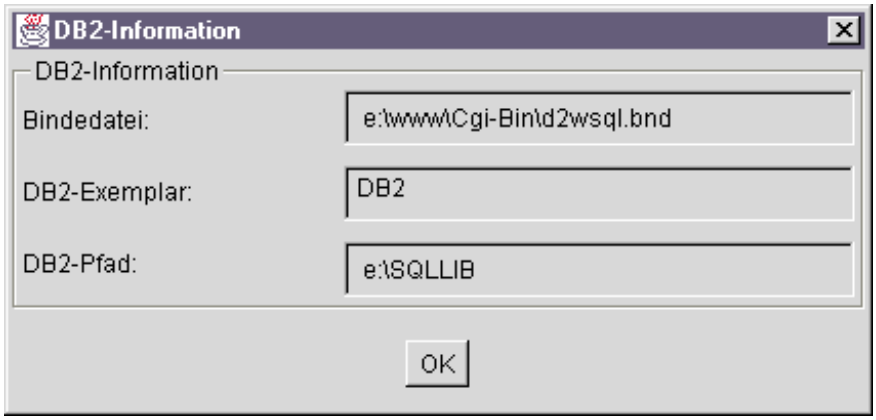


Abbildung 13. Das Fenster "DB2-Information" von Net.Data Administration Tool. Auf dieser Seite können Sie spezifische DB2-Datenbankinformationen angeben.

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_SYB, DTW_SQL, DTW_JAVAPPS

Für OS/2:
DTW_SQL, DTW_JAVAPPS

Für AIX: DTW_ODBC, DTW_ORA, DTW_SYB, 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 Standardsprachumgebungen.

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**.

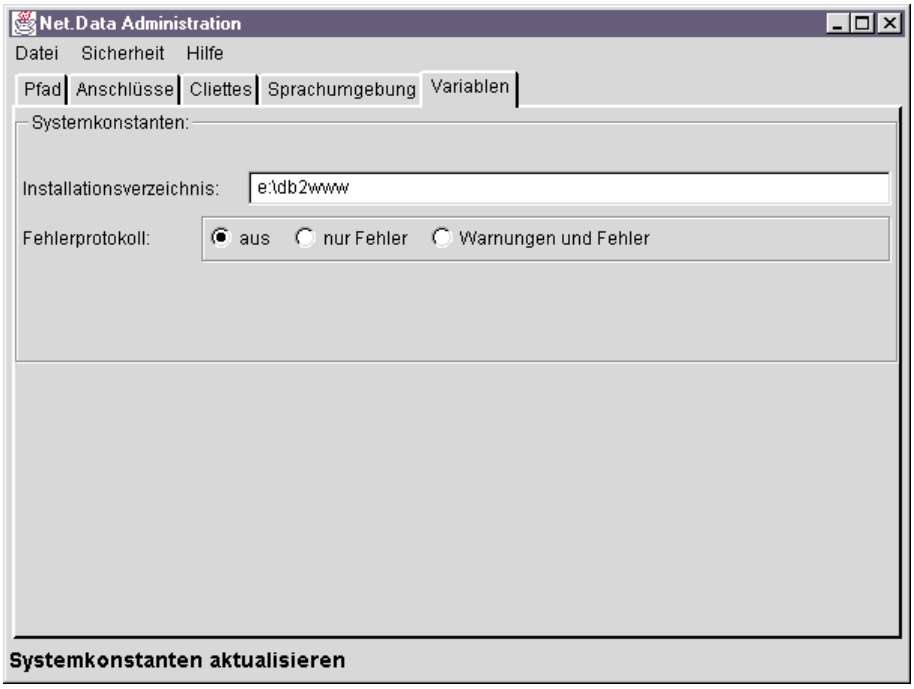


Abbildung 14. Die Seite "Variablen" von Net.Data Administration Tool. Auf dieser Seite können Sie Initialisierungsvariablen angeben.

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 Web-Server 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-Makrodateien 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 CACHEMGR.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.

Sichern der Datenbestände

Internet-Sicherheit wird durch eine Kombination aus Firewall-Technologie, Betriebssystemfunktionen, Web-Server-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 Web-Site 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 65
- „Verwenden der Authentifizierung“ auf Seite 66
- „Verwenden der Berechtigung“ auf Seite 66
- „Verwenden von Net.Data-Mechanismen“ auf Seite 67

Außerdem bietet Net.Data Kennwortverschlüsselung für Datenbank-Cliettes. Weitere Informationen hierzu finden Sie in „Konfigurieren von Cliettes“ auf Seite 52.

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

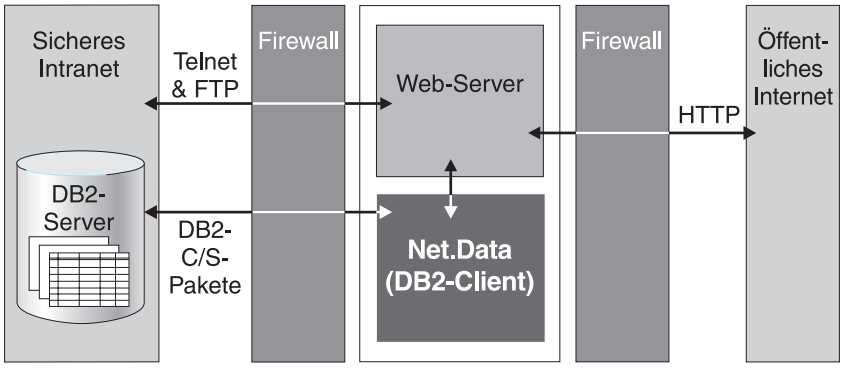


Abbildung 15. Konfiguration mit hoher Sicherheit

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

  – Installieren Sie Net.Data auf der Web-Server-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 Web-Server-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 Web-Server und dem sicheren Intranet. Eine Möglichkeit besteht darin, einen Socks-Server auf der Web-Server-Maschine zu installieren.

  – Geben Sie in der Konfigurationsdatei für die Paketfilterung der Firewall-Software an, daß eingehende TCP-Pakete vom HTTP-Standardanschluß auf den Web-Server zugreifen dürfen. Geben Sie weiterhin an, daß abgehende TCP-Bestätigungspakete vom Web-Server 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 Web-Server befinden sich außerhalb der Firewall auf einer Workstation-Plattform. Diese Konfiguration ist einfacher als die oben beschriebene, bietet jedoch trotzdem einen Datenbankschutz. Abb. 16 auf Seite 65 zeigt diese Konfiguration.

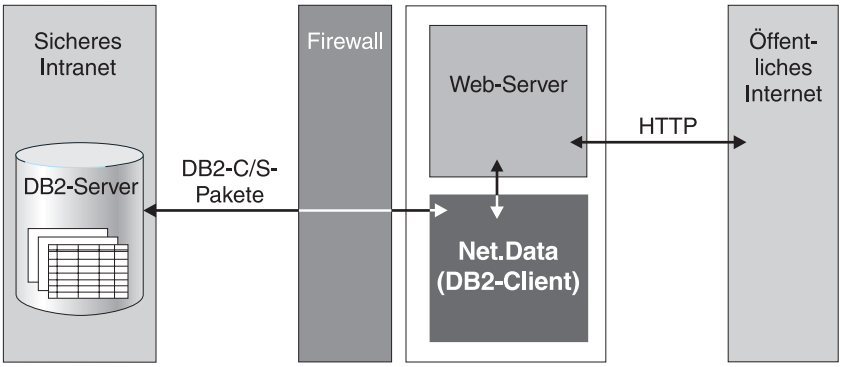


Abbildung 16. Konfiguration mit mittlerer Sicherheit

Hierzu müssen Sie CAE auf dem Web-Server 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 zeigt diese Konfiguration.

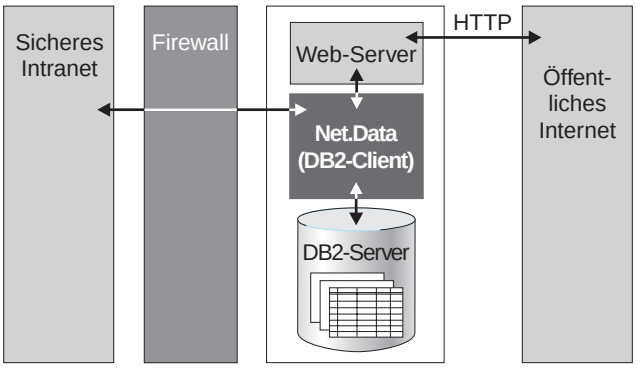


Abbildung 17. Konfiguration mit niedriger Sicherheit

Verschlüsseln Ihrer Daten im Netzwerk

Wenn Sie einen Web-Server verwenden, der SSL (Secured Sockets Layer) unterstützt, können Sie alle Daten, die zwischen einem Client-System und Ihrem Web-Server gesendet werden, verschlüsseln. Diese Sicherheitsmaßnahme unterstützt die Verschlüsselung von Anmelde-IDs, Kennwörtern und aller Daten, die über HTML-Formulare vom Client-System an den Web-Server übertragen werden, sowie aller Daten, die vom Web-Server an das Client-System gesendet werden. Die meisten Web-Server wie Internet Connection Secure Server, Version 2 Release 2 oder spätere Versionen und Lotus Domino Go Webserver, Version 4.6.1 oder spätere Versionen, 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 Web-Server 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 Web-Servern 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 Web-Server.
- 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 Programmverbindung (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 Web-Server-Konfigurationsdatei ein.
2. Stellen Sie sicher, daß die Benutzer-ID, unter der Net.Data ausgeführt wird, Zugriffsrechte für die Makrodateien besitzt. Weitere Informationen zur Erteilung von Zugriffsrechten finden Sie in „Erteilen von Zugriffsrechten für Dateien, auf die Net.Data zugreift“ auf Seite 62.

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 des Prozesses, der die Net.Data-Anforderung ausführt, eine ordnungsgemäße Authentifizierung ausgeführt wurde. Eine genauere Erklärung hierzu finden Sie in „Verwenden der Authentifizierung“.

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 eines oder mehrere Verzeichnisse an, die Net.Data durchsucht, wenn es versucht, Makrodateien, 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 „ Konfigurieren von Net.Data“ auf Seite 5.

Sie sollten zudem Berechtigungsprüfungen verwenden (siehe „Verwenden der Berechtigung“ auf Seite 66) und sicherstellen, daß Dateinamen in INCLUDE-Anweisungen nicht geändert werden können (siehe „Makro-Entwicklungsverfahren“ auf Seite 68).

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 16.
- Benutzer von Net.Data Version 2.0.5 und früher können die Funktion DTW_ASSIGN() verwenden (siehe „Makro-Entwicklungsverfahren“ auf Seite 68).

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 15.

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 `@DTW_ASSIGN(SHOWSQL, "NO")` anstelle von `%DEFINE SHOWSQL="NO"`, um die Net.Data-Variable SHOWSQL festzulegen. In diesem Fall überschreibt eine Abfragezeichenfolgezuordnung wie `SHOWSQL=YES` die Makroeinstellung nicht.

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

- Bei der Verwendung von Net.Data-Versionen, die die Konfigurationsvariable DTW_SHOWSQL unterstützen, können Sie mit dieser Variable 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 16.

- Ordnen Sie den Wert für SHOWSQL mit der im obigen Beispiel beschriebenen Funktion DTW_ASSIGN() zu, um eine Überschreibung zu vermeiden.

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

Mit DTW_ASSIGN können Sie auch sicherstellen, daß andere Net.Data-Variablen wie RPT_MAX_ROWS oder START_ROW_NUM nicht überschrieben werden. Weitere Informationen zu diesen Variablen finden Sie im entsprechenden Kapitel über Variablen des Handbuchs *Net.Data Reference*.

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 der Eingabewert, der der Variablen shlogid zugeordnet ist, aus einer einzelnen Käufer-ID besteht:

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

Beispiel 2: Verwenden von DTW_TRANSLATE()

Angenommen, Ihre Anwendung muß sicherstellen, daß der in der Eingabevariablen number_of_orders bereitgestellte Wert eine ganze Zahl ist. Hierzu können Sie die Umsetztabelle input_translation_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(number_of_orders, "x",
input_translation_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 112.

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 Web-Adresse finden:

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

Aufrufen von Net.Data

In diesem Kapitel wird beschrieben, wie Sie Net.Data mit den verschiedenen Web-Server-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 Web-Server-Schnittstellen konfigurieren:

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

Weitere Informationen zum Konfigurieren von Net.Data für diese Schnittstellen finden Sie in „ Konfigurieren von Net.Data“ auf Seite 5. Der Web-Server 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 Web-Server 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

Unabhängig von der von Ihnen gewählten Methode zum Aufrufen von Net.Data können Sie eine von zwei Anforderungsarten angeben. Die erste Anforderungsart gilt für die Ausführung eines Makros, die zweite für die Ausführung einer einzelnen SQL-Anweisung, gespeicherten Prozedur oder Funktion.

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

Direktanforderung
Gibt an, daß Net.Data eine SQL-Anweisung, gespeicherte Prozedur oder Funktion ausführt. In der Anforderung wird folgendes angegeben:

- Der Name einer Sprachumgebung
- Eine SQL-Anweisung oder der Name einer Funktion zusammen mit beliebigen Parameterwerten, die für den Aufruf der Funktion erforderlich sind
- Für den Aufruf der SQL-Anweisung oder Funktion erforderliche Formulardaten

Web-Entwickler, 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.

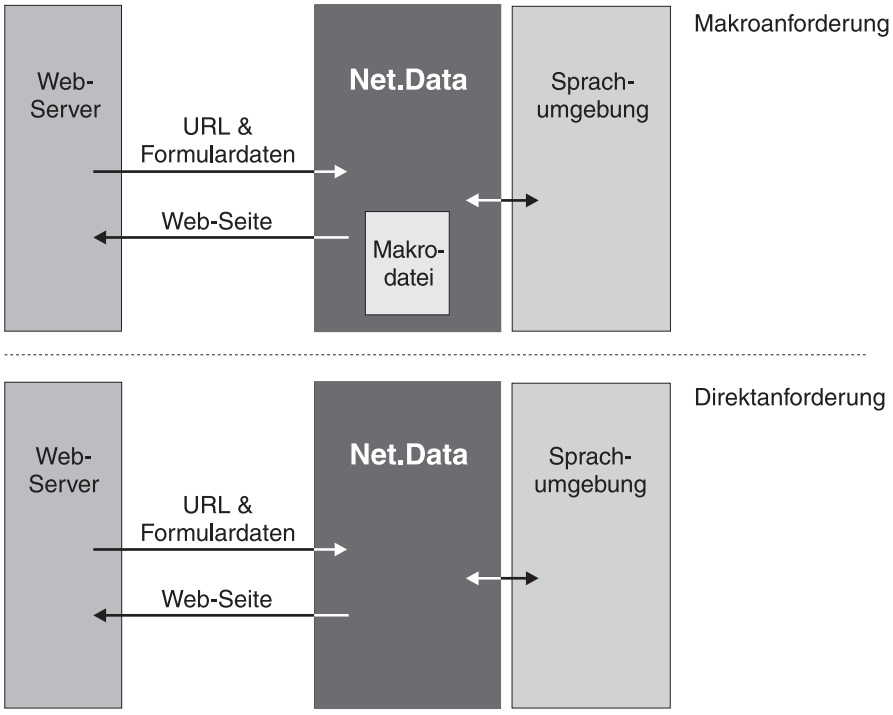


Abbildung 18. Makroanforderung im Vergleich zu Direktanforderung

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-Programmverbindung (Link) oder als HTML-Formular in die HTML-Seite codiert werden. Der Web-Server ruft Net.Data über CGI, FastCGI oder eine der Web-Server-APIs auf. Sie können Net.Data zudem mit Net.Data-Servlets aufrufen.

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.

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 78

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 Web-Server-APIs“ auf Seite 83
- „Aufrufen von Net.Data mit Java-Servlets und JavaBeans“ auf Seite 85

Aufrufen von Net.Data mit einem Makro (Makroanforderung)

In diesem Abschnitt wird gezeigt, wie Sie Net.Data durch Angabe eines Makros aufrufen können.

Die folgenden Syntaxanweisungen zeigen, wie Net.Data aufgerufen werden kann.

- URL-Adresse:

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

Parameter:

server Gibt den Namen und Pfad des Web-Servers an. Wenn es sich bei dem Server um den lokalen Server handelt, kann der Server-Name 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 19.

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

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

?*name***=***val***&...**
Gibt einen oder mehrere wahlfreie Parameter an, die an Net.Data weitergegeben werden.

Sie können die URL-Adresse direkt in Ihrem Browser angeben, oder Sie können sie wie folgt in einer HTML-Programmverbindung (Link) bzw. in einem HTML-Formular verwenden:

- HTML-Programmverbindung (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-Programmverbindung (Link)

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

Beispiel 2: Aufrufen von Net.Data mit einem Formular

```
<FORM METHOD=POST
ACTION="http://server/cgi-bin/db2www/myMacro.d2w/report">
.
.
.
</FORM>
```

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

- „HTML-Programmverbindungen (Links)“
- „HTML-Formulare“ auf Seite 77

HTML-Programmverbindungen (Links)

Wenn Sie eine Web-Seite verfassen, können Sie eine HTML-Programmverbindung (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-Programmverbindung (Link) definiert ist, führt Net.Data den HTML-Block im Makro aus.

Verwenden Sie den HTML-Befehl <a>, um eine HTML-Programmverbindung (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 Programmverbindung (Link) gezeigt, die zur Ausführung einer SQL-Abfrage führt, wenn ein Benutzer den Text "List all monitors" auf einer Web-Seite auswählt.

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

Durch das Anklicken der Programmverbindung (Link) wird das Makro listA.d2w mit dem HTML-Block "report" aufgerufen. Siehe folgendes Beispiel:

```
%DEFINE DATABASE="MNS97"


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

%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. Dieses Beispiel zeigt die Ergebnisse der Abfrage durch Generierung eines Standardberichts an. Informationen zum Anpassen Ihrer Berichte mit einem REPORT-Block finden Sie in „REPORT-Blöcke“ auf Seite 131.

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 Web-Seite beeinflussen.

Das folgende Beispiel baut auf dem Beispiel zur Monitorliste aus „HTML-Programmverbindungen (Links)“ auf Seite 76 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/equiplst.d2w/report">
<P>What type of hardware do you want to see?
<MENU>
<LI><INPUT TYPE="RADIO" NAME="hdware" VALUE="MON" checked> Monitors
<LI><INPUT TYPE="RADIO" NAME="hdware" VALUE="PNT"> Pointing devices
<LI><INPUT TYPE="RADIO" NAME="hdware" VALUE="PRT"> Printers
<LI><INPUT TYPE="RADIO" NAME="hdware" VALUE="SCN"> Scanners
</MENU>

<INPUT TYPE="SUBMIT" VALUE="Submit">
</FORM>
```

Nachdem der Benutzer im Browser seine Auswahl getroffen und den Knopf für die Übergabe angeklickt hat, verarbeitet der Web-Server den Parameter ACTION des Befehls FORM, wodurch Net.Data aufgerufen wird. Net.Data führt anschließend den HTML-REPORT-Block im Makro `equiplst.d2w` aus:

```
%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)
%}
%}
%}

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

Im obigen Beispiel wird der Wert für `TYPE=$(hdware)` der SQL-Anweisung der HTML-Formulareingabe entnommen.

Eine detaillierte Beschreibung der im ROW-Block verwendeten Variablen finden Sie im Handbuch *Net.Data Reference*.

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 15.

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, Sybase, Java, SYSTEM, Perl und REXX unterstützen Direktanforderungen . Sie können Net.Data mit einer URL-Adresse, einem HTML-Formular oder einer HTML-Programmverbindung (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-Programmverbindung (Link) enthält zum Beispiel eine Direktanforderung:

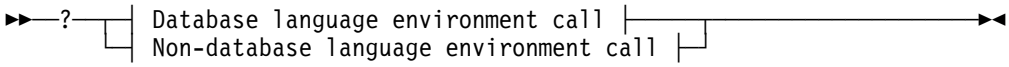
```
<A HREF="http://server/cgi-bin/db2www/?LANGENV=DTW_PERL&FUNC=my_perl(hi)">
  any text</A>
```

Syntax für Direktanforderungen

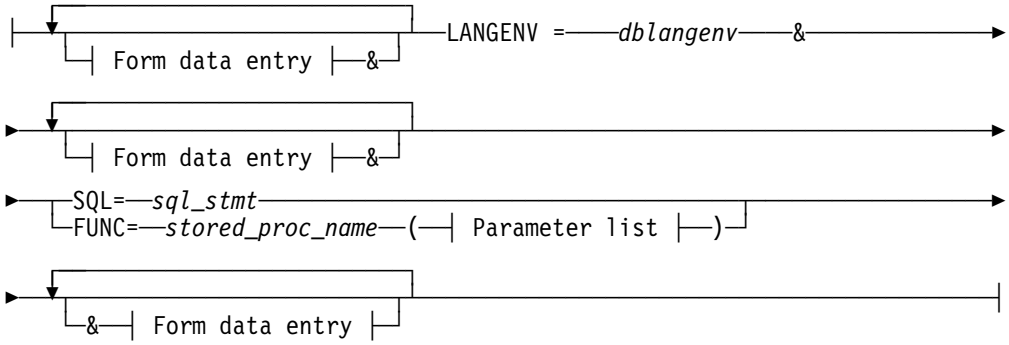
Die Syntax für das Aufrufen von Net.Data mit einer Direktanforderung kann den Aufruf einer Datenbank- bzw. Nicht-Datenbanksprachumgebung enthalten.

Syntax

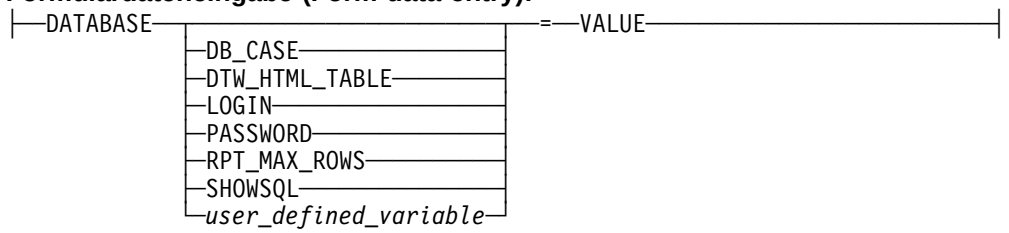
Syntax (Database language environment call)



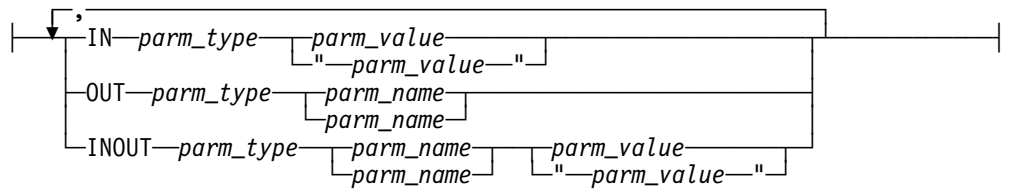
Aufruf einer Datenbanksprachumgebung (Database language environment:



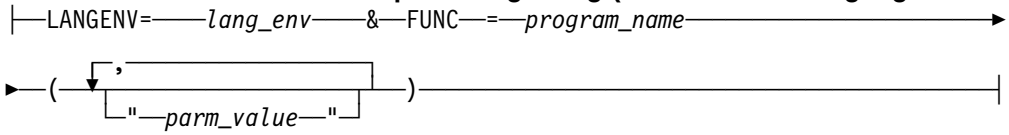
Formulardateneingabe (Form data entry):



Parameterliste (Parameter list):



Aufruf einer Nicht-Datenbanksprachumgebung (Non-database language:



Parameter

Aufruf einer Datenbanksprachumgebung (Database language environment call)

Gibt eine Direktanforderung an Net.Data an, die eine Datenbanksprachumgebung aufruft.

Formulardateneingabe (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.

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 in einer Tabelle an, die eine Funktion in einem Bericht zurückgibt.

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

START_ROW_NUM
Gibt die Zeilennummer in einer Tabelle für eine Funktion an, die als Start des Berichts verwendet werden 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
- DTW_SYB

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.

Aufruf einer Nicht-Datenbanksprachumgebung (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-Programmverbindungen (Links): In den folgenden Beispielen werden Direktanforderungen zum Aufrufen von Net.Data über Programmverbindungen (Links) verwendet.

Beispiel 1: Eine Programmverbindung (Link), die die Perl-Sprachumgebung sowie eine Perl-Prozedur aufruft, die 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 Programmverbindung (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>
```

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. Sie können mit der integrierten Funktion DTW_URLESCSEQ Text codieren, der innerhalb einer URL-Adresse codiert werden muß. Weitere Informationen zur Funktion DTW_URLESCSEQ finden Sie in der entsprechenden Beschreibung im Handbuch *Net.Data Reference*.

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

Beispiel 1: 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>
```

Dieses Beispiel enthält eine Variablensubstitution in der SQL-Anweisung, um die WHERE-Klausel dynamisch zu gestalten.

URL-Adresse: In den folgenden Beispielen werden Direktanforderungen zum Aufrufen von Net.Data über URL-Adressen verwendet.

Beispiel 1: Eine URL-Adresse, die zur Ausführung einer SQL-Abfrage mit Hilfe der SQL-Sprachumgebung führt

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

Beispiel 2: Eine URL-Adresse, die die Perl-Sprachumgebung und eine ausführbare Datei aufruft, die sich nicht in der Pfadanweisung EXEC der Net.Data-Initialisierungsdatei befindet

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

Beispiel 3: Eine URL-Adresse, die die Systemsprachumgebung und eine externe Perl-Prozedur aufruft

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

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

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

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

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

Aufrufen von Net.Data über die Web-Server-APIs

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

GWAPI-Plug-In und ICAPI-Plug-In
: API-Plug-In für Lotus Domino Go Webserver, dem Nachfolgeprodukt vom Plug-In für IBM Internet Connection Secure Server

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 Web-Server-APIs für Ihr Betriebssystem unterstützt werden. Informationen zum Konfigurieren von Net.Data und des Web-Servers zur Verwendung mit APIs finden Sie in „Konfigurieren von Net.Data zur Verwendung mit den Web-Server-APIs“ auf Seite 44.

Anforderungen:

- Wenn Net.Data im Modus GWAPI, ICAPI, ISAPI oder NSAPI ausgeführt wird, müssen Sie Ihren Web-Server erneut starten, damit er Net.Data erneut laden und als Prozeß ausführen kann.

- Wenn Sie Änderungen an der Initialisierungsdatei vorgenommen haben, nachdem der Web-Server Net.Data im API-Modus aufgerufen hat, müssen Sie den Web-Server 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 Sybase-Sprachumgebungen eine Direktverbindung erforderlich.

Gehen Sie wie folgt vor, um Web-Server-APIs aufzurufen:

Für ICAPI und GWAPI:

Syntax:

`http://`*`server_name`*`/CGI-BIN/db2www/`*`macro_name`*`/`*`html_block`*

Parameter:

server_name
Name des Servers

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

html_block
Name des zu verarbeitenden HTML-Blocks im Makro

Beispiel:

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

Für ISAPI:

Syntax:

`http://`*`server_name`*`/`*`server_HTML_root_directory`*`/`*`dll_name`*`/`
`macro_name``/`
`html_block`

Parameter:

server_name
Name des Servers

server_HTML_root_directory
Name des HTML-Stammverzeichnisses des Web-Servers

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

html_block
Name des zu verarbeitenden HTML-Blocks im Makro

Beispiel:

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

Für NSAPI:

Syntax:

```
http://server_name/macro_name/html_block
```

Parameter:

server_name
Name des Servers

macro_name
Der relative Pfad Ihres Makros unter dem Verzeichnis, das mit MACRO_PATH angegeben wird. Die Erweiterung der Makrodatei, zum Beispiel `.d2w`, muß in der Web-Server-Konfigurationsdatei definiert werden. Weitere Informationen hierzu finden Sie in „Konfigurieren von Net.Data zur Verwendung mit den Web-Server-APIs“ auf Seite 44.

html_block
Name des zu verarbeitenden HTML-Blocks im Makro

Beispiel:

```
http://myserver/mymacro.d2w/report
```

Aufrufen von Net.Data mit Java-Servlets und JavaBeans

Servlets sind Java-Klassen, die eine ähnliche Funktion wie CGI-Programme oder Plug-Ins für Web-Server-APIs haben. Ein Web-Server kann mit Servlets HTML-Seiten dynamisch erstellen. Servlets haben zwar keine eigene grafische Benutzerschnittstelle, ihre Klassen können jedoch lokal oder über das Netzwerk dynamisch geladen werden. Sie können über eine URL-Adresse (fern) oder einen Klassennamen (lokal) aufgerufen werden.

Net.Data stellt Servlets bereit, mit denen Sie über Net.Data auf einem beliebigen Java-fähigen Betriebssystem, das von Net.Data unterstützt wird, Net.Data-Makros aufrufen und Net.Data-Funktionen bzw. SQL-Anweisungen ausführen können.

In diesem Kapitel werden die Konzepte und Funktionen folgender Einrichtungen beschrieben:

„Net.Data-Servlets“

„Net.Data-JavaBeans“ auf Seite 92

Net.Data-Servlets

Net.Data stellt Servlets und Plug-Ins für NetObjects Fusion mit Net.Data-Funktionen bereit, die in einer Java-Umgebung verwendet werden können. Mit Servlets und Plug-Ins können Sie folgende Aktionen ausführen:

- Ausführen von Net.Data-Makros von einer URL-Adresse aus und als SSI (Server-Side-Include)
- Ausführen von Net.Data-Funktionen von einer URL-Adresse aus und als SSI

- Verwenden von NetObjects Fusion (NOF) zum Verwalten Ihrer Makros. NetObjects Fusion bietet bessere Integration in Ihre Web-Site-Verwaltung und eine benutzerfreundliche grafische Benutzerschnittstelle zum Entwickeln von einfachen Makros. Informationen zur Verwendung von Plug-Ins für NetObjects Fusion finden Sie in Anhang E, „Verwenden von Plug-Ins für NetObjects Fusion (NOF) mit Net.Data-Servlets“ auf Seite 247.

In diesem Abschnitt werden die folgenden Servlet-spezifischen Themen behandelt:

- „Informationen zu Net.Data-Servlets“
- „Ausführen von Net.Data-Servlets“ auf Seite 87

Informationen zu Net.Data-Servlets

Net.Data wird mit Servlets geliefert, die Sie beim Entwickeln und Verwalten von Makros mit Hilfe der Java-Umgebung unterstützen. Servlets sind Java-Klassen, die eine ähnliche Funktion wie CGI-Programme oder Plug-Ins für Web-Server-APIs haben. Servlets werden von einem Java-Servlet-fähigen Web-Server zum Erstellen von HTML-Seiten verwendet. Servlets haben zwar keine eigene grafische Benutzerschnittstelle, ihre Klassen können jedoch lokal oder über das Netzwerk dynamisch geladen werden und können über eine URL-Adresse (fern) oder einen Klassennamen (lokal) aufgerufen werden. Servlets sind für die Betriebssysteme Windows NT und AIX verfügbar.

Die Net.Data-Servlets sind auf Java basierende Oberflächen, die ein Makro oder eine Direktanforderung von Net.Data Version 2 mit Hilfe einer nativen DLL-Datei ausführen. Mit diesen Servlets können Sie ein vorhandenes Makro (MacroServlet - Makro-Servlet) oder eine einzelne Funktion (FunctionServlet - Funktions-Servlet) ausführen. Für diese Servlets ist Net.Data Version 2 erforderlich. Net.Data-Servlets können die Datenbank- bzw. Nicht-Datenbanksprachumgebungen ausführen und mit Direktverbindungen ausgeführt werden.

Die Servlets werden mit einer API zur Verwendung mit Ihren Anwendungen geliefert. Die Servlet-APIs sind in Net.Data dokumentiert. Die API-Dokumentation finden Sie in der Datei <*inst_dir*>/servlets/NetDataServlets.jar.

Net.Data wird mit zwei Servlets geliefert:

Makro-Servlet (com.ibm.netdata.servlets.MacroServlet)

Führt ein vorhandenes Net.Data-Makro aus.

Die Verwendung des Makro-Servlets ähnelt der Ausführung einer Net.Data-Makrodatei über die CGI-BIN-Schnittstelle, außer daß Sie das Makro über ein Java-Servlet ausführen. Für den Einsatz des Makro-Servlets ist die Installation von Net.Data Version 2 oder höher erforderlich.

Nachfolgend einige Vorteile des Makro-Servlets:

- SQL-Abfragen werden zur Leistungsverbesserung über ODBC mit Hilfe von Live Connection Manager ausgeführt.
- Makros können über SSIs (Server-Side-Includes) ausgeführt werden, um mehrere Makros in eine HTML-Datei einzubetten.

Das Makro-Servlet ermöglicht zudem nativen Zugriff auf heterogene Datenbanken wie DB2, Oracle und Sybase sowie verschiedene Sprachumgebungen wie Perl, REXX und Java.

Funktions-Servlet (com.ibm.netdata.servlets.FunctionServlet)

Führt die Net.Data-Funktion oder die SQL-Anweisung über eine Servlet-Schnittstelle, wie %FUNCTION DTW_SQL() aus. Weitere Informationen hierzu finden Sie in „Aufrufen von Net.Data ohne Makro (Direktanforderung)“ auf Seite 78.

Für den Einsatz des Funktions-Servlets ist die Installation von Net.Data Version 2 erforderlich.

Ausführen von Net.Data-Servlets

Die Net.Data-Servlets können von einer URL-Adresse aus oder als SSI innerhalb einer HTML-Datei ausgeführt werden. Mit den Plug-Ins für NetObjects Fusion können Sie die Net.Data-Servlets in Ihre NetObjects Fusion-Site aufnehmen. In den folgenden Abschnitten wird erläutert, wie Sie die Servlets durch Eingabe der entsprechenden Syntax für das Servlet ändern und ausführen können. Informationen zum Ändern und Ausführen der Servlets mit NetObjects Fusion finden Sie in Anhang E, „Verwenden von Plug-Ins für NetObjects Fusion (NOF) mit Net.Data-Servlets“ auf Seite 247.

Ausführen des Makro-Servlets: Geben Sie unter Verwendung einer der folgenden Syntaxoptionen in einer HTML-Datei die Servlet-Parameter ein:

1. URL-Adresse:

   ```
   http://myserver/servlet/com.ibm.netdata.servlets.MacroServlet
     ?MACRO=macro_value&BLOCK=block_value&parmnn=valuenn
   ```

   Beispiel:

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

2. SSI:

   ```
   <servlet code="com.ibm.netdata.servlets.MacroServlet">
   <param name="MACRO" value="macro_value">
   <param name="BLOCK" value="block_value">
   <param name="parmnn" value="valuenn">
   </servlet>
   ```

   Beispiel:

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

Parameter:

macro_value
Der vollständig qualifizierte Pfad zu einem vorhandenen Net.Data-Makro

block_value
Der Name des auszuführenden HTML-Blocks im angegebenen Net.Data-Makro, der Standardwert ist `report` (wahlfrei).

parmnn Zusätzliche, für das Makro erforderliche Parameter wie

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

valuenn Zusätzliche, für das Makro erforderliche Werte wie

```
... value="custno"
```

Parameter HTMLPATH: Wenn Sie eine Fehlernachricht erhalten, die auf einen fehlenden Parameter HTMLPATH hinweist, fügen Sie Ihrem Aufrufbefehl für das Servlet den Parameter HTMLPATH hinzu:

- URL-Adresse:

  ```
  http://myserver/servlet/com.ibm.netdata.servlets.MacroServlet
  ?MACRO=macro_name&BLOCK=block_value&htmlpath=html_path&parmnn=valuenn
  ```

  Beispiel:

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

- SSI:

  ```
  <servlet code="com.ibm.netdata.servlets.MacroServlet">
  <param name="MACRO" value="macro_value">
  <param name="BLOCK" value="block_value">
  <param name="htmlpath" value="html_path">
  <param name="parmnn" value="valuenn">
  </servlet>
  ```

  Beispiel:

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

Parameter OUTBUFLEN: Wenn die Ergebnisse Ihres Makros größer sind als 32 KB, geben Sie den Parameter OUTBUFLEN an. Wenn diese Parameter erforderlich sind und nicht angegeben werden, kann dies unter Umständen zu unvorhersehbaren Ergebnissen führen.

- URL-Adresse:

  ```
  http://myserver/servlet/com.ibm.netdata.servlets.MacroServlet
  ?MACRO=macro_name&BLOCK=block_value
  &OUTBUFLEN=output_buffer_size&parmnn=valuenn
  ```

Beispiel:

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

- SSI:

```
<servlet code="com.ibm.netdata.servlets.MacroServlet">
<param name="MACRO" value="macro_value">
<param name="BLOCK" value="block_value">
<param name="OUTBUFLEN" value="output_buffer_size">
<param name="parmnn" value="valuenn">
</servlet>
```

Beispiel:

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

Ausführen des Funktions-Servlets: Das Funktions-Servlet kann Net.Data mit Hilfe einer Direktanforderung aufrufen, um eine Funktion (wie eine REXX-Funktion) oder eine SQL-Anweisung auszuführen. Die für das Servlet angegebenen Parameter hängen davon ab, ob Sie eine Funktion oder eine SQL-Anweisung ausführen. Geben Sie unter Verwendung einer der folgenden Syntaxoptionen in einer HTML-Datei die Servlet-Parameter ein:

1. Für eine URL-Adresse:
   - ***Aufrufen einer Funktion:***

     ```
     http://myserver/servlet/com.ibm.netdata.servlets.FunctionServlet
     ?LANGENV=lang_env_name&FUNC=program_name&parmnn=valuenn
     ```

     Beispiel:

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

   - ***Aufrufen einer SQL-Anweisung:***

     ```
     http://myserver/servlet/com.ibm.netdata.servlets.FunctionServlet
     ?LANGENV=database_lang_env_name&SQL=SQL_statement
     &DATABASE=database_name&parmnn=valuenn
     ```

Beispiel:

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

2. SSI:

   - ***Aufrufen einer Funktion:***

     ```
     <servlet code="com.ibm.netdata.servlets.FunctionServlet">
     <param name="LANGENV" value="lang_env_name">
     <param name="FUNC" value="program_name">
     <param name="parmnn" value="valuenn">
     </servlet>
     ```

     Beispiel:

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

   - ***Aufrufen einer SQL-Anweisung:***

     ```
     <servlet code="com.ibm.netdata.servlets.FunctionServlet">
     <param name="LANGENV" value="lang_env_name">
     <param name="SQL" value="SQL_stmt_name">
     <param name="DATABASE" value="database_name">
     <param name="parmnn" value="valuenn">
     </servlet>
     ```

     Beispiel:

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

Parameter:

lang_env_name
: Die Net.Data-Sprachumgebung (wie DTW_SQL, DTW_REXX), die aufgerufen wird, um die Funktion, SQL-Anweisung oder gespeicherte Prozedur zu verarbeiten. Für diesen Parameter muß FUNC oder SQL verwendet werden.

program_name
: Der Name des Programms, das die auszuführende Funktion enthält, zum Beispiel `my_rexx`. Dabei steht `my_rexx` für den Namen einer ausführbaren REXX-Datei. Geben Sie mit den Parametern *parmnn* und *valuenn* Eingabeparameter für die Funktion ein.

database_name
: Der Name der Datenbank, der dem Parameter DATABASE zugeordnet ist

SQL_stmt_name
: Der Name einer SQL-Anweisung bzw. gespeicherten Prozedur, die auf eine Datenbank zugreift, zum Beispiel: `"select * from employee"`. Geben Sie mit den Parametern *parmnn* und *valuenn* Eingabeparameter für die SQL-Anweisung bzw. gespeicherte Prozedur ein.

parmnn
: Zusätzliche, für das Makro erforderliche Parameter wie `<param name="field1" ...`

valuenn
: Zusätzliche, für das Makro erforderliche Werte wie `... value="custnum"`

Parameter HTMLPATH: Wenn Sie eine Fehlernachricht erhalten, die auf einen fehlenden Parameter HTMLPATH hinweist, fügen Sie Ihrem Aufrufbefehl für das Servlet den Parameter HTMLPATH hinzu:

- URL-Adresse:

  ```
  http://myserver/servlet/com.ibm.netdata.servlets.FunctionServlet
  ?LANGENV=lang_env_name&FUNC=program_name&htmlpath=html_path
  &parmnn=valuenn
  ```

  Beispiel:

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

- SSI:

  ```
  <servlet code="com.ibm.netdata.servlets.FunctionServlet">
  <param name="LANGENV" value="lang_env_name">
  <param name="SQL" value="SQL_stmt_name">
  <param name="htmlpath" value="html_path">
  <param name="parmnn" value="valuenn">
  </servlet>
  ```

Beispiel:

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

Dabei gibt *html_path* den Pfad zum HTML-Stammverzeichnis des Web-Servers an, zum Beispiel `htmlpath=e:\html`.

Parameter OUTBUFLEN: Wenn die Ergebnisse Ihres Makros größer sind als 32 KB, müssen Sie den Parameter OUTBUFLEN angeben. Wenn diese Parameter erforderlich sind und nicht angegeben werden, kann dies unter Umständen zu unvorhersehbaren Ergebnissen führen.

- URL-Adresse:

  http://myserver/servlet/com.ibm.netdata.servlets.FunctionServlet
  ?LANGENV=*lang_env_name*&FUNC=*program_name*&**OUTBUFLEN=***output_*
  buffer_size&*parmnn=valuenn*

  Beispiel:

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

- SSI:

  ```
  <servlet code="com.ibm.netdata.servlets.FunctionServlet">
  <param name="LANGENV" value="lang_env_name">
  <param name="FUNC" value="program_name">
  <param name="OUTBUFLEN" value="output_buffer_size">
  <param name="parmnn" value="valuenn">
  </servlet>
  ```

  Beispiel:

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

Net.Data-JavaBeans

Net.Data stellt JavaBeans bereit, die in einer Java-Umgebung verwendet werden können, ohne daß ein Web-Server ausgeführt werden muß. Ein JavaBean ist eine objektorientierte Programmierschnittstelle, mit der Sie wiederverwendbare Anwendungen, d. h. Programmbausteine, erstellen können. Diese Objekte können in einem Netzwerk auf einem Java-fähigen Betriebssystem verwendet werden.

Das JavaBean ruft Net.Data unter Verwendung einer nativen DLL-Datei von Net.Data auf und füllt den Rückkehrcode und eine Zeichenfolge mit der Net.Data-Ausgabe (Ergebnisse) auf. Da JavaBeans eine native DLL-Datei verwenden, können Net.Data-Funktionen auch ohne die Ausführung eines Web-Servers eingesetzt werden.

Hinweis zur Ausgabe: Die von Net.Data-JavaBeans zurückgegebenen Ergebnisse werden in dem Format der Rückgabe Ihres Makros oder Ihrer Funktion ausgegeben. In der Regel ist dies HTML. Es wird empfohlen, die Ergebnisse an ein HTML-ähnliches JavaBean zu übergeben, das HTML versteht und die Ergebnisse anzeigen kann.

JavaBeans bieten Ihnen die folgenden Möglichkeiten:

- Ausführen von Net.Data-Makros
- Ausführen von SQL-Anweisungen über Net.Data

In diesem Abschnitt werden die folgenden JavaBeans-spezifischen Themen behandelt:

- „Informationen zu Net.Data-JavaBeans“
- „Definieren und Ausführen der Net.Data-JavaBeans“

Informationen zu Net.Data-JavaBeans

Net.Data wird mit JavaBeans geliefert, die Sie beim Entwickeln und Verwalten von Makros mit Hilfe der Java-Umgebung unterstützen. JavaBeans sind Java-Objekte mit folgender Schnittstelle:

- Bei Verwendung in einer JavaBean-Entwicklungsumgebung (wie Lotus BeanMachine) können Sie die gewünschten Komponenten mit Hilfe der mitgelieferten Anpassungsfunktion zur Verarbeitung verbinden, die Ergebnisse eines Makros bzw. einer SQL-Anweisung anzeigen und ein Java-Applet erstellen.
- Bei Verwendung der API können Sie mit den JavaBeans Net.Data-Funktionalität für Ihr eigenes Java-Applet bzw. Ihre eigene Java-Anwendung bereitstellen. Die API-Dokumentation befindet sich in der Datei <*inst_dir*>/beans/NetDataBeans.jar.

Net.Data stellt zwei Arten von JavaBeans bereit:

Net.Data-Makro-JavaBean
: Stellt eine auf Java basierende Schnittstelle zum Ausführen eines vorhandenen Net.Data-Makros über Net.Data bereit.

Net.Data-SQL-JavaBean
: Stellt eine auf Java basierende Schnittstelle zum Ausführen einer SQL-Anweisung über Net.Data bereit.

Die Net.Data-JavaBeans sind auf Java basierende Oberflächen, die mit Hilfe einer nativen DLL-Datei über Net.Data ausgeführt werden. Für beide ist die Installation von Net.Data Version 2 oder höher und JDK (Java Development Kit) Version 1.1 oder höher erforderlich.

Definieren und Ausführen der Net.Data-JavaBeans

In diesem Abschnitt wird beschrieben, wie Sie Net.Data-JavaBeans mit einem JavaBean-Entwicklungs-Tool wie Bean Machine definieren und ausführen können. Die Schritte zur Verwendung der Entwicklungs-Tools sind generisch, d. h., Sie können das Tool Ihrer Wahl einsetzen.

- „Das Makro-Bean“ auf Seite 94
- „Das SQL-Bean“ auf Seite 94

Das Makro-Bean
Mit dem Net.Data-Makro-Bean (com.ibm.netdata.beans.NetDataMacro) können Sie Java zum Ausführen eines vorhandenen Makros verwenden. Zur Verwendung dieses Beans müssen Sie Net.Data-Merkmale für das Bean angeben, damit es mit dem Makro arbeiten kann.***: Gehen Sie wie folgt vor, um das Net.Data-Makro-JavaBean mit einem JavaBean-Entwicklungs-Tool zu definieren:***

1. Fügen Sie die Datei `<inst_dir>/beans/NetDataBeans.jar` Ihrem JavaBean-Entwicklungs-Tool hinzu, bzw. importieren Sie sie.
2. Stellen Sie die folgenden Eingabemerkmale mit der Schnittstelle der Anpassungsfunktion für das Entwicklungs-Tool ein:

   Macro Gibt den Namen des auszuführenden, vorhandenen Makros an, zum Beispiel: `MyMacro.mac`

   Block Gibt den Namen des auszuführenden HTML-Blocks an. Der Standardwert ist `report`.

   HTML path
   Gibt den Pfad zur Net.Data-Datei `db2www.ini` an.

   Parameters
   Gibt den Parameternamen und bei der Ausführung des Makros zu verwendende Werte an.

   Syntax:

   ```
   name1=value1&nameN=valueN
   ```

Gehen Sie wie folgt vor, um das Net.Data-Makro-JavaBean mit einem JavaBean-Entwicklungs-Tool auszuführen:

1. Wählen Sie zur Ausführung des Makros die von Ihrem JavaBean-Entwicklungs-Tool bereitgestellte Ausführungsmaßnahme aus.
2. Nach der Ausführung des Makros können Sie auf die folgenden Ausgabemerkmale verweisen:

   RC Gibt den von Net.Data zurückgegebenen Rückkehrcode an.

   Results Gibt die von der Ausführung des Net.Data-Makros zurückgegebenen Daten an.

Das SQL-Bean: Mit Net.Data-SQL-Bean (com.ibm.netdata.beans.NetDataSQL) können Sie Java zum Ausführen einer SQL-Anweisung über Net.Data verwenden. Zur Verwendung dieses Beans müssen Sie Net.Data-Merkmale für das Bean angeben, damit es mit dem Makro arbeiten kann.

Gehen Sie wie folgt vor, um das Net.Data-SQL-JavaBean mit einem JavaBean-Entwicklungs-Tool zu definieren:

1. Fügen Sie die Datei NetDataBeans.jar Ihrem JavaBean-Entwicklungs-Tool hinzu, bzw. importieren Sie sie.
2. Stellen Sie die folgenden Eingabemerkmale mit der Schnittstelle der Anpassungsfunktion für das Entwicklungs-Tool ein:

   Language environment
   Gibt die zu verwendende Sprachumgebung an. Der Standardwert ist `DTW_SQL`.

   SQL Gibt die auszuführende SQL-Anweisung an. Der Standardwert ist `select * from employee`.

   DATABASE
   Gibt die zu verwendende Datenbank an. Der Standardwert ist `SAMPLE`.

   HTML path
   Gibt den Pfad zur Net.Data-Datei `db2www.ini` an.

   Parameters
   Gibt den Parameternamen und die bei der Ausführung der SQL-Anweisung zu verwendenden Werte an.

   Syntax:

   ```
   name1=value1&nameN=valueN
   ```

Gehen Sie wie folgt vor, um das Net.Data-SQL-JavaBean mit einem JavaBean-Entwicklungs-Tool auszuführen:

1. Wählen Sie zur Ausführung des Makros die von Ihrem JavaBean-Entwicklungs-Tool bereitgestellte Ausführungsmaßnahme aus.
2. Nach der Ausführung der SQL-Anweisung können Sie auf die folgenden Ausgabemerkmale verweisen:

   RC Gibt den von Net.Data zurückgegebenen Rückkehrcode an.

   Results Gibt die von der SQL-Anweisung zurückgegebenen Daten an.

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 Web-Seiten
- 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.

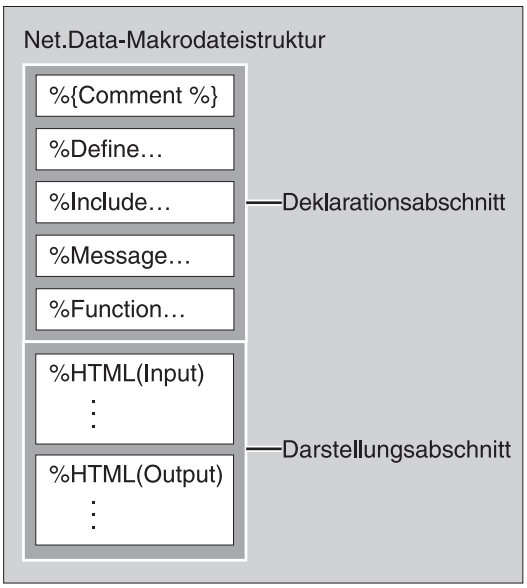


Abbildung 19. Makrostruktur

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

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 der Web-Server Zugriffsrechte für diese Datei hat. Weitere Informationen hierzu finden Sie in „Erteilen von Zugriffsrechten für Dateien, auf die Net.Data zugreift“ auf Seite 62.

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“
- „Net.Data-Makrovariablen“ auf Seite 104
- „Net.Data-Funktionen“ auf Seite 119
- „Generieren von Web-Seiten in einem Makro“ auf Seite 129
- „Bedingte Logik und Schleifen in einem Makro“ auf Seite 137

Aufbau eines Net.Data-Makros

Das Makro besteht aus zwei Abschnitten:

- Dem Deklarationsabschnitt, der die im Darstellungsabschnitt verwendeten Definitionen enthält. Im Deklarationsabschnitt werden zwei wahlfreie 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 der Web-Server Zugriffsrechte für Dateien hat, auf die in EXEC- und INCLUDE-Anweisungen verwiesen wird. Weitere Informationen hierzu finden Sie in „Erteilen von Zugriffsrechten für Dateien, auf die Net.Data zugreift“ auf Seite 62.
- Der Darstellungsabschnitt definiert das Layout der Web-Seite, verweist auf Variablen und ruft Funktionen mit Hilfe von HTML-Blöcken auf, die als Eingangs- und Endpunkte für das Makro verwendet werden. Wenn Sie Net.Data aufrufen, geben Sie den Namen eines HTML-Blocks als Eingangspunkt zur Verarbeitung des Makros an. Die HTML-Blöcke werden in „HTML-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 HTML-Seite 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">

</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 Web-Makros 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 Web-Server über Net.Data mit Hilfe von CGI, einer Web-Server-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
                               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 Web-Seite, 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 Übergabe
                                  des Formulars
                                  definiert. Auf diese Variable kann an
                                  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>]
 </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 alphanumerische Zeichen, Unterstreichungszeichen oder Punkte enthalten. 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 Web-Server befinden. So greift z. B. der ACTION-Parameter `ACTION="../othermacro.d2w/main"` auf den HTML-Block `main` im Makro `othermacro.d2w` 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.d2w/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.

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 Sprachumgebungen übergeben und von dort empfangen.

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 106
- „Verweisen auf Variablen“ auf Seite 108
- „Variablenarten“ auf Seite 109

Geltungsbereich von Kennungen

Eine Kennung, die eine Variable oder einen Funktionsaufruf darstellt, wird *sichtbar*, d. h. im Makro verwendbar, wenn sie deklariert oder von Net.Data verfügbar gemacht wurde. 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 (%}). (Beachten Sie hierbei, daß DEFINE-Blöcke von dieser Definition ausgeschlossen sind und als separate DEFINE-Anweisungen behandelt werden sollten.) 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. Alle Variablennamen mit Ausnahme der Tabellenvariablen N_*columnName* und V_*columnName* sind von der Groß-/Kleinschreibung abhängig.

  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 77 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.d2w/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 Programmverbindung (Link)

Sie definieren beispielsweise die Variable *homeURL* wie folgt:

```
%DEFINE homeURL="http://www.ibm.com/"
```

Dann können Sie Verweise auf die Home-Page in der Form $(*homeURL*) verwenden und eine Programmverbindung (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:

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 Variable 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 113 finden Sie ein Beispiel für die Verwendung von Bedingungsvariablen mit Listenvariablen.

Umgebungsvariablen

Sie können auf Umgebungsvariablen verweisen, die der Web-Server 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.software.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"
%DEFINE dateRpt="Today is $(date)"
```

Für jedes Vorkommen des Variablenverweises `$(dateRpt)` im Net.Data-Makro sucht Net.Data nach dem ausführbaren Programm `date`. Wenn das Programm gefunden wird, gibt Net.Data folgenden Wert zurück:

```
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 20.
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)"
```

Im nächsten Beispiel wird das Systemdatum als Teil des Berichts zurückgegeben:

```
%DEFINE database="celdial"
 %DEFINE tstamp=%EXEC "date"

 %FUNCTION(DTW_SQL) myQuery(){
SELECT CUSTNO, CUSTNAME from dist1.customer
   %REPORT{
   %ROW{
<A HREF="/cgi-bin/db2www/exmp.d2w/report?value1=$(V1)&value2=$(V2)">
$(V1) $(V2) </A> <BR>
%}
%}
%}

%HTML(report){
<H1>Report made: $(tstamp) </H1>
@myQuery()
%}
```

Jeder Bericht zeigt zur besseren Übersicht das Datum an. In diesem Beispiel werden außerdem die Kundennummer (CUSTNO) und der Kundenname (CUSTNAME) in eine Verbindung (Link) für ein anderes Net.Data-Makro eingefügt. Durch Anklicken eines Kunden im Bericht wird das Net.Data-Makro `exmp.d2w` aufgerufen, und die Nummer und der Name des Kunden werden an das Net.Data-Makro übergeben.

Verdeckte Variablen

Sie können verdeckte Variablen verwenden, um den tatsächlichen Namen einer Variablen für Anwendungsbenutzer unsichtbar zu machen, die die Quelle Ihrer Web-Seite 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.d2w/report">
<H2>Select one or more cities:</H2>
<INPUT TYPE="checkbox" NAME="conditions" VALUE="$$(cond1)">Sao Paolo<BR>
<INPUT TYPE="checkbox" NAME="conditions" VALUE="$$(cond2)">Seattle<BR>
<INPUT TYPE="checkbox" NAME="conditions" VALUE="$$(cond3)">Shanghai<BR>
<INPUT TYPE="submit" VALUE="Submit Query">
</FORM>
%}

%DEFINE{
DATABASE="custcity"
 %LIST " OR " conditions
cond1="cond1='Sao Paolo'"
cond2="cond2='Seattle'"
cond3="cond3='Shanghai'"
whereClause= ? "WHERE $(conditions)" : ""
%}

%FUNCTION(DTW_SQL) mySelect() {
SELECT name, city FROM citylist
$(whereClause)
%}

%HTML(REPORT){
@mySelect()
%}
```

Wenn im HTML-Formular keine Kästchen ausgewählt werden, ist `conditions` gleich NULL, so daß `whereClause` in der Abfrage ebenfalls NULL 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. Die Prozedur bzw. das Programm der Sprachumgebung definiert die Tabellenwerte jedoch zellweise. Weitere Informationen dazu, wie Tabellenvariablen von Sprachumgebungen verwendet werden, finden Sie in „Verwenden der Sprachumgebungen“ auf Seite 143.

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 116. Informationen zum Zugreifen auf einzelne Elemente einer Tabelle mit einer Tabellenfunktion finden Sie in „Tabellenfunktionen“ auf Seite 128. 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 Web-Seitenausgabe 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 Web-Seitenausgaben, die durch das Makro generiert werden, in einem Standardberichtsformat an. Das Standardberichtsformat wird in einem Tabellenformat mit Hilfe von <PRE> </PRE> angezeigt. 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.

Berichtsvariablen unterstützen Sie bei der Anpassung der Anzeige der Web-Seitenausgabe und bei der Verwendung der Web-Seitenausgabe mit Standardberichten und Net.Data-Tabellen. 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, legen HTML-Tabellenausgaben oder Standardtabellenausgaben fest und bestimmen weitere Anzeigemerkmale. Zum Beispiel können Sie mit der Variablen ALIGN die Verwendung führender und nachgestellter Leerzeichen für die Variablen zur Tabellenverarbeitung steuern. Im folgenden Beispiel wird die Variable ALIGN verwendet, um jeden Spaltennamen in einer Liste, die durch eine Abfrage zurückgegeben wird, durch Leerzeichen zu trennen.

```
%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 125
- „Aufrufen von integrierten Net.Data-Funktionen“ auf Seite 125

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-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 125.

Sie können mehrere FUNCTION-oder MACRO_FUNCTION-Blöcke mit dem gleichen Namen definieren, so daß sie gleichzeitig verarbeitet werden. Jeder der Blöcke muß eine 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 „Verwenden der Sprachumgebungen“ auf Seite 143.

parameter
Der Name einer Variablen mit lokalem Geltungsbereich, die durch den Wert eines entsprechenden in einem Funktionsaufruf angegebenen Arguments ersetzt wird. Parameterverweise, z. B. $(*parm1*), in den ausführbaren Anweisungen oder REPORT-Blöcken werden durch den tatsächlichen Wert des Parameters ersetzt. Zudem werden die Parameter an die Sprachumgebung übergeben. Ausführbare Anweisungen, die die spezifische Syntax dieser Sprache verwenden, können darauf 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 111.

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 einen oder mehrere REPORT-Blöcke zur Behandlung der Ausgabe des FUNCTION- oder MACRO_FUNCTION-Blocks. Siehe „REPORT-Blöcke“ auf Seite 131.

message-block

Definiert den MESSAGE-Block, der alle vom FUNCTION-Block zurückgemeldeten Nachrichten verarbeitet. Siehe „MESSAGE-Blöcke“ auf Seite 123.

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 der MESSAGE-Block in einem FUNCTION-Block definiert ist, ist sein Geltungsbereich für diesen FUNCTION-Block lokal. 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.

Net.Data verwendet die folgenden Regeln zur Verarbeitung des Werts der Variablen RETURN_CODE von einem Funktionsaufruf:

1. Ein lokaler MESSAGE-Block wird auf eine exakte Übereinstimmung hin überprüft, und die Verarbeitung wird je nach Angabe beendet (exit) oder fortgesetzt (continue).
2. Wenn RETURN_CODE ungleich 0 ist, wird ein lokaler MESSAGE-Block auf `+default` bzw. `-default` hin überprüft, und die Verarbeitung wird abhängig vom Vorzeichen von RETURN_CODE je nach Angabe beendet oder fortgesetzt.
3. Wenn RETURN_CODE ungleich 0 ist, wird ein lokaler MESSAGE-Block auf `default` hin überprüft, und die Verarbeitung je nach Angabe beendet oder fortgesetzt.
4. Ein globaler MESSAGE-Block wird auf eine exakte Übereinstimmung hin überprüft, und die Verarbeitung wird je nach Angabe beendet oder fortgesetzt.
5. Wenn RETURN_CODE ungleich 0 ist, wird ein globaler MESSAGE-Block auf `+default` bzw. `-default` hin überprüft, und die Verarbeitung wird abhängig vom Vorzeichen von RETURN_CODE je nach Angabe beendet oder fortgesetzt.
6. Wenn RETURN_CODE ungleich 0 ist, wird ein globaler MESSAGE-Block auf `default` hin überprüft, und die Verarbeitung je nach Angabe beendet oder fortgesetzt.
7. Wenn RETURN_CODE 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 Web-Seiten 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. 20 auf Seite 126 zeigt die Interaktion zwischen den integrierten Net.Data-Funktionen und dem Makro.

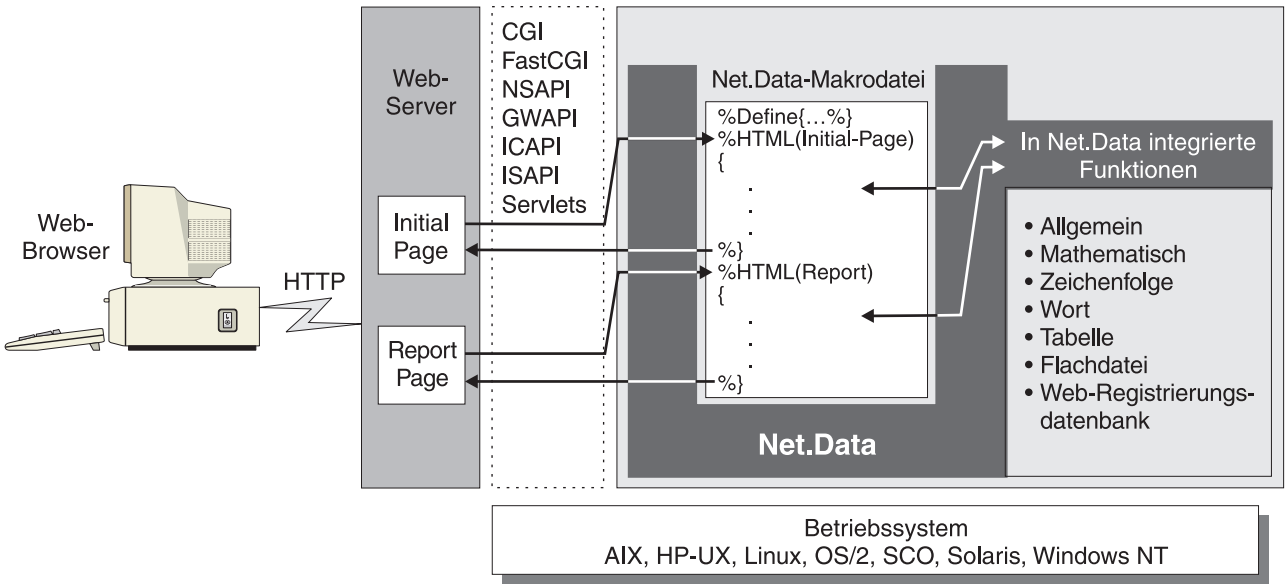


Abbildung 20. In Net.Data integrierte Funktionen

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 Web-Registrierdatenbanken.)
- **DTW_r, DTWF_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. Beschreibungen der einzelnen Funktionen mit Syntax und Beispielen finden Sie im Handbuch *Net.Data Reference*. Für einige dieser Funktionen müssen Variablen definiert werden, bevor sie verwendet werden können, oder sie müssen in einem spezifischen Kontext verwendet werden. Nicht alle Betriebssysteme unterstützen jede integrierte Funktion. Sie können anhand des Handbuchs *Net.Data Reference* ermitteln, welche Funktionen für Ihr Betriebssystem unterstützt werden.

- „Allgemeine Funktionen“ auf Seite 127
- „Mathematische Funktionen“ auf Seite 127
- „Zeichenfolgefunktionen“ auf Seite 128

Allgemeine Funktionen

Mit Hilfe dieser Gruppe von Funktionen können Sie Web-Seiten 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(cache_example) {

<html>
<head>
 <title>This is the page title</title>
</head>
<body>
 <center>
 <h3>This is the Main Heading</h3>
 <!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
 <! Joe Smith sees a very short page                    !>
 <!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!>
 %IF (customer == "Joe Smith")
</body>
</html>

@DTW_EXIT()

%ENDIF

...

</body>
</html>
 %}
```

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 den Wert einer Eingabevariablen einer Ausgabevariablen zuzuordnen. Sie können diese Funktion auch verwenden, um eine Variable in einem Makro zu verwenden. Im folgenden Beispiel wird der Variablen RC 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 einen oder mehrere HTML-Markierungsfeldeingabefehle 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.

Funktionen für Web-Registrierdatenbanken

Die Funktionen für Web-Registrierdatenbanken können zum Verwalten der Registrierdatenbanken und der darin enthaltenen Einträge verwendet werden. Eine Web-Registrierdatenbank 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 Registrierdatenbankeinträgen in einem OUT-Tabellenparameter zurück, und DTWR_UPDATEENTRY ersetzt vorhandene Werte für einen angegebenen Registrierdatenbankeintrag durch einen neuen Wert.

Generieren von Web-Seiten in einem Makro

Mit Net.Data können Sie leicht Standard-Web-Seiten im Browser des Anwendungsbenutzers darstellen. In den folgenden Abschnitten werden die HTML- und REPORT-Blöcke des Makros beschrieben und die Formatierung von Web-Seiten in Net.Data-Makros erläutert. Informationen zur Syntax dieser Blöcke finden Sie im Kapitel über die Sprachkonstrukte im Handbuch *Net.Data Reference*.

HTML-Blöcke

Ein Net.Data-Makro enthält HTML-Blöcke, die Textdarstellungsanweisungen wie HTML für einen Web-Browser generieren. In einem Makro muß mindestens ein HTML-Block definiert werden. Jeder HTML-Block generiert eine einzelne Web-Seite beim Browser. Net.Data verarbeitet bei jedem Aufruf nur jeweils einen HTML-Block. Wenn Sie eine Anwendung erstellen möchten, die mehrere Web-Seiten umfaßt, können Sie Net.Data mehrfach aufrufen, um HTML-Blöcke mit normalen Navigationsverfahren wie Programmverbindungen (Links) und Formularen zu verarbeiten.

Alle gültigen Textdarstellungsanweisungen wie HTML oder JavaScript können in einem HTML-Block vorkommen. Darüber hinaus können Sie in einem HTML-Block INCLUDE-Anweisungen, Funktionsaufrufe und Variablenverweise verwenden. Das folgende Beispiel zeigt eine gängige Verwendung von HTML-Blöcken in einem Net.Data-Makro:

```
%DEFINE DATABASE="MNS96"

   %HTML(INPUT){
<H1>Hardware Query Form</H1>
<HR>
<FORM METHOD="POST" ACTION="/cgi-bin/db2www/equiplst.d2w/report">
<dl>
<dt>What hardware do you want to list?
<dd><input type="radio" name="hdware" value="MON" checked>Monitors
<dd><input type="radio" name="hdware" value="PNT">Pointing devices
<dd><input type="radio" name="hdware" value="PRT">Printers
<dd><input type="radio" name="hdware" value="SCN">Scanners
</dl>
<HR>
<input type="submit" value="Submit">
</FORM>
%}

 %FUNCTION(DTW_SQL) myQuery(){
SELECT MODNO, COST, DESCRIP FROM EQPTABLE WHERE TYPE=$(hdware)
   %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 wie die im folgenden Beispiel aufrufen:

```
<a href="http://www.ibm.com/cgi-bin/db2www/equiplst.d2w/input">
  List of hardware</a>
```

Wenn der Anwendungsbenutzer diese Programmverbindung (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 der Block HTML(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, führt Net.Data den ACTION-Abschnitt des HTML-Elements FORM aus, der einen Aufruf an den Block HTML(OUTPUT) des Net.Data-Makros angibt. Net.Data verarbeitet daraufhin den Block HTML(OUTPUT) ebenso wie zuvor den Block HTML(INPUT).

Anschließend verarbeitet Net.Data den Funktionsaufruf `myQuery()`, der wiederum den SQL-Block FUNCTION 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 Block HTML(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. Wahlfrei 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 wahlfrei 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.d2w/details?name=$(V1)&location;=$(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 116 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 Standardberichtsformat sieht folgendermaßen aus:

```
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 154). 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 nur in der SQL-Sprachumgebung der Wert "YES" für DTW_DEFAULT_REPORT 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 ist, werden für beide Tabellen Standardberichte generiert.

```
%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 im REPORT-Block 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 beide der in der Parameterliste des FUNCTION-Blocks angegebenen Tabellen angegeben. Da keine Tabellennamen in den REPORT-Blöcken angegeben sind, werden Berichte für die zwei Tabellen in der Reihenfolge angezeigt, 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 in der Makrodatei 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 oder mehrere REPORT-Blöcke pro Ergebnismenge bzw. Tabellenname angeben. Der für den REPORT-Block angegebene Name muß mit einem entsprechenden Parameter für den Ergebnismengennamen oder Tabellennamen in der Anweisung CALL oder der Parameterliste des FUNCTION-Blocks übereinstimmen.
- 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 Web-Seite 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 Tabellen zurückgibt, wird die erste von der Funktion zurückgegebene Tabelle der Tabelle DTW_SAVE_TABLE_IN zugeordnet.
- Mehrere REPORT-Blöcke können in jeder Sprachumgebung verwendet werden.

Einschränkungen:

- Die Werte aller Berichtsvariablen in einer Funktion 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.
- Tabellenvariablen müssen mit der Anweisung TABLE definiert werden, bevor sie an eine Funktion übergeben werden können.
- Wenn der erste REPORT-Block einen Tabellennamen angibt, dann müssen alle REPORT-Blöcke Tabellennamen angeben.
- Wenn der erste REPORT-Block keinen Tabellennamen angibt, dann können keine REPORT-Blöcke 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 eine oder mehrere Bedingungen 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 141

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, eine oder mehrere Bedingungen 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 wahlfrei 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, eine oder mehrere Bedingungen 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)
  %}
%}
```

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. 21 zeigt die Beziehung zwischen dem Web-Server, Net.Data und den Net.Data-Sprachumgebungen.

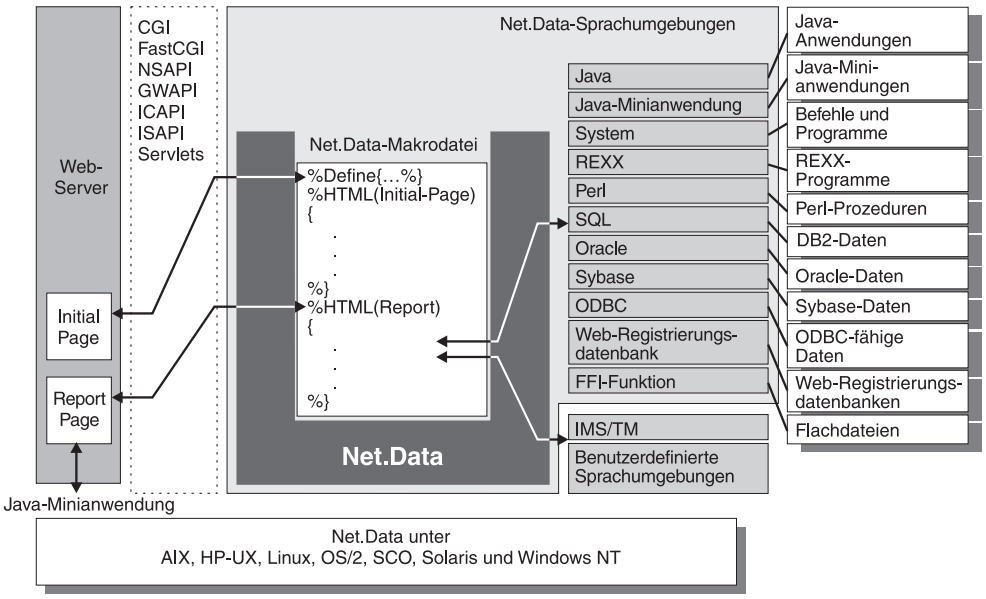


Abbildung 21. Die Net.Data-Sprachumgebungen

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 146
- „Datensprachumgebungen“ auf Seite 146
- „Programmiersprachumgebungen“ auf Seite 173

Konfigurationsinformationen zu den von Net.Data bereitgestellten Sprachumgebungen finden Sie in „Einrichten der Sprachumgebungen“ auf Seite 26.

Informationen zur Leistungssteigerung bei Verwendung der Sprachumgebungen finden Sie in „Optimieren der Sprachumgebungen“ auf Seite 226.

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:

- „Datensprachumgebungen“ auf Seite 146
- „Programmiersprachumgebungen“ auf Seite 173

Tabelle 8 auf Seite 145 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 8. Net.Data-Sprachumgebungen

Sprachumgebung	Umgebungsname	Beschreibung
FFI	DTW_FILE	FFI (Flat File Interface -Schnittstelle für Flachdateien) stellt Funktionen bereit, die Textdateien als Datenquellen unterstützen.
IMS-Web	HWS_LE	In der IMS-Web-Sprachumgebung können Sie eine IMS-Transaktion über das IMS-Web übergeben und die Ausgabe der Transaktion über Ihren Web-Browser empfangen.
Java-Minianwendung	DTW_APPLET	In der Sprachumgebung für Java-Minianwendungen können Sie Java-Minianwendungen in Ihren Net.Data-Anwendungen verwenden. Zum Generieren eines Minianwendungsbefehls müssen Sie die Qualifikationsmerkmale des Minianwendungsbefehls und die Parameterliste der Minianwendung bereitstellen.
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 Datenbankverwaltungssysteme aus.
Oracle	DTW_ORA	In der Oracle-Sprachumgebung können Sie direkt auf Ihre Oracle-Daten zugreifen.
Perl	DTW_PERL	Die Perl-Sprachumgebung interpretiert interne Perl-Prozeduren, die in einem FUNCTION-Block des Net.Data-Makros angegeben sind, oder sie führt externe Perl-Prozeduren 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 Tabellenvariablen zurückgegeben werden.
Sybase	DTW_SYB	In der Sybase-Sprachumgebung können Sie direkt auf Ihre Sybase-Daten zugreifen.
System	DTW_SYSTEM	Die SYSTEM-Sprachumgebung unterstützt das Ausführen von Befehlen und das Aufrufen externer Programme.
Web-Registrierungsdatenbank	DTW_WEBREG	Die Sprachumgebung für Web-Registrierungsdatenbanken 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.
- Verwenden Sie einen Funktionsaufruf an die Sprachumgebung.

Beispiel:

```
%FUNCTION(DTW_SQL) custinfo() {
 select CUSTNAME, CUSTNO from ibmuser.customer
 %}
 ...
%HTML(REPORT){
 @custinfo()
 %}
```

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 Messages and Codes Reference* dokumentiert sind.
- Die Datenbanksprachumgebungen wie die SQL-Sprachumgebung weisen der Variablen RETURN_CODE Fehlercodes (sogenannte SQLCODE-Werte) zu, die durch das Datenbankverwaltungssystem (DBMS) zurückgegeben werden. Weitere Informationen zu den von Ihrem DBMS verwendeten SQLCODE-Werten finden Sie im Handbuch zu Fehlernachrichten Ihres DBMSs.

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 vom Ziel einer Sprachumgebungsanweisung verwiesen wird. Die SQL-Sprachumgebung z. B. führt SQL-Anweisungen aus, und SQL-Anweisungen greifen auf Datenbankdateien zu. Das heißt, daß die Benutzer-ID, unter der Net.Data ausgeführt wird, über die Berechtigung für die Datenbankdateien verfügen muß.

Datensprachumgebungen

Mit den von Net.Data bereitgestellten Datensprachumgebungen können Sie auf Daten von relationalen und hierarchischen Datenbanken und auf andere Datenquellen von einem Net.Data-Makro aus zugreifen. In den folgenden Abschnitten werden die von Net.Data bereitgestellten Sprachumgebungen und ihre Verwendung in Ihren Net.Data-Makros erläutert:

- „Relationale Datenbanksprachumgebungen“ auf Seite 147
- „FFI-Sprachumgebung“ auf Seite 167
- „Web-Registrierungsdatenbank-Sprachumgebung“ auf Seite 168
- „IMS-Web-Sprachumgebung“ auf Seite 172

Relationale Datenbanksprachumgebungen

Net.Data stellt relationale Datenbanksprachumgebungen bereit, um Ihnen den Zugriff auf Ihre relationalen Datenquellen zu erleichtern. Net.Data stellt die folgenden relationalen Datenbanksprachumgebungen zur Verfügung:

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:

Sie müssen über einen ODBC-Treiber und einen Treiber-Manager verfügen. Informationen zur Installation und Konfiguration Ihrer ODBC-Umgebung finden Sie in Ihrer ODBC-Treiberdokumentation.

Stellen Sie sicher, daß die folgende Konfigurationsanweisung in der Initialisierungsdatei enthalten ist und auf einer Zeile steht.

```
ENVIRONMENT (DTW_ODBC) DTWODBC ( IN DATABASE, LOGIN, PASSWORD,
  TRANSACTION_SCOPE, SHOWSQL, ALIGN, START_ROW_NUM, DTW_SET_TOTAL_ROWS)
```

Einschränkungen:

- 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, ICAPI oder GWAPI auf Oracle-Datenbanken zugreifen. Diese Sprachumgebung unterstützt Oracle 7.2, 7.3 und 8.0.

Gehen Sie wie folgt vor, um die Oracle-Sprachumgebung zu verwenden:

Stellen Sie sicher, daß die folgende Konfigurationsanweisung in der Initialisierungsdatei enthalten ist und auf einer Zeile steht.

```
ENVIRONMENT (DTW_ORA)      DTWORA    ( IN DATABASE, LOGIN, PASSWORD,
  TRANSACTION_SCOPE, SHOWSQL, ALIGN, START_ROW_NUM, DTW_SET_TOTAL_ROWS)
```

Wichtig: Informationen zum Definieren der Oracle-Sprachumgebung finden Sie in „Definieren der Oracle-Sprachumgebung“ auf Seite 28.

Einschränkungen:

Bei der Oracle-Sprachumgebung gibt es die folgenden 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:

  ```
  LOGON=admin@ora73
  ```

- Sie müssen bei Verwendung einer anderen Schnittstelle als CGI die Direktverbindung verwenden.
- Gespeicherte Prozeduren werden nicht unterstützt.

SQL-Sprachumgebung

Die SQL-Sprachumgebung bietet Zugriff auf DB2-Datenbanken. Verwenden Sie diese Sprachumgebung für optimale Leistung beim Zugreifen auf DB2.

Gehen Sie wie folgt vor, um die SQL-Sprachumgebung zu verwenden:

Stellen Sie sicher, daß die folgende Konfigurationsanweisung in der Initialisierungsdatei enthalten ist und auf einer Zeile steht.

```
ENVIRONMENT (DTW_SQL)      DTWSQL    ( IN DATABASE, LOGIN, PASSWORD,
  TRANSACTION_SCOPE, SHOWSQL, ALIGN, START_ROW_NUM, DTW_SET_TOTAL_ROWS)
```

Einschränkung:

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.

Sybase-Sprachumgebung

Die Sybase-Sprachumgebung bietet Basiszugriff auf Ihre Sybase-Daten. Sie können von Net.Data über CGI, FastCGI, NSAPI, ISAPI, ICAPI oder GWAPI auf Sybase-Datenbanken zugreifen.

Gehen Sie wie folgt vor, um die Sybase-Sprachumgebung zu verwenden:

Stellen Sie sicher, daß die folgende Konfigurationsanweisung in der Initialisierungsdatei enthalten ist und auf einer Zeile steht.

```
ENVIRONMENT (DTW_SYB)      DTWSYB    ( IN DATABASE, LOGIN, PASSWORD,
  TRANSACTION_SCOPE, SHOWSQL, ALIGN, START_ROW_NUM, DTW_SET_TOTAL_ROWS)
```

Wichtig: Informationen zum Definieren der Sybase-Sprachumgebung finden Sie in „Definieren der Sybase-Sprachumgebung“ auf Seite 31.

Einschränkungen:

Bei der Sybase-Sprachumgebung gibt es die folgenden Einschränkungen:

- Die Sybase-Sprachumgebung unterstützt keine großen Objekte wie Abbilder und Audiodateien. Gespeicherte Prozeduren werden nur für Prozeduren ohne eine Anweisung SELECT unterstützt.
- SQL-Anweisungen im Inline-Anweisungsblock können bis zu 64 KB groß sein. Ihre Datenbank hat eventuell andere Begrenzungen. Ermitteln Sie anhand der Dokumentation für Ihre Datenbank, ob Ihre Datenbank eine niedrigere Begrenzung hat.
- Sie müssen bei Verwendung einer anderen Schnittstelle als CGI die Direktverbindung verwenden.
- Gespeicherte Prozeduren werden nicht unterstützt.

In den folgenden Abschnitten wird beschrieben, wie Sie diese Sprachumgebungen verwenden können.

- „Verwalten von Transaktionen in einer Net.Data-Anwendung“
- „Verwenden großer Objekte“ auf Seite 150
- „Gespeicherte Prozeduren“ auf Seite 154
- „Codieren von DataLink-URL-Adressen in Ergebnismengen“ auf Seite 161
- „Beispiele für die relationale Datenbanksprachumgebung“ auf Seite 163

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 Sie die COMMIT-Anweisung explizit im Makro angeben. Zulässige Werte für TRANSACTION_SCOPE sind MULTIPLE und SINGLE.

MULTIPLE

Gibt an, daß Net.Data alle SQL-Anweisungen vor dem Absetzen einer COMM IT- und etwaigen ROLLBACK-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 ab, die die Datenbank auf ihren ursprünglichen Status zurücksetzt. MULTIPLE ist der Standardwert, wenn TRANSACTION_SCOPE nicht festgelegt wird.

Sie aktivieren diese COMMIT-Methode, indem Sie TRANSACTION_SCOPE auf MULTIPLE setzen.

Beispiel:

```
@DTW_ASSIGN(TRANSACTION_SCOPE,"MULTIPLE")
```

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.

Sie aktivieren diese COMMIT-Methode, indem Sie TRANSACTION_SCOPE auf SINGLE setzen. Beispiel:

```
@DTW_ASSIGN(TRANSACTION_SCOPE,"SINGLE")
```

Sie können mit der SQL-Anweisung COMMIT am Ende jeder beliebigen SQL-Anweisung in Ihrem Makro eine COMMIT-Anweisung absetzen. 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()
  ...
%}
```

Verwenden großer Objekte

Sie können Dateien großer Objekte (LOBs) in DB2-Datenbanken speichern und mit der SQL- oder ODBC-Sprachumgebung für Ihre Web-Anwendungen auf sie zugreifen.

Die SQL- oder ODBC-Sprachumgebung speichert große Objekte weder in Net.Data-Variablen zur Tabellenverarbeitung (wie V1 oder V2) noch in Net.Data-Tabellenfeldern, wenn eine SQL-Abfrage LOBs in einer Ergebnismenge zurückgibt. Anstelle dessen wird das große Objekt nach seiner Feststellung durch Net.Data in einer von Net.Data erstellten Datei gespeichert.

Diese Datei befindet sich in einem durch die Pfadkonfigurationsvariable HTML_PATH angegebenen Verzeichnis. Die Werte der Net.Data-Tabellenfelder und -Variablen zur Tabellenverarbeitung werden auf den Pfad zu dieser Datei gesetzt. Der Zugriff auf die Datei ist auf die Benutzer-ID beschränkt, unter der Net.Data ausgeführt wird.

Der Dateiname, unter dem das große Objekt gespeichert wird, wird dynamisch erstellt und weist folgendes Format auf:

name[.*extension*]

Dabei gilt folgendes:

name Dies ist eine 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 versucht die SQL-Sprachumgebung, die Erweiterung zu ermitteln, indem sie in den ersten Byte der LOB-Datei nach einer Kennung sucht, die angibt, was das große Objekt darstellt. Die SQL-Sprachumgebung erkennt die folgenden Datentypen (in den runden Klammern steht die verwendete Erweiterung):

- Bitmap-Abbild (.bmp)
- Graphical Image Format (.gif)
- JPEG-Abbilddateien (.jpg)
- Tagged Image File Format (.tif)
- PostScript (.ps)
- MIDI-Audiodatei (.mid)
- AIFF-Audiodatei (.aif)
- AVI-Datei (.avi)
- Basisaudiodateien (.au)
- RA-Dateien (.ra)
- WAV-Dateien (.wav)
- Portable Document Format (.pdf)
- MIDI-Sequenzdatei (.rmi)

Wird der Objekttyp des binären großen Objekts (BLOB) nicht erkannt, wird dem Dateinamen keine Erweiterung hinzugefügt.

Wenn in der Makrodatei auf das große Objekt verwiesen wird, gibt die SQL-Sprachumgebung den Dateinamen zurück und stellt ihm die Zeichenfolge /tmplobs/ vor. Dabei wird folgende Syntax verwendet:

/tmplobs/*name*.[*extension*]

Hinweis zur Planung: Jede Abfrage, die LOBs zurückgibt, erstellt im durch die Pfadkonfigurationsvariable HTML_PATH angegebenen Verzeichnis Dateien. Bei der Verwendung von LOBs ist die jeweilige Systemausstattung zu berücksichtigen, da diese Objekte die Systemressourcen schnell aufbrauchen können. Erwägen Sie das Erstellen eines Stapelverarbeitungsprogramms, das das Verzeichnis periodisch aufräumt, oder führen Sie den Dämon `dtwclean` aus. Weitere Informationen hierzu finden Sie in Verwalten temporärer LOBs auf Seite 153. 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: Der Anwendungsbenutzer muß den Dateinamen anklicken, um die Anzeigefunktion aufzurufen, weil die Anwendung eine MPEG-Audiodatei (.MPA) verwendet. Die SQL-Sprachumgebung erkennt diesen Dateityp nicht. Daher wird die Erweiterung mit einer EXEC-Variablen an die Datei angehängt.

```
%DEFINE{
docroot="/usr/lpp/internet/server_root/html"
myFile=%EXEC "rename $(docroot)$(V3) $(docroot)$(V3).mpa"
%}

%{ where rename is the rename command on your operating system %}
%FUNCTION(DTW_SQL) queryData() {
SELECT Name, IDPhoto, Voice FROM RepProfile
   %REPORT{
<P>Here is the information you selected:<P>
   %ROW{
$(myFile)
$(V1) Voice sample <IMG SRC="$(V2)">
<A HREF="$(V3).mpa")>Voice sample</A><P>
%}
%}
%}

%HTML(REPORT){
@queryData()
%}
```

Die Funktion `queryData` gibt folgende HTML-Ausgabe zurück:

```
<P>Here are the images 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.

- V1 ist der Name einer Person. Dabei handelt es sich um unverschlüsselten Text.

- V2 ist das Foto der Person in einer GIF-Datei. Das Bild wird inline angezeigt. Die SQL-Sprachumgebung enthält automatisch das Präfix `/tmplobs/` und die Erweiterung GIF.
- V3 ist ein Beispiel für ein Stimmuster der Person in einer MPA-Datei. Wenn Net.Data ein unbekanntes Dateiformat wie zum Beispiel eine MPA-Datei vorfindet, wird die Datei ohne Dateierweiterung in das temporäre LOB-Verzeichnis geschrieben. Dieses Beispiel zeigt, wie ein solcher Dateityp durch Hinzufügen der Erweiterung mit einer EXEC-Variablen verwendet werden kann. Wenn die Variable *$(V3)* aufgelöst wird, wird vor dem Dateinamen der Pfad `/tmplobs/` eingefügt, zum Beispiel `/tmplobs/sound2a`. Zur Verwendung solcher Dateien kann ein REXX-Programm geschrieben werden, das die Schrägstriche in umgekehrte Schrägstriche ändert und die Dateien umbenennt. Das Stimmuster wird wiedergegeben, wenn der Anwendungsbenutzer `Voice sample` anklickt.

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 Web-Server ausgeführt wird, Schreibzugriff auf das durch HTML_PATH angegebene Verzeichnis hat. (Weitere Informationen finden Sie in „HTML_PATH“ auf Seite 23.)

Verwalten temporärer LOBs:

Net.Data speichert temporäre LOBs im Unterverzeichnis `tmplobs`, unter dem in der Pfadkonfigurationsvariablen HTML_PATH angegebenen Verzeichnis. 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 DB2 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 Datenbank-Server speichern. Net.Data unterstützt die Verwendung gespeicherter Prozeduren unter DB2 über die SQL- und ODBC-Sprachumgebungen.

In diesem Abschnitt werden folgende Themen behandelt:

- „Syntax für gespeicherte Prozeduren“
- „Aufrufen einer gespeicherten Prozedur“ auf Seite 155
- „Übergeben von Parametern“ auf Seite 156
- „Verarbeiten von Ergebnismengen“ auf Seite 157

Syntax für gespeicherte Prozeduren: Die Syntax für die gespeicherte Prozedur verwendet die Anweisung FUNCTION, die Anweisung CALL und wahlfrei einen REPORT-Block.

```
%FUNCTION (DTW_SQL) function_name ([IN datatype arg1, INOUT datatype arg2,
    OUT tablename, ...]) {
  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 155 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.

tablename
: Ist der Name einer 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.

resultsetname

Ist der Name, der ein von einer gespeicherten Prozedur zurückgegebenes Ergebnis 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. Datentypen für gespeicherte Prozeduren

BIGINT	DOUBLEPRECISION	SMALLINT
CHAR	FLOAT	TIME
DATE	INTEGER	TIMESTAMP
DECIMAL	GRAPHIC	VARCHAR
DOUBLE	LONGVARCHAR	VARGRAPHIC
	LONGVARGRAPHIC	

Aufrufen einer gespeicherten Prozedur:

1. Definieren Sie eine Funktion, die einen Aufruf an die gespeicherte Prozedur initialisiert.

   ```
   %FUNCTION (DTW_SQL) function_name()
   ```

2. Geben Sie wahlfrei IN-, INOUT- oder OUT-Parameter für die gespeicherte Prozedur an, einschließlich eines Tabellenvariablennamens zum Speichern der Ergebnismenge in einer Net.Data-Tabelle (Sie müssen eine Net.Data-Tabelle nur dann angeben, wenn die Ergebnismenge in der Net.Data-Tabelle gespeichert werden soll). Sie können für die Tabellennamen bzw. Ergebnismengen auch IN- oder INOUT-Parameter aus einer anderen gespeicherten Prozedur angeben.

   ```
   %FUNCTION (DTW_SQL) function_name (IN datatype
   arg1, INOUT datatype arg2,
   OUT tablename...)
   ```

3. Geben Sie mit Hilfe der Anweisung CALL den Namen der gespeicherten Prozedur an.

   ```
    CALL stored_procedure
   ```

4. Wenn die gespeicherte Prozedur nur eine Ergebnismenge generiert, können Sie wahlfrei 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 wahlfrei einen oder mehrere REPORT-Blöcke 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 {  ...   %}
            ...
      %}
     %}
     ```

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: Sie können eine oder mehrere Ergebnismengen 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, -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 136.

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:

```
%FUNCTION (DTW_SQL) mystoredproc () {
     CALL  myproc
%REPORT {
       ...
  %ROW {  ...   %}
       ...
 %}
%}
```

Alternativ kann die folgende Syntax verwendet werden:

```
%FUNCTION (DTW_SQL) function_name () {
    CALL stored_procedure (resultsetname)

 %REPORT (resultsetname) {
  ...
 %}
%}
```

Beispiel:

```
%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 (mytable1)
%}
```

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 tablename1, tablename2, ...) {
    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 (tablename1)
    ...
  %ROW {  ...   %}
    ...
 %}
 %REPORT (tablename2)
    ...
  %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:

```
%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.

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 Web-Server 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 relationale Datenbanksprachumgebung

Die folgenden Beispiele zeigen, wie Sie die relationalen Datenbanksprachumgebungen 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) sql1() {
create table int_null (int1 int, int2 int)
%}

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

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

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

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

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

Oracle

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="udatabase"
   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"
 Procedure      = "TESTTYPE"
 parm1          = "1"              %{SMALLINT          %}
 parm2          = "11"             %{INT               %}
 parm3          = "1.1"            %{DECIMAL (2,1)     %}
 %}
 %FUNCTION(DTW_SQL)   myProc
  (INOUT SMALLINT               parm1,
   INOUT INT                    parm2,
   INOUT DECIMAL(2,1)           parm3){
CALL $(Procedure)
%}
%HTML(REPORT){
<HEAD>
<TITLE>Net.Data : SQL Stored Procedure: Example '$(MACRO_NAME)'.
</TITLE>
</HEAD>
<BODY BGCOLOR="#BBFFFF" TEXT="#000000" LINK="#000000">
<p><p>
Calling the function to create the stored procedure.
<p><p>
 @CRTPROC()
<hr>
<h2>
Values of the INOUT parameters
 prior to calling the stored procedure:<p>
</h2>
<b>parm1  (SMALLINT)</b><br>
$(parm1)<p>
<b>parm2  (INT)</b><br>
$(parm2)<p>
<b>parm3  (DECIMAL)</b><br>
$(parm3)<p>
<hr>
<h2>
Calling the function that executes the stored procedure.
</h2>
<p><p>
 @myProc(parm1,parm2,parm3)
<hr>
<h2>
Values of the INOUT parameters after
calling the stored procedure:<p>
</h2>
<b>parm1  (SMALLINT)</b><br>
$(parm1)<p>
<b>parm2  (INT)</b><br>
$(parm2)<p>
<b>parm3  (DECIMAL)</b><br>
$(parm3)<p>
</body>
%}
```

Sybase Das folgende Beispiel zeigt ein Makro mit einer Funktionsdefinition DTW_SYB, die die Sybase-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="udatabase"
   table= "utable"
%}


%FUNCTION(DTW_SYB) 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()
%}
```

FFI-Sprachumgebung

Wenn Sie unstrukturierte Textdateien (d. h. unverschlüsselte Textdateien) als Datenquellen verwenden wollen, verwenden Sie die FFI-Schnittstelle (Flat File Interface - Schnittstelle für Flachdateien) und ihre zugehörigen Funktionen zum Öffnen, Schließen, Lesen, Schreiben und Löschen von Dateien auf dem Web-Server. Die Dateisprachenunterstützung verwendet FFI-Funktionen zum Lesen von bzw. Schreiben in Dateien auf dem Web-Server nach Anforderung des Web-Clients über den Browser. FFI zeigt die Datei als eine Satzdatei an. Dabei entspricht jeder Datensatz einer Zeile in einer Net.Data-Makrotabellenvariablen, und jeder Wert in einem Datensatz entspricht einem Feldwert in einer Net.Data-Makrotabellen-variablen. FFI liest Datensätze aus einer Datei in Zeilen einer Net.Data-Makro-tabelle und schreibt Zeilen aus einer Tabelle in Datensätze.

Im Handbuch *Net.Data Reference* finden Sie eine Beschreibung und eine Syntaxdarstellung der integrierten FFI-Funktionen.

Konfigurieren der FFI-Sprachumgebung

Prüfen Sie, ob sich die folgende Konfigurationsanweisung in der Initialisierungsdatei befindet und in einer Zeile steht:

```
ENVIRONMENT (DTW_FILE)   DTWFILE   ( OUT RETURN_CODE )
```

Weitere Informationen zur Net.Data-Initialisierungsdatei sowie zu Anweisungen ENVIRONMENT für die Sprachumgebung finden Sie in „Umgebungskonfigurations-anweisungen“ auf Seite 23.

Aufrufen von integrierten FFI-Funktionen

Rufen Sie eine FFI-Funktion so wie jede andere Funktion auf. Definieren Sie mit einer Anweisung DEFINE die zu übergebenden Parameter als Variablen, z. B. :

```
%DEFINE {
    myFile = "c:/private/myfile"
    myTable = %TABLE
    myWait = "1500"
    myRows = "2"
 %}
```

Verwenden Sie dann eine Funktionsaufrufanweisung zum Aufrufen der Funktion, z. B.:

```
@DTWF_UPDATE(myFile, "Delimited", "|", myTable, myWait, myRows)
```

Beispiel

In diesem Beispiel liest Net.Data den Inhalt der Datei `ffi001.dat` in eine Net.Data-Tabelle und schreibt den Inhalt dieser Tabelle in die Datei `tmp.dat`. Abschließend löscht Net.Data die Datei `tmp.dat`.

```
%DEFINE {
mytable = %TABLE(ALL)
myfile  = "/usr/lpp/netdata/ffi//ffi001.dat"
tmpfile = "/usr/lpp/netdata/ffi/tmp.dat"
%}
%HTML(report){
@DTWF_READ(myfile, "ASCIITEXT", " ", mytable)
@DTW_TB_TABLE(mytable)

@DTWF_WRITE(tmpfile, "ASCIITEXT", " ", mytable)
@DTW_TB_TABLE(mytable)

@DTWF_REMOVE(tmpfile)
%}
```

Web-Registrierungsdatenbank-Sprachumgebung

Die Net.Data-Web-Registrierungsdatenbank bietet permanente Speicherung für anwendungsbezogene Daten. In einer Web-Registrierungsdatenbank können Sie Konfigurationsdaten und andere Daten speichern, auf die Sie während der Ausführung durch Web-gestützte Anwendungen dynamisch zugreifen können. Sie können nur über Net.Data-Makros, die Net.Data und die integrierte Unterstützung für Web-Registrierungsdatenbanken verwenden, und von für diesen Zweck geschriebenen CGI-Programmen aus auf Web-Registrierungsdatenbanken zugreifen. Die Web-Registrierungsdatenbank ist in einem Teil der Betriebssysteme verfügbar. Im Handbuch *Net.Data Reference* finden Sie eine Beschreibung und eine Syntaxdarstellung der integrierten Funktionen für Web-Registrierungsdatenbanken sowie eine Liste der Betriebssysteme, die die Sprachumgebung unterstützen.

Bei der Entwicklung einer Web-Page müssen URL-Adressen standardmäßig direkt in die HTML-Quelle für die Seite gestellt werden. Dies erschwert die Änderung von Programmverbindungen (Links). Durch die statische Beschaffenheit werden zudem die Verbindungsarten begrenzt, die einfach auf eine Web-Page plaziert werden können.

Die Verwendung einer Web-Registrierungsdatenbank zum Speichern von anwendungsbezogenen Daten, z. B. URL-Adressen, kann sich beim Erstellen von HTML-Seiten mit dynamisch festgelegten Programmverbindungen (Links) als nützlich herausstellen.

Daten können durch Anwendungsentwickler und Web-Administratoren 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 Programmverbindungen (Links) HTML-Seiten erstellen.

Daten werden in einer Web-Registrierungsdatenbank 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 10 zeigt eine Beispiel-Web-Registrierungsdatenbank:

Tabelle 10. Beispiel-Web-Registrierungsdatenbank

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 Web-Registrierungsdatenbank:

- In einer Web-Registrierungsdatenbank können Sie Aliasnamen für Server und URL-Adressen speichern und damit die Verlagerung von Anwendungen und Servern erleichtern.
- Anwendungsentwickler können ihre Web-gestützten 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 Web-Registrierungsdatenbank können Sie URL-Suchvorgänge basierend auf Produktname, Landessprache, Hersteller usw. ausführen.

Indexierte Einträge in der Web-Registrierungsdatenbank 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 Index-zeichenfolgewerte kann ihre Eindeutigkeit jedoch bewahrt werden.

Tabelle 11. Indexierte Beispiel-Web-Registrierungsdatenbank

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 Web-Registrierungsdatenbanken als zwei unterschiedliche Einträge behandelt.

Konfigurieren der Web-Registrierungsdatenbank-Sprachumgebung

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 „Umgebungskonfigurations-anweisungen“ auf Seite 23.

Aufrufen der integrierten Funktionen für Web-Registrierungsdatenbanken

Rufen Sie eine Funktion für Web-Registrierungsdatenbanken 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.software.ibm.com/",
    "WORK_URL"
```

Beispiel

Das folgende Beispiel erstellt eine Web-Registrierungsdatenbank 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)

%}
```

IMS-Web-Sprachumgebung

Die IMS-Web-Sprachumgebung gehört zu einer vollständigen Endpunkt-zu-Endpunkt-Lösung zum Ausführen Ihrer IMS-Transaktionen in der World Wide Web-Umgebung über Net.Data. Die IMS-Web-Sprachumgebung stellt folgendes bereit:

- Ein Net.Data-Makro mit folgenden Bestandteilen:
  - Zur Eingabe der Transaktionseingabedaten verwendeter HTML-Code
  - Net.Data-FUNCTION-Block, der die IMS-Web-Sprachumgebung aufruft
  - HTML-Code, der die Ausgabe der Transaktion anzeigt
- Die Quelle einer Transaktions-DLL-Datei, die durch die IMS-Web-Sprachumgebung aufgerufen wird

IMS Web Studio generiert aus der MFS-Quelle (MFS - Message Format Service) Code für die DLL-Datei und das Makro sowie eine Make-Datei zum Erstellen der ausführbaren DLL-Datei oder gemeinsam benutzten Bibliothek für die Transaktion und eine Beispiel-HTML-Seite für die IMS-Web-Net.Data-Anwendung. Nach der Erstellung der ausführbaren Form der DLL-Datei verschiebt der Benutzer die DLL-Datei und die Makros auf den Web-Server, auf dem Net.Data ausgeführt wird. Die Transaktion ist nun zur Ausführung in der Web-Umgebung bereit.

IMS Web verwendet IMS TCP/IP Open Transaction Manager Access (OTMA) Connection zur Kommunikation zwischen dem Web-Server und den IMS-Umgebungen.

Weitere Informationen zur Verwendung von IMS Web finden Sie auf der Home-Page von IMS Web.

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

Konfigurieren der IMS-Web-Sprachumgebung

Zur Verwendung der IMS-Web-Sprachumgebung müssen Sie die Net.Data-Einstellungen für die Initialisierung prüfen und die Sprachumgebung definieren.

Stellen Sie sicher, daß die folgende Konfigurationsanweisung in der Initialisierungsdatei enthalten ist und auf einer Zeile steht.

```
ENVIRONMENT (HWS_LE)       DTWHWS    ( OUT RETURN_CODE )
```

Weitere Informationen zur Net.Data-Initialisierungsdatei sowie zu Anweisungen ENVIRONMENT für die Sprachumgebung finden Sie in „Umgebungskonfigurationsanweisungen“ auf Seite 23.

Wichtig: Informationen zum Definieren der IMS-Sprachumgebung finden Sie in „Definieren der IMS-Web-Sprachumgebung“ auf Seite 26.

Einschränkungen

Die IMS-Web-Sprachumgebung von Net.Data wird nur unterstützt, wenn Net.Data als CGI-Anwendung ausgeführt wird.

Programmiersprachumgebungen

Net.Data stellt die folgenden Sprachumgebungen für das Aufrufen externer Programme bereit:

- „Java-Applet-Sprachumgebung“
- „Java-Anwendungssprachumgebung“ auf Seite 181
- „Perl-Sprachumgebung“ auf Seite 183
- „REXX-Sprachumgebung“ auf Seite 187
- „SYSTEM-Sprachumgebung“ auf Seite 191

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 62.

Java-Applet-Sprachumgebung

In der Java-Applet-Sprachumgebung können Sie einfach HTML-Befehle für Java-Applets in Ihren Net.Data-Anwendungen generieren. Wenn Sie die Java-Applet-Sprachumgebung aufrufen, geben Sie den Namen Ihres Applets an und übergeben Sie alle vom Applet benötigten Parameter. Die Sprachumgebung verarbeitet das Makro und generiert die HTML-Applet-Befehle, mit denen der Web-Browser das Applet ausführt.

Außerdem stellt Net.Data eine Reihe von Schnittstellen zur Verfügung, mit denen Ihr Applet auf Tabellenparameter zugreifen kann. Diese Schnittstellen befinden sich in der Klasse DTW_Applet.class.

In den folgenden Abschnitten wird beschrieben, wie Sie die Java-Applet-Sprachumgebung zur Ausführung Ihrer Java-Applets verwenden können.

Konfigurieren der Java-Applet-Sprachumgebung

Prüfen Sie, ob sich die folgende Konfigurationsanweisung in der Initialisierungsdatei befindet und in einer Zeile steht:

```
ENVIRONMENT (DTW_APPLET)   DTWJAVA    ( OUT RETURN_CODE )
```

Weitere Informationen zur Net.Data-Initialisierungsdatei sowie zu Anweisungen ENVIRONMENT für die Sprachumgebung finden Sie in „Umgebungskonfigurationsanweisungen“ auf Seite 23.

Erstellen von Java-Applets

Vor der Verwendung der Net.Data-Java-Applet-Sprachumgebung müssen Sie ermitteln, welche Applets verwendet werden sollen bzw. welche Applets Sie schreiben müssen. Weitere Informationen zum Erstellen von Applets finden Sie in Ihrer Java-Dokumentation.

Generieren der Applet-Befehle

Sie geben einen Aufruf an die Applet-Sprachumgebung mit einem Net.Data-Funktionsaufruf an. Für den Funktionsaufruf ist keine Deklaration erforderlich. Die Syntax für den Funktionsaufruf sieht wie folgt aus:

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

- DTWA_ gibt den Funktionsaufruf an die Applet-Sprachumgebung an.
- AppletName ist der Name des Applets, für das die Befehle generiert werden.
- parm1 bis parmN sind Parameter, mit denen PARAM-Befehle generiert werden.

Gehen Sie wie folgt vor, um ein Makro zu schreiben, das Applet-Befehle generiert:

1. Definieren Sie im DEFINE-Abschnitt des Makros vom Applet benötigte Parameter. Zu diesen Parametern gehören Applet-Befehlsattribute, Net.Data-Variablen und Net.Data-Tabellenparameter, die Sie als Eingabe für das Applet benötigen. Beispiel:

   ```
   %define{
   DATABASE = "celdial"                <=Net.Data-Variable: Name der Datenbank
   MyGraph.codebase = "/netdata-java/" <=Erforderliches Minianwendungsattribut
   MyGraph.height = "200"              <=Erforderliches Minianwendungsattribut
   MyGraph.width = "400"               <=Erforderliches Minianwendungsattribut
   MyTitle = "This is my Title"        <=Net.Data-Variable: Name    der Web-Seite
   MyTable = %TABLE(all)               <=Tabelle zum Speichern der Abfrageergebnisse
   %}
   ```

2. Wahlfrei: Geben Sie eine Abfrage für die Datenbank an, um eine Ergebnismenge als Eingabe für das Applet zu generieren. Dies ist nützlich, wenn Sie ein Applet verwenden, das ein Diagramm oder eine Tabelle generiert. Beispiel:

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

3. Geben Sie den Funktionsaufruf im Net.Data-Makro an, um die Java-Applet-Sprachumgebung und das Applet aufzurufen. Der Funktionsaufruf gibt den Namen des Applets und die an die Sprachumgebung zu übergebenden Parameter an. Zu diesen Parametern gehören Net.Data-Variablen und Net.Data-Tabellenparameter oder Net.Data-Spaltenparameter, die Sie als Eingabe für das Applet benötigen.

   Beispiel:

   ```
   %HTML(report){                              <=Anfang des HTML-Blocks
   @mySQL(MyTable)                             <=Aufruf an die SQL-Funktion
                                                     mySQL
   @DTWA_MyGraph(MyTitle, DTW_COLUMN(ages) MyTable)
                      <=Funktionsaufruf für Minianwendung
   %}
   ```

Applet-Befehlsattribute: Sie können Attribute für Applet-Befehle an einer beliebigen Stelle in Ihrem Net.Data-Makro angeben. Net.Data ersetzt alle Variablen mit der Form *AppletName.attribute* im Applet-Befehl durch Attribute. Die Syntax zum Definieren eines Attributs in einem Applet-Befehl sieht wie folgt aus:

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

Die folgenden Attribute sind für alle Applets erforderlich:

- *codebase:* Die Speicherposition des Applets, die durch eine URL-Adresse angegeben ist
- *height:* Die Höhe des Applets in Pixel
- *width:* Die Breite des Applets in Pixel

Die folgenden Attribute sind wahlfrei:

- *align:* Die Ausrichtung des Applets
- *alt:* Beliebiger Text, der angezeigt werden soll, wenn der Browser den Befehl APPLET versteht, Java-Applets jedoch nicht ausführen kann
- *archive:* Archiv mit Klassen und anderen Ressourcen
- *hspace:* Anzahl der Pixel auf jeder Seite des Applets
- *name:* Name für das Applet-Exemplar
- *object:* Name der Datei mit einer serialisierten Darstellung eines Applets
- *vspace:* Anzahl der Pixel ober- und unterhalb des Applets

Wenn z. B. Ihr Applet MyGraph heißt, können Sie die erforderlichen Attribute wie folgt definieren:

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

Die tatsächliche Zuordnung muß nicht in einem DEFINE-Abschnitt vorhanden sein. Sie können den Wert mit der Funktion DTW_ASSIGN festlegen. Wenn Sie für die Variable *AppletName.code* keine Variable definieren, fügt Net.Data dem Applet-Befehl den Standardparameter *code* hinzu. Der Wert des Parameters *code* ist *AppletName.class*, wobei *AppletName* der Name des Applets ist.

Applet-Befehlsparameter: Sie definieren eine Liste der an die Java-Applet-Sprachumgebung zu übergebenden Parameter im Funktionsaufruf. Sie können Parameter mit folgenden Elementen übergeben:

- Net.Data-Variablen (einschließlich LIST-Variablen)
- Net.Data-Tabellen
- Spalten von Net.Data-Tabellen

Wenn Sie einen Parameter übergeben, erstellt Net.Data einen Java-Applet-Befehl PARAM in der HTML-Ausgabe mit dem Namen und dem Wert, die Sie dem Parameter zuordnen. Sie können weder Zeichenfolgeliterale noch die Ergebnisse von Funktionsaufrufen übergeben.

Net.Data-Variablenparameter:

Sie können Net.Data-Variablen als Parameter verwenden. Wenn Sie eine Variable im DEFINE-Block des Makros definieren und den Variablenwert im Funktionsaufruf DTWA_*AppletName* übergeben, generiert Net.Data den Befehl PARAM, der den gleichen Namen und Wert wie die Variable hat. Angenommen, die Makroanweisung sieht wie folgt aus:

```
%define{

...

MyTitle = "This is my Title"
%}

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

Net.Data erstellt den folgenden Applet-Befehl PARAM:

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

Net.Data-Tabellenparameter:

Net.Data generiert automatisch den Befehl PARAM mit dem Namen DTW_NUMBER_OF_TABLES bei jedem Aufruf der Java-Applet-Sprachumgebung. Dabei wird angegeben, ob der Funktionsaufruf Tabellenvariablen übergeben hat. Der Wert ist die Anzahl der von Net.Data in der Funktion verwendeten Tabellenvariablen. Wenn im Funktionsaufruf keine Tabellenvariablen angegeben werden, wird der folgende Befehl generiert:

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

Sie können eine oder mehrere Net.Data-Tabellenvariablen als Parameter im Funktionsaufruf übergeben. Wenn Sie eine Net.Data-Tabellenvariable im Funktionsaufruf DTWA_*AppletName* angeben, generiert Net.Data die folgenden Befehle PARAM:

Parameterbefehl für Tabellennamen:

Dieser Befehl gibt die Namen der zu übergebenden Tabellen an. Der Befehl hat die folgende Syntax:

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

Dabei gilt folgendes: `i` ist die Nummer der Tabelle basierend auf der Reihenfolge des Funktionsaufrufs, und *tname* ist der Name der Tabelle.

Parameterbefehl für Zeilen- und Spaltenspezifikation:

PARAM-Befehle werden generiert, um die Anzahl der Zeilen und Spalten einer bestimmten Tabelle anzugeben. Dieser Befehl hat die folgende Syntax:

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

Dabei gilt folgendes: *tname* ist der Name der Tabelle, *rows* ist die Anzahl der Zeilen in der Tabelle, und *cols* ist die Anzahl der Spalten in der Tabelle. Dieses Befehlspaar wird für jede eindeutige, im Funktionsaufruf angegebene Tabelle angegeben.

Parameterbefehl für den Spaltenwert:

Dieser Befehl PARAM gibt den Spaltennamen einer bestimmten Spalte an. Dieser Befehl hat die folgende Syntax:

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

Dabei gilt folgendes: *tname* ist der Tabellenname, *j* ist die Spaltennummer, und *cname* ist der Name der Spalte in der Tabelle.

Parameterbefehl für den Zeilenwert:

Dieser Befehl PARAM gibt die Werte einer bestimmten Zeile und Spalte an. Dieser Befehl hat die folgende Syntax:

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

Dabei gilt folgendes: *tname* ist der Tabellenname, *cname* ist der Spaltenname, *k* ist die Zeilennummer, und *val* ist der Wert, der mit dem Wert in der entsprechenden Zeile und Spalte übereinstimmt.

Tabellenspaltenparameter: Sie können eine Tabellenspalte als einen Parameter in einem Funktionsaufruf übergeben, um Befehle für eine bestimmte Spalte zu generieren. Net.Data generiert die entsprechenden Applet-Befehle nur für die angegebene Spalte. Ein Tabellenspaltenparameter verwendet die folgende Syntax:

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

Dabei gilt folgendes: *x* ist der Name bzw. die Nummer der Spalte in der Tabelle.

Tabellenspaltenparameter verwenden die gleichen Applet-Befehle, die für die Tabellenparameter definiert sind.

Alternativer Text für den Applet-Befehl in nicht Java-fähigen Browsern: Die Variable DTW_APPLET_ALTTEXT gibt den Text an, der in Browsern, die Java nicht unterstützen oder für die Java-Unterstützung inaktiviert ist, angezeigt werden soll. Angenommen, es liegt folgende Variablendefinition vor:

```
%define DTW_APPLET_ALTTEXT = "<P>Sorry, your browser is not Java-enabled."
```

Es wird folgender HTML-Befehl und Text erstellt:

```
<P>Sorry, your browser is not Java-enabled.<BR>
```

Wird diese Variable nicht definiert, wird kein alternativer Text angezeigt.

Java-Applet-Beispiel

Das folgende Beispiel veranschaulicht ein Net.Data-Makro, das die Java-Applet-Sprachumgebung aufruft, und den sich ergebenden Applet-Befehl, der die Sprachumgebung generiert.

Das Net.Data-Makro enthält die folgenden Funktionsaufrufe an die Java-Applet-Sprachumgebung:

```
%define{
DATABASE = "celdial"
DTW_APPLET_ALTTEXT = "<P>Sorry, your browser is not Java-enabled."
DTW_DEFAULT_REPORT = "no"
MyGraph.codebase = "/netdata-java/"
MyGraph.height = "200"
MyGraph.width = "400"
MyTitle = "This is my Title"
%}
%FUNCTION(DTW_SQL) mySQL(OUT table){
select name, ages from ibmuser.guests
%}
%HTML(report){
@mySQL(MyTable)
@DTWA_MyGraph(MyTitle, DTW_COLUMN(ages) MyTable)
%}
```

Die Net.Data-Makrozeilen im DEFINE-Abschnitt geben die Attribute des Applet-Befehls an:

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

Die Sprachumgebung generiert einen Applet-Befehl mit den folgenden Qualifikationsmerkmalen:

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

Net.Data gibt die SQL-Abfrageergebnisse aus dem SQL-Abschnitt des Net.Data-Makros in der Ausgabetabelle `MyTable` zurück. Diese Tabelle wird im DEFINE-Abschnitt angegeben:

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

Der Aufruf an das Applet im Makro wird im HTML-Abschnitt angegeben:

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

Auf Grundlage der Parameter im Funktionsaufruf generiert Net.Data den vollständigen Applet-Befehl mit den Daten zur Ergebnistabelle, wie die Anzahl der Spalten, die Anzahl der zurückgegebenen Zeilen und die Ergebniszeilen. Net.Data generiert einen Parameterbefehl für jede Zelle in der Ergebnistabelle. Siehe folgendes Beispiel:

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

Der Parametername *DTW_MyTable_ages_VALUE_1* gibt die Tabellenzelle (Zeile 1, Spalte `ages`) in der Tabelle `MyTable` an, die den Wert 4 hat. Das Schlüsselwort DTW_COLUMN im Funktionsaufruf an das Applet gibt an, daß Sie nur an den Spaltenaltern der Ergebnistabelle MyTable interessiert sind:

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

Die folgende Ausgabe zeigt den vollständigen Applet-Befehl, den Net.Data für das Beispiel generiert:

```
<applet code = 'MyGraph.class' codebase = '/netdata-java/'
width = '400 'height = '200'>
<param name = 'MyTitle' value = "This is my Title" >
<param name = 'DTW_NUMBER_OF_TABLES' value = "1" >
<param name = 'DTW_TABLE_1_NAME' value = "MyTable" >
<param name = 'DTW_MyTable_NUMBER_OF_ROWS' value = "5" >
<param name = 'DTW_MyTable_NUMBER_OF_COLUMNS' value = "1" >
<param name = 'DTW_MyTable_COLUMN_NAME_1' value = "ages" >
<param name = 'DTW_MyTable_ages_VALUE_1' value = "35">
<param name = 'DTW_MyTable_ages_VALUE_2' value = "32">
<param name = 'DTW_MyTable_ages_VALUE_3' value = "31" >
<param name = 'DTW_MyTable_ages_VALUE_4' value = "28" >
<param name = 'DTW_MyTable_ages_VALUE_5' value = "40" >
<P>Sorry, your browser is not Java-enabled.<BR>
</applet>
```

Verwenden der Net.Data-Java-Applet-Schnittstelle

Net.Data stellt eine Reihe von Schnittstellen in der Klasse DTW_Applet.class bereit. Sie können sie zusammen mit Ihren Java-Applets zur Verarbeitung der Befehle PARAM verwenden, die für Tabellenvariablen generiert werden. Sie können ein Applet erstellen, das diese Schnittstelle um das Aufrufen der Routinen von Ihrem Applet erweitert.

Net.Data stellt die folgenden Schnittstellen bereit:

- **int GetNumberOfTables()** gibt die Anzahl der im Applet-Befehl gefundenen Anzahl Tabellen zurück.
- **String [] GetTableNames()** gibt eine Liste der im Applet-Befehl gefundenen Tabellennamen zurück.
- **int GetNumberOfColumns(String table_name)** gibt die Anzahl der Spalten in der Tabelle table_name zurück.
- **int GetNumberOfRows(String table_name)** gibt die Anzahl der Zeilen in der Tabelle table_name zurück.
- **String[] GetColumnNames(String table_name)** gibt die Namen der Spalten in der Tabelle table_name zurück.
- **String[][] GetTable(String table_name)** gibt einen zweidimensionalen Zeichenfolgebereich mit den Werten der Tabellenzeilen und -spalten zurück.

Verwenden Sie zum Zugriff auf die Schnittstellen das Schlüsselwort EXTENDS in Ihrem Applet-Code, um Ihr Applet als Unterklasse der Klasse DTW_APPLET zu definieren. Siehe folgendes Beispiel:

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

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

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

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

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

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

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

        String [] col_names = GetColumnNames(table_name);

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

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

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

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

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

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.

Einzelangaben zu JDBC können Sie von folgenden Web-Sites abrufen:

- Unter „IBM Software“ finden Sie JDK 1.1 oder höher. Dies ist zur Verwendung von JDBC mit Net.Data erforderlich:

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

- „JavaSoft“ bietet zusätzliche JDBC-Treiber, JDBC-API-Dokumentation und die neuesten Aktualisierungen von JDBC:

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

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 die folgende Konfigurationsanweisung in der Initialisierungsdatei befindet und in einer Zeile steht:

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

Weitere Informationen zur Net.Data-Initialisierungsdatei sowie zu Anweisungen ENVIRONMENT für die Sprachumgebung finden Sie in „Umgebungskonfigurationsanweisungen“ auf Seite 23.

Wichtig: Informationen zum Definieren der Java-Sprachumgebung finden Sie in „Definieren der Java-Sprachumgebung“ auf Seite 27.

Aufrufen von Java-Funktionen

Die Java-Sprachumgebung bietet eine RPC-ähnliche Schnittstelle (Remote Procedure Call - Fernprozeduraufruf). Sie können Java-Funktionsaufrufe von Ihrem Net.Data-Makro aus mit Net.Data-Zeichenfolgen als Parametern absetzen, und Ihre aufgerufene Java-Funktion kann eine Zeichenfolge zurückgeben. Sie müssen die Net.Data-Direktverbindung verwenden, wenn Sie mit der Java-Sprachumgebung arbeiten (weitere Informationen zur Direktverbindung finden Sie in „Verwalten von Verbindungen“ auf Seite 196).

Gehen Sie wie folgt vor, um Java-Funktionen 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.

Erstellen der Java-Funktion: Ändern Sie die Java-Funktionsbeispieldatei `UserFunctions.java`, oder erstellen Sie eine neue Datei auf Grundlage der folgenden Beispieldatei `myfile.java`:

```
====================myfile.java====================
import mypackage.*          <=Funktionen aufnehmen
 public String myfctcall(...parameters from macro...)
{
 return ( mypackage.mymethod(...parameters...));
                            <=übergeordneter Aufruf an Ihre Funktionen
 }

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.

Tabelle 12 beschreibt die Verzeichnis- und Dateinamen für die Dateien auf Ihrem Betriebssystem.

Tabelle 12. 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
UNIX	UserFunctions.java	javaapps
	launchjv	javaapps

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`

Das Schreiben einer JDBC-Anwendung bzw. eines JDBC-Applets ähnelt sehr dem Schreiben einer C-Anwendung mit DB2 CLI oder ODBC zum Zugreifen auf eine Datenbank. Der Hauptunterschied zwischen Anwendungen und Applets ist der, daß eine Anwendung eventuell besondere Software zur Kommunikation mit DB2 benötigt, z. B. DB2 Client Application Enabler. Das Applet hängt von einem Java-fähigen Web-Browser ab. Dafür braucht kein DB2-Code auf dem Client installiert zu sein.

Vor der Verwendung von JDBC sind auf Ihrem System gewisse Konfigurationsschritte auszuführen. Informationen dazu finden Sie auf der Web-Site für DB2-JDBC-Anwendungen und -Applets:

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

Beispiel für die Java-Sprachumgebung

Nach dem Erstellen der Java-Funktion, Definieren der Cliette-Klasse und Konfigurieren von Net.Data können Sie das Makro mit Verweisen auf die Java-Funktion ausführen. **Wichtig:** Starten Sie Connection Manager vor dem Aufrufen des Net.Data-Makros.

Im folgenden Beispiel ruft der Funktionsaufruf *myfctcall* die mit Net.Data ausgelieferte Beispielfunktion über die Cliette DTW_JAVAPPS auf.

```
%FUNCTION (DTW_JAVAPPS) myfctcall( ....parameters from macro ....)

%{ to call the sample provided with Net.Data %}
%FUNCTION (DTW_JAVAPPS) reverse_line(str);

%HTML(report){
you should see the string "Hello World" in reverse.
@reverse_line("Hello World")
You should have the result of your function call.
@myfctcall( ... ....)
%}
```

Perl-Sprachumgebung

Die Perl-Sprachumgebung kann interne Perl-Prozeduren interpretieren, die in einem FUNCTION-Block des Net.Data-Makros angegeben sind, oder sie kann externe Perl-Prozeduren 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 23.

Aufrufen externer Perl-Prozeduren

Aufrufe an externe Perl-Prozeduren 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-Prozedurname, in einem für die Konfigurationsvariable EXEC_PATH angegebenen Pfad in der Net.Data-Initialisierungsdatei aufgelistet ist.

```
%FUNCTION(DTW_PERL) rexx1() {
%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 der Perl-Prozedur. 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 die Perl-Prozedur übergeben. Die Parameter werden auf gleiche Weise an die Perl-Prozedur übergeben wie bei einem Aufruf der Perl-Prozedur über die Befehlszeile. Die Parameter, die der Perl-Prozedur mit dieser Methode übergeben werden, werden als Eingabeparameter behandelt (die der Perl-Prozedur übergebenen Parameter können von der Perl-Prozedur verwendet und bearbeitet werden, Änderungen an den Parametern werden jedoch nicht an Net.Data zurückgemeldet).

Indirekt

Übergeben Sie Parameter direkt beim Aufruf der Perl-Prozedur mit einer der folgenden Methoden:

- Lassen Sie Net.Data Eingabeparameter an die Perl-Prozedur als Umgebungsvariablen übergeben. Die Perl-Prozedur kann die Parameter dann über die Umgebungsvariablen abrufen.
- Lassen Sie die Perl-Prozedur Ausgabeparameter an die Sprachumgebung durch Schreiben in eine benannte Pipe zurückgeben, 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_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 die Perl-Prozedur 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 eine Perl-Prozedur übergeben, die von der Perl-Sprachumgebung aufgerufen wird. Die Perl-Prozedur 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-Abschnitten

REPORT- und MESSAGE-Blöcke sind so wie in jedem anderen FUNCTION-Abschnitt zulässig. Sie werden von Net.Data und nicht von der Sprachumgebung verarbeitet. Eine Perl-Prozedur kann jedoch Text in den Standardausgabedatenstrom schreiben, um als Teil der Web-Page aufgenommen zu werden.

Beispiel für die Perl-Sprachumgebung

Das folgende Beispiel zeigt, wie Net.Data eine Tabelle durch Ausführen der externen Perl-Prozedur generiert.

```
%define {
 c = %TABLE(20)
 rows = "5"
 columns = "5" %}
 %function(DTW_PERL) genTable(in rows, in columns, out table) {
 %exec{ perl.pl
 %}

 %message{
 default: "genTable: Unexpected Error"
 %}
 %}
%HTML(REPORT){
 @genTable(rows, columns, c)
 return code is $(RETURN_CODE)
 %}
 The Perl script (perl.pl):

 open(D2W,"> $ENV{DTWPIPE}");
 print "genTable begins ...

 ";
 $r = $ENV{ROWS};
 $c = $ENV{COLUMNS};
 print D2W "table_ROWS=\"$r\" ";
 print D2W "table_COLS=\"$c\" ";
 print "rows: $r
 ";
 print "columns: $c";
 for ($j=1; $j<=$c; $j++)
 {
 print D2W "table_N_$j=\"COL$j\" ";
 }
 for ($i=1; $i<=$r; $i++)
 {
 for ($j=1; $j<=$c; $j++)
 {
 print D2W "table_V_$i","_","$j=\"| $i $j ¿\" ";
 }
 }
 close(D2W);
```

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.

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 Anweisungen ENVIRONMENT für die Sprachumgebung finden Sie in „Umgebungskonfigurationsanweisungen“ auf Seite 23.

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.exe%}
%}
```

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 20.

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"   %}
%}
```

Der Verweis auf die Net.Data-Variable INPARM1 wird aufgehoben, und die Variable wird 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 Programm mit dieser Methode übergeben werden, werden als Eingabeparameter behandelt (die dem Programm übergebenen Parameter können vom Programm verwendet und bearbeitet werden, Änderungen an den Parametern werden jedoch nicht zurück an Net.Data übertragen).

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 Eingabe-/Ausgabeparameter (INOUT) von der REXX-Sprachumgebung im Variablenpool 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_N.`*`j`* und die Feldwerte `myTable_N.`*`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-Web-Servers; 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 Web-Server.

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:

```
  COL1      COL2
[ 1 1 ]   [ 1 2 ]
[ 2 1 ]   [ 2 2 ]
[ 3 1 ]   [ 3 2 ]
```

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 auf einer Zeile hinzu:

```
ENVIRONMENT (DTW_SYSTEM)   DTWSYS    ( OUT RETURN_CODE )
```

Weitere Informationen zur Net.Data-Initialisierungsdatei sowie zu Anweisungen ENVIRONMENT für die Sprachumgebung finden Sie in „Umgebungskonfigurationsanweisungen“ auf Seite 23.

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 20.

Beispiel 1: Absetzen eines Befehls

```
%FUNCTION(DTW_SYSTEM) sys2() {
      %EXEC { MYPGM %}
 %}
```

Beispiel 2: 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 (die dem Programm übergebenen Parameter können vom Programm verwendet und bearbeitet werden, Änderungen an den Parametern werden jedoch nicht zurück an Net.Data übertragen).

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`.

Es ist nicht empfehlenswert, Tabellen mit vielen Zeilen zu übergeben, weil die Anzahl der Umgebungsvariablen für den Prozeß beschränkt ist.

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.

 @sys1("ValueOfParm1", MYPARM2, MYPARM3)

 <P>
 After the function call, the value of MYPARM2 is "$(MYPARM2)",
 and the value of MYPARM3 is "$(MYPARM3)".
 %}
```

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 Web-Server-APIs“
- „Verwenden von FastCGI“
- „Verwalten von Verbindungen“ auf Seite 196
- „Net.Data-Caching“ auf Seite 200
- „Festlegen der Fehlerprotokollstufe“ auf Seite 226
- „Optimieren der Sprachumgebungen“ auf Seite 226

Stellen Sie zudem sicher, daß Ihr Web-Server entsprechend optimiert wurde. Die Leistung Ihres Web-Servers wirkt sich direkt auf die Antwortzeit aus, unabhängig davon, wie schnell Net.Data ein Makro oder eine Direktanforderung verarbeitet.

Verwenden der Web-Server-APIs

Sie können die Leistung optimieren, indem Sie Net.Data mit einer Web-Server-API wie ICAPI oder GWAPI anstelle von CGI aufrufen. Wenn Net.Data über eine Web-Server-API ausgeführt wird, erfolgt die Ausführung von Net.Data als Thread im Web-Server-Prozeß. Da ein Web-Server-Prozeß Multi-Thread-Fähigkeiten hat, können mehrere Net.Data-Anforderungen im gleichen Adreßraum gleichzeitig verarbeitet werden. Dadurch entfällt der beim Aufrufen von Net.Data als CGI-Prozeß entstehende Systemaufwand.

Hinweis: Die Verwendung einer Web-Server-API optimiert die Leistung ohne Anwendungsisolation. Da Net.Data in einer Multi-Thread-Umgebung ausgeführt wird, können durch benutzerdefinierte Sprachumgebungen hervorgerufene Fehler, inkorrekte Aufrufe oder gar Datenbankausfälle zu Problemen mit dem Web-Server führen. Dies kann möglicherweise zum Absturz des Web-Servers führen. Berücksichtigen Sie bei der Wahl einer der Web-Server-APIs, ob für Ihre Anwendung die Leistung oder die Anwendungsisolation wichtiger ist.

Verwenden von FastCGI

FastCGI bietet optimierte Leistung mit der Zuverlässigkeit von CGI-BIN. Der Einsatz von FastCGI ermöglicht die Ausführung von Makros mit der Geschwindigkeit von API-Servern und der zuverlässigeren Methode, getrennten Speicherbereich zu verwenden. Net.Data wird standardmäßig mit CGI aufgerufen.

Sie können Net.Data mit FastCGI auf allen Servern verwenden, die FastCGI unterstützen.

Informationen zum Konfigurieren für FastCGI finden Sie in „Konfigurieren von Net.Data für FastCGI“ auf Seite 41.

Mit dem Konfigurationsparameter für Prozesse können Sie FastCGI so optimieren, daß die entsprechende Anzahl von Prozessen ausgeführt wird, um die Anzahl eingehender Anforderungen zu bearbeiten. Beispiel: Für einen Kunden gingen durchschnittlich 100 Anforderungen pro Sekunde ein, und jede Anforderung wurde innerhalb einer halben Sekunde verarbeitet. Deshalb wird der Parameter für Prozesse auf 50 eingestellt.

FastCGI wird von den folgenden Sprachumgebungen unterstützt:

- SQL
- Oracle 7.2, 7.3, 8.0
- ODBC
- Sybase
- REXX
- Perl

Anforderung: Im FastCGI-Modus ist für die Sprachumgebungen für Oracle und Sybase eine Direktverbindung 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 ist dies die Datei `httpd.conf`.
   - Bei ICS und Lotus Domino Go Webserver ist dies die Datei `lgw_fcgi.conf`.
2. Ändern Sie den Wert für den Konfigurationsparameter, der die Anzahl von Prozessen angibt:
   - Bei Apache: `Process=`*num*
   - Bei ISC: `NumProcess=`*num*

   Dabei gibt *num* die Anzahl der Prozesse an.

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. Eine virtuelle Java-Maschine, die für die Ausführung von Java-Anwendungen (nicht von Java-Applets) erforderlich ist, stellt ein weiteres Beispiel für eine kostenintensive Startzeit dar. Aufgrund der Arbeitsweise der CGI-Programme fallen diese Startkosten bei jeder Anforderung an den Web-Server an. Net.Data bietet Direktverbindung für die Betriebssysteme OS/2, Windows NT und UNIX, um dauerhaft Verbindungen aufrechtzuerhalten.

In den folgenden Abschnitten wird die Direktverbindung beschrieben.

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 Web-Server-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. 22 zeigt die Interaktion zwischen der Direktverbindung, dem Makro und den Sprachumgebungen.

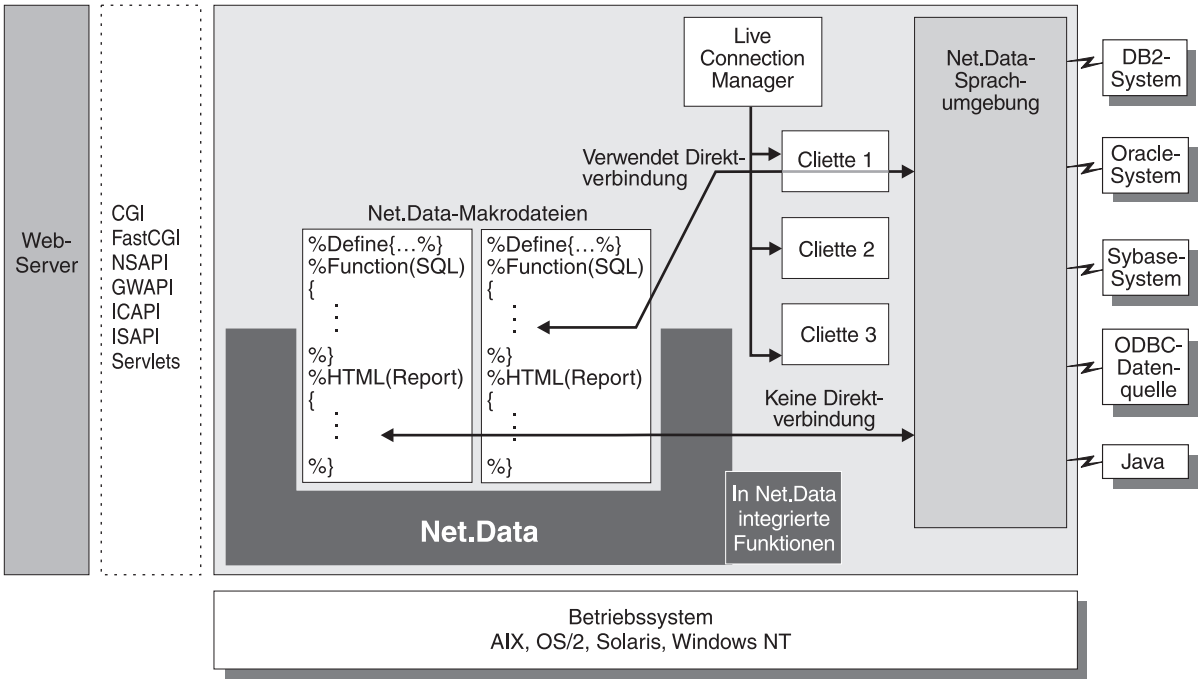


Abbildung 22. Direktverbindung mit Cliettes

In den folgenden Abschnitten wird die Direktverbindung näher beschrieben. Informationen zum Konfigurieren der Direktverbindung finden Sie in „Konfigurieren der Direktverbindung“ auf Seite 33.

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 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 oder START 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 Oracle und Sybase 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 Web-Server 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 33.

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 Connection Manager** aus, und klicken Sie anschließend den Knopf **Starten** an.
3. Wählen Sie **Startart** aus, und klicken Sie anschließend **OK** an.

Verarbeitungsablauf für Net.Data und Direktverbindung

Nach der Konfiguration und dem Start der Datenbank, des Web-Servers und von Connection Manager umfaßt die Net.Data-Verarbeitung bei Aktivierung der Direktverbindung in der Regel die folgenden drei Schritte:

1. Der Web-Server 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 Web-Server 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.

Informationen zum Web-Caching

Viele Softwarekomponenten führen Caching für Web-Anwendungen aus. Es folgen einige Beispiele von Caching-Anwendungen:

- Ein Web-Browser sichert Web-Seiten 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-Proxy-Server-Cache sichert Web-Seiten und zugehörige Objekte auf einem lokalen Server in der Nähe einer Benutzergruppe, um die Netzwerkzugriffszeit auf ferne Web-Server zu verringern. Dadurch kann zum Beispiel die Anzahl der Abrufe angeforderter Objekte durch die Web-Server gesenkt werden. Ein Web-Proxy-Server-Cache ermöglicht zudem, daß von mehreren Benutzern häufig aufgerufene Seiten effektiver benutzt werden können.
- Ein Web-Server 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 Web-Komponenten (Browser, Proxy-Server und Web-Server) 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 Web-Seiten zurückgegebenen HTTP-Kopfzeilen und zugehörige Informationen vom Web-Server, 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. 23 zeigt, daß Net.Data einen Cache-Manager zum Verwalten des Cachings der HTML-Ausgabe von einer Makrodatei verwendet. Zu dieser Ausgabe können Daten aus einer Datenbank gehören.

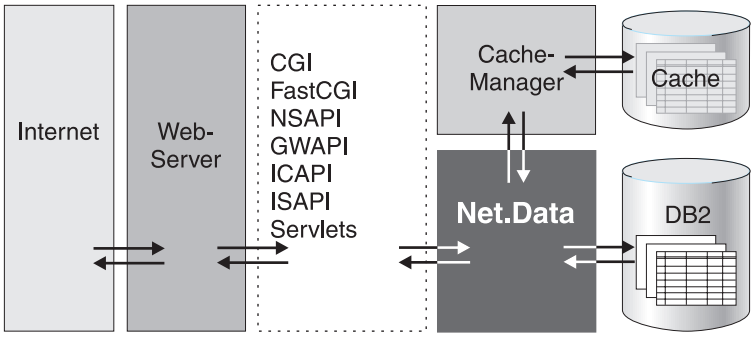


Abbildung 23. Net.Data-Caching

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-Web-Seiten 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-Cachings

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. 24 zeigt einen Cache-Manager, der mit mehreren Makrodateien arbeitet und zwei Caches verwaltet.

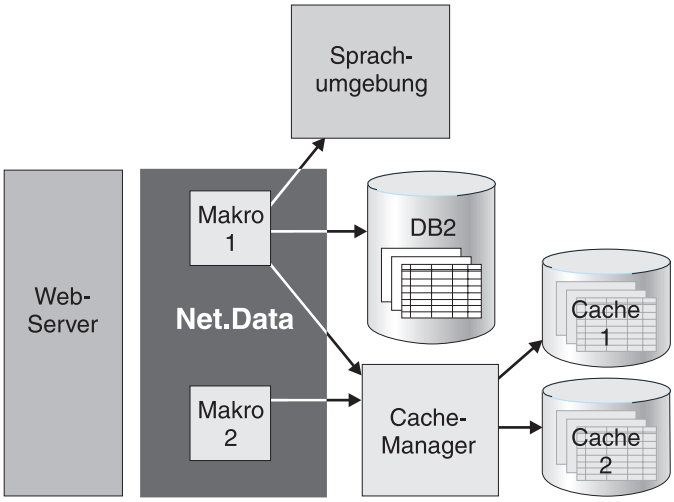


Abbildung 24. Cache-Manager arbeitet mit mehreren Makrodateien und Caches

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 die Makrodatei die Verarbeitung erfolgreich abgeschlossen hat. Dadurch wird sichergestellt, daß nur erfolgreich generierte Web-Seiten 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-Cachings

Net.Data-Caching hat die folgenden Einschränkungen:

Sicherheit

Der Cache-Manager bietet keine Sicherheitsfunktion. Wenn zum Beispiel 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-Cachings

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 13 auf Seite 206 erläutert. Außerdem finden Sie Informationen dazu, wo diese Funktionen beschrieben werden.

Tabelle 13. 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 208
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 Caches“ auf Seite 210
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 13
Integrierte Net.Data-Cache-Funktionen	Sie können den Inhalt eines Net.Data-Caches 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 208.
- 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 Caches und seiner Konfiguration vorgenommen werden müssen.

Cache-Fehler

Der Cache-Manager speichert keine Web-Seiten zwischen, wenn Net.Data einen internen Fehler feststellt, durch den die Makrodatei 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 die Makrodatei jedoch aufgrund einer Konfigurationsvariablen CONTINUE in einem Nachrichtenblock weiter ausführen und beendet Sie normal.
- 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 wahlfrei, 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 {
log=/u/cached/logs/cached.log
port=7177
connection-timeout=0
logging=yes
wrap-log=yes
log-size=64000
trace-flags=D_ALL
}
```

Definieren eines Caches

Definieren Sie die Zeilengruppe für die Cache-Definition, indem Sie Werte für die zulässigen Schlüsselwörter angeben. Die meisten Schlüsselwörter sind wahlfrei 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=`*nn*`B|`*nn*`KB|`*nn*`M`

   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=`*nn*`B|`*nn*`KB|`*nn*`MB`

   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|`*`nn`*`S|`*`nn`*`M|`*`nn`*`H|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)=`*nn*`B|`*nn*`KB|`*nn*`MB`

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)=`*nn*`B|`*nn*`KB|`*nn*`MB`

   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 = `*nn*`S|`*nn*`M|`*nn*`H`

   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:

```
test1
{
caching=on
fssize=5MB
mem-size=10MB
lifetime=6000000
check-expiration=150
datum-memory-limit=5KB
datum-disk-limit=500KB
stat-interval=60
reset-stat-counters=no
root=/u/cached/chaches/cache0
stat-files=/u/cached/logs/chache0.stats
tran-log=/u/cached/logs/cache0.log
tran-logging=yes
wrap-tran-log=yes
tran-log-size=100k
}
```

Starten und Stoppen des Cache-Managers

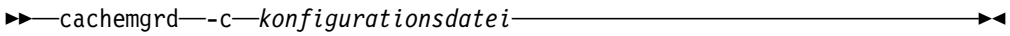
In den folgenden Abschnitten wird beschrieben, wie Sie den Cache-Manager starten und stoppen können.

- „Starten des Cache-Managers“
- „Stoppen des Cache-Managers“

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`.

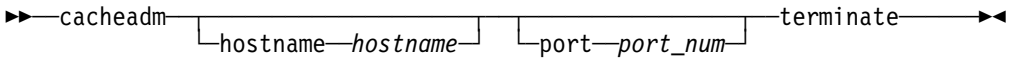
Beispiel:

```
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 Web-Seiten

Sie können eine Web-Seite mit der integrierten Funktion DTW_CACHE_PAGE zwischenspeichern. Wenn Net.Data die Funktion DTW_CACHE_PAGE in der Makrodatei erkennt, steuert es den Cache-Manager an und fängt mit dem Sichern der HTML-Ausgabe für die Makrodatei 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. 25 gezeigt.

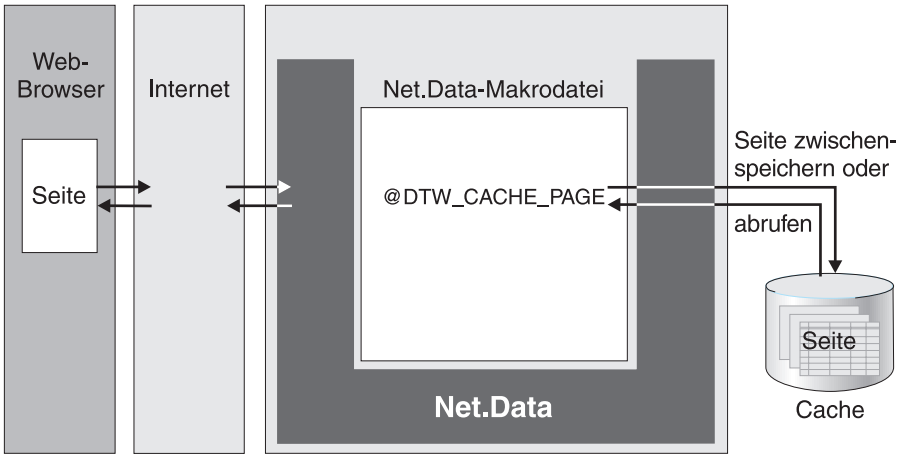


Abbildung 25. Funktion DTW_CACHE_PAGE leitet Caching ein

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 der Makrodatei 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 Web-Seite aus dem Cache abgerufen wurde.

Hinweis zur Leistung: Stellen Sie DTW_CACHE_PAGE() als die erste bzw. eine der ersten Anweisungen in eine Makrodatei, um den Aufwand bei der Ausführung der Makrodatei zu minimieren.

Gehen Sie wie folgt vor, um eine Seite zwischenzuspeichern:

1. Fügen Sie im HTML-Block einer Makrodatei vor der HTML-Codierung die folgende Funktionsanweisung ein:

   ```
   @DTW_CACHE_PAGE("cache_id", cached_page_id, "age", status)
   ```

   Geben Sie mit dieser Funktion an, daß Net.Data die gesamte HTML-Ausgabe vom Makro nach dieser Anweisung zwischenspeichern soll. Plazieren Sie diese Anweisung an den Anfang in der Makrodatei, 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.

   cached_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.d2w", "http://www.mypage.org", "-1", status)
%ENDIF
 ...
<html>

<head>
 <:title>This is the page title</title>
</head>

<body>
 <center>
 <h3>This is the Main Heading</h3>
 <p>It is $(time). Have a nice day!
</body>

</html>
 %}
```

Erweitertes Caching: Dynamisches Ermitteln der Notwendigkeit zur Zwischenspeicherung

Die Funktion DTW_CACHE_PAGE() leitet Caching von ihrer Position in der Makrodatei an ein. In der Regel stellen Sie die Funktion an den Anfang der Makrodatei, 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 der Makrodatei. Diese Entscheidung kann beispielsweise davon abhängig sein, wie viele Zeilen von einer Abfrage oder von einem Funktionsaufruf zurückgegeben werden.

Beispiel: Stellen der Funktion in den HTML-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.d2w", "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.

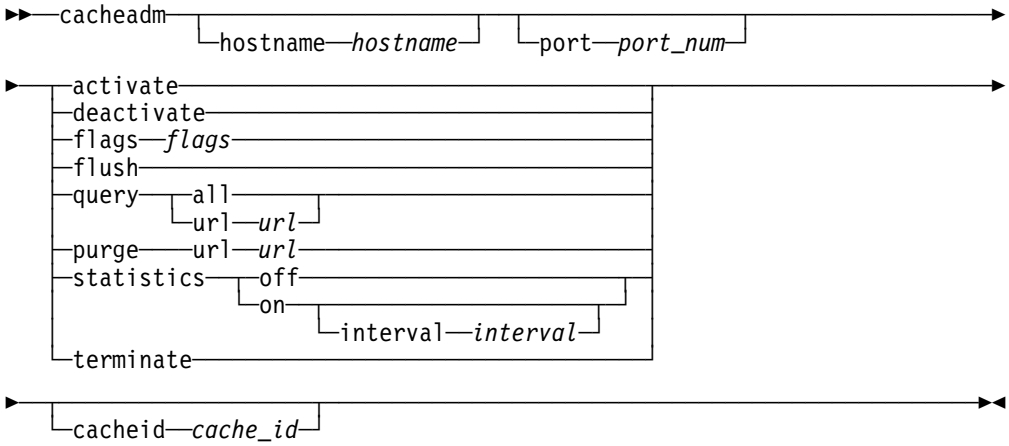
Der Befehl CACHEADM

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 Web-Server angibt, zum Beispiel: `http://www.ibm.com/mydir/page1`.

Das Cache-Protokoll

Mehrere Arten von Statistikdaten für interne Operationen werden aufbewahrt und wahlfrei 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“ auf Seite 224

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 Caches
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 Caches

- 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 Caches
<*statistics*>	Liste der durch Leerzeichen begrenzten und für diesen Cache gesammelten Statistikdaten wie in Tabelle 14 auf Seite 225 dargestellt.

Tabelle 14. Liste der Statistikdaten

Feld-nummer	Inhalt	Beschreibung	Zähler auf Null zurückgesetzt
1	Leseoperationen	Anzahl der Leseoperationen für den Cache	Ja
2	Schreiboperationen	Anzahl der Schreiboperationen für den Cache	Ja
3	Schließoperationen	Anzahl der Schließoperationen für Objekte im Cache	Ja
4	Öffnungs- und Leseoperationen	Anzahl der Öffnungs- und Leseoperationen für Objekte im Cache	Ja
5	Öffnungs- und Schreiboperationen	Anzahl der Öffnungs- und Schreiboperationen für Objekte im Cache	Ja
6	Öffnungs-, Schreib- und Abfrageoperationen	Anzahl der Öffnungs-, Schreib- und Abfrageoperationen für Objekte im Cache	Ja
7	Erfolgreiche Leseoperationen	Anzahl der erfolgreichen Leseoperationen für Objekte im Cache	Ja
8	Erfolgreiche Schreiboperationen	Anzahl der erfolgreichen Schreiboperationen für Objekte im Cache	Ja
9	Erfolgreiche Schreib- und Abfrageoperationen	Anzahl der erfolgreichen Schreib- und Abfrageoperationen für Objekte im Cache	Ja
10	Initialisierungen	Anzahl der mit dem Cache neu erstellten Sitzungen	Ja
11	Beendigungen	Anzahl der mit dem Cache beendeten Sitzungen	Ja
12	Löschoperationen	Anzahl der aus diesem Cache gelöschten Objekte	Nein
13	Speicherbelegung	Von Objekten im Hauptspeicherbereich des Caches belegter Speicher	Nein
14	Plattenbelegung	Von Objekten im Plattenbereich des Caches belegter Speicher	Nein
15	Verfügbarer Hauptspeicher	Für Objekte im Hauptspeicherbereich des Caches noch verfügbarer Speicherbereich	Nein
16	Verfügbarer Plattenspeicherplatz	Für Objekte im Plattenbereich des Caches noch verfügbarer Plattenspeicherplatz	Nein
17	Anzahl Hauptspeicherobjekte	Anzahl der Objekte im Hauptspeicherbereich des Caches	Nein
18	Anzahl Dateiobjekte	Anzahl der Objekte im Plattenbereich des Caches	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 227
- „SYSTEM- und Perl-Sprachumgebungen“ auf Seite 228

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.

SQL-Sprachumgebung

In den folgenden Abschnitten werden einige Leistungsverfahren für die Datenbank und SQL-Sprachumgebung beschrieben. Informationen zu Überlegungen hinsichtlich der DB2-Leistung finden Sie im World Wide Web unter: *http://review.software.ibm.com/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` werden Verbindungen zur Datenbank wesentlich schneller hergestellt. Siehe *DB2 Systemverwaltung*. Bei der Ausführung von Net.Data mit CGI ist dieses Leistungsverfahren eine gute Alternative zur Verwendung der Direktverbindung, die bei FastCGI oder Web-APIs erforderlich ist.
- 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

Verwenden Sie die folgenden Verfahren für die SQL-Sprachumgebung, um die Leistung zu optimieren.

- Verwenden Sie die Net.Data-Variablen START_ROW_NUM und RPT_MAX_ROWS, um die Größe der zurückgegebenen Tabellen zu verringern. 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 die Zeilennummer der ersten zurückzugebenden Zeile an, und RPT_MAX_ROWS gibt die Anzahl der zurückzugebenden Zeilen an.

  Wichtig: Net.Data gibt die Abfrage für jede Anforderung erneut aus, weil die Cursorposition im Verlauf der Anforderungen nicht beibehalten wird.

- Ziehen Sie den Aufruf einer gespeicherten Prozedur in Erwägung, die statisches SQL verwendet. Dynamisches SQL wird zur Laufzeit vorbereitet, während statisches SQL bei der Vorkompilierung vorbereitet wird. Die SQL-Sprachumgebung verwendet dynamisches SQL, so daß SQL-Anweisungen zur Programmlaufzeit ausgeführt werden können. Da die Vorbereitung von Anweisungen zusätzliche Verarbeitungszeit erfordert, ist statisches SQL eventuell effektiver.

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-Prozeduren in der Befehlszeile in der %EXEC-Anweisung auf die Variablen. Verweisen Sie bei internen Perl-Prozeduren direkt in der Perl-Quelle auf die Variablen.

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.

In den folgenden Abschnitten wird Net.Data-Protokollierung beschrieben:

- „Protokollieren von Net.Data-Fehlernachrichten“
- „Protokollieren von Cliette-Nachrichten und Fehlernachrichten bei Direktverbindung“ auf Seite 232

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 in der Makrodatei 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.

  warning Net.Data protokolliert sowohl Warnungen als auch Fehlernachrichten.

In diesem Abschnitt werden die folgenden, die Protokollierung betreffenden Themen erörtert:

- „Planen für das Net.Data-Fehlerprotokoll“ auf Seite 230
- „Steuern der Net.Data-Protokollstufe“ auf Seite 230

- „Arten nicht protokollierter Net.Data-Fehlernachrichten“ auf Seite 230
- „Größe und Archivierung der Net.Data-Fehlerprotokolldatei“ auf Seite 231
- „Net.Data-Fehlerprotokollformat“ auf Seite 231

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: Variable für Speicherposition des Fehlerprotokolls“ auf Seite 15.

- 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 226.

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

:REFID='T2'.

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“ auf Seite 233
- „Steuern der Direktverbindungsprotokollstufe“ auf Seite 233
- „Arten nicht protokollierter Direktverbindungsnachrichten“ auf Seite 233
- „Namen der Direktverbindungsprotokolldateien“ auf Seite 234
- „Größe und Archivierung der Direktverbindungsprotokolldatei“ auf Seite 235
- „Direktverbindungsprotokollformat“ auf Seite 235

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 232.

  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 226 und „Steuern der Direktverbindungsprotokollstufe“.

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 ...
```

Anhang A. Literaturübersicht

Dieser Anhang enthält eine Liste der Dokumente, auf die in diesem Handbuch verwiesen wird.

- „Net.Data Technical Library“

Net.Data Technical Library

Die Net.Data Technical Library auf der Net.Data-Web-Site ist unter folgender Adresse verfügbar:

`http://www.software.ibm.com/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 Messages and Codes Reference	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-Web-Servers. 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 Web-Server.

Überlegungen zu Landessprachen

Net.Data wird mit der gleichen Codepage ausgeführt wie der Web-Server. Damit Net.Data die korrekte Codepage für Ihre länderspezifischen Angaben verwendet, muß der Web-Server 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
startsc -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 INI-Datei auf YES. Net.Data unterstützt derzeit UTF-8-Zeichen im Makro, allerdings nicht UTF-16. Net.Data kann jedoch Datenbankdaten verarbeiten, die in UTF-8 oder UTF-16 codiert sind. Die Net.Data-Ausgabe liegt immer in UTF-8 vor.

Tip 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-SmartGuides

Die Net.Data-SmartGuides bieten Ihnen eine schnelle und einfache Methode, benutzerdefinierte Net.Data-Anwendungen zu erstellen. Wählen Sie einfach einen SmartGuide 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 Web-fähige Drilldown-Anwendung, die es Ihnen ermöglicht, in unterschiedlichen Detaillierungsgraden auf Ihre Daten zuzugreifen. Wahlfrei kann der SmartGuide **Drilldown** eine Verbindung zu Ihrer Datenbank herstellen und Ihre Datenbankinformationen erfassen. Sie können bis zu fünf Web-Berichte nach Ihren Wünschen definieren. Das generierte Makro verwendet in Ihrer Datenbank gespeicherte Primär- und Fremdschlüsselinformationen, um Ihre Web-Berichte 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 Web-fähiges 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-Web-Site unter `http://www.software.ibm.com/data/net.data` finden Sie die neueste Version des Net.Data-SmartGuide-Pakets.

In diesem Anhang werden die folgenden Themen behandelt:

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 SmartGuides

Die Net.Data-SmartGuides werden von der Befehlszeile aus gestartet. Sie befinden sich in der Datei `NetDataSmartGuides.jar`.

Gehen Sie wie folgt vor, um einen SmartGuide 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 wahlfreien Pfad zur Datei `NetDataSmartGuides.jar`.

2. Wenn Sie die Funktion zum Verbinden der Datenbank der SmartGuides **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 wahlfreien Pfad zur Datei NetDataSmartGuides.jar und *[UDBInstallationPath]* für den Pfad zu Ihrer UDB-Installation, zum Beispiel C:\SQLLIB.

3. Starten Sie den SmartGuide.

   - 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 SmartGuides, wie in Abb. 26 dargestellt, aufgelistet werden.

Abbildung 26. Die SmartGuide-Klickstartleiste

4. Klicken Sie den Namen des gewünschten SmartGuides an.

Gehen Sie wie folgt vor, um einen SmartGuide 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 SmartGuides auszuführen:

   ```
   jre -cp [Path]NetDataSmartGuides.jar TaskGuide LaunchPad.class
   ```

   Dabei steht *[Path]* für den wahlfreien Pfad zur Datei `NetDataSmartGuides.jar`.

   Nun wird eine Klickstartleiste geöffnet, in der die verfügbaren SmartGuides, wie in Abb. 26 dargestellt, aufgelistet werden.

2. Wenn Sie die Funktion zum Verbinden der Datenbank der SmartGuides **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 wahlfreien 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 SmartGuides 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“
- „Ausführen von Net.Data SQL Assist“ auf Seite 246

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. Net.Data-Servlets werden in „Net.Data-Servlets“ auf Seite 85 beschrieben.

Sie können NetObjects Fusion (NOF) zur Integration Ihrer Net.Data-Makros verwenden. NOF bietet bessere Integration in Ihre Web-Site-Verwaltung 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 Web-Server 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 Web-Site ein. Dies wird in Abb. 27 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.

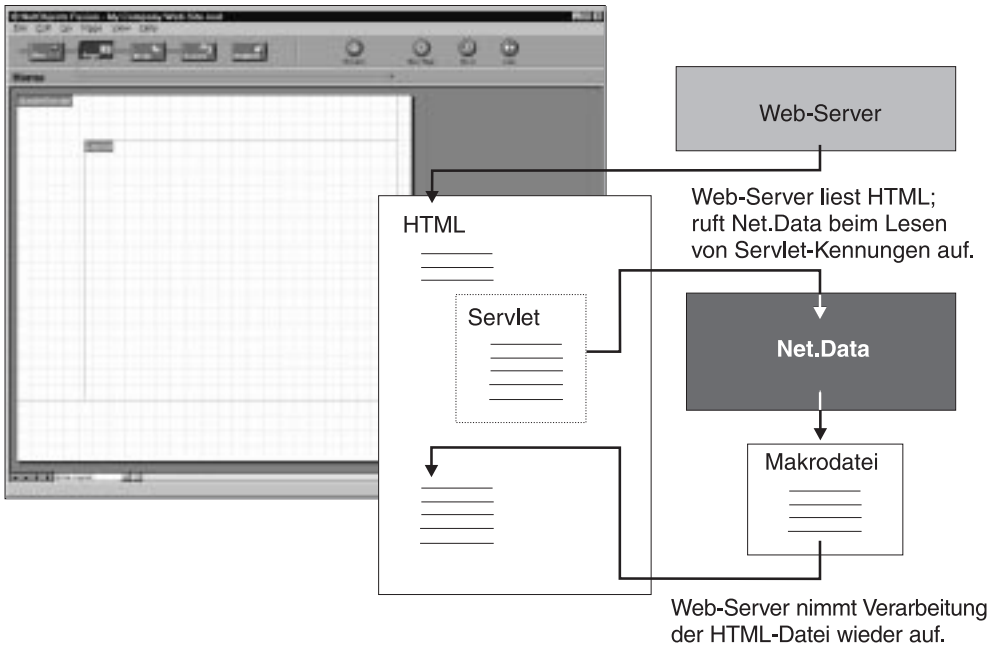


Abbildung 27. Net.Data-Servlets-Plug-ins

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. 28 an.