



# Net.Data Verwaltung und Programmierung für OS/2, Windows NT und UNIX





# Net.Data Verwaltung und Programmierung für OS/2, Windows NT und UNIX

**Anmerkung**

Vor Verwendung dieser Informationen und des darin beschriebenen Produkts sollten die allgemeinen Informationen unter Anhang E, „Bemerkungen“ auf Seite 187 lesen.

Diese Online-Version 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  
© Copyright International Business Machines Corporation 1997, 1998

© Copyright IBM Deutschland Informationssysteme GmbH 1998

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  
August 1998

---

# Inhaltsverzeichnis

<b>Vorwort</b> . . . . .	vii
Informationen zu Net.Data . . . . .	vii
Informationen zu diesem Handbuch . . . . .	ix
Zielgruppe . . . . .	ix
Informationen zu Beispielen in diesem Handbuch . . . . .	ix
 <b>Übersicht über Net.Data</b> . . . . .	 1
 <b>Konfigurieren von Net.Data</b> . . . . .	 5
Informationen zur Net.Data-Initialisierungsdatei . . . . .	7
Informationen zu den Net.Data-Konfigurationsdateien für wahlfreie Komponenten . . . . .	8
Die Konfigurationsdatei für Direktverbindungen . . . . .	8
Die Konfigurationsdatei für den Cache-Manager . . . . .	9
Gemeinsame Abschnitte der Initialisierungs-, Steuer- und Makrodateien von Net.Data . . . . .	10
Anpassen der Net.Data-Initialisierungsdatei . . . . .	13
Konfigurationsvariablenanweisungen . . . . .	14
Anpassen der Pfadkonfigurationsanweisungen . . . . .	18
Umgebungskonfigurationsanweisungen . . . . .	22
Konfigurieren der Direktverbindung . . . . .	25
Konfigurieren von Net.Data für FastCGI . . . . .	31
Konfigurieren von Net.Data zur Verwendung mit den Web-Server-APIs . . . . .	35
Konfigurieren von Net.Data mit Net.Data Administration Tool . . . . .	38
Vorbereitung . . . . .	38
Starten von Administration Tool . . . . .	38
Konfigurieren von Pfadanweisungen . . . . .	39
Konfigurieren von Anschlüssen . . . . .	41
Konfigurieren von Cliettes . . . . .	42
Konfigurieren von Sprachumgebungen . . . . .	47
Definieren von Konfigurationsvariablen . . . . .	51
Angeben von Zugriffsrechten auf Net.Data-Dateien . . . . .	52
 <b>Sichern der Datenbestände</b> . . . . .	 53
Verwenden von Firewalls . . . . .	53
Verschlüsseln Ihrer Daten im Netzwerk . . . . .	56
Verwenden der Authentifizierung . . . . .	56
Verwenden der Berechtigung . . . . .	56
Verwenden von Net.Data-Mechanismen . . . . .	57
 <b>Aufrufen von Net.Data</b> . . . . .	 59
Aufrufen von Net.Data mit einer Makrodatei (Makroanforderung) . . . . .	61
HTML-Programmverbindungen (Links) . . . . .	61
HTML-Formulare . . . . .	62
Aufrufen von Net.Data ohne Makrodatei (Direktanforderung) . . . . .	63
Syntax für Direktanforderungen . . . . .	64
Beispiele für Direktanforderungen . . . . .	67
 <b>Entwickeln von Net.Data-Makros</b> . . . . .	 69
Aufbau einer Net.Data-Makrodatei . . . . .	70
Der DEFINE-Block . . . . .	72

Der FUNCTION-Block	73
HTML-Blöcke	74
Net.Data-Makrovariablen	76
Definieren von Variablen	78
Verweisen auf Variablen	79
Variablenarten	80
Net.Data-Funktionen	90
Definieren von Funktionen	91
Verwenden von Sonderzeichen in Sprachanweisungen	93
Aufrufen von Funktionen	94
Aufrufen gespeicherter Prozeduren	96
MESSAGE-Blöcke	105
Generieren von HTML in einem Makro	107
HTML-Blöcke	107
REPORT-Blöcke	109
Bedingte Logik und Schleifen in einer Makrodatei	111
Bedingte Logik	111
Schleifenkonstrukte	114
Verwenden großer Objekte	115
<b>Verwenden integrierter Funktionen</b>	<b>117</b>
<b>Verwenden der Sprachumgebungen</b>	<b>119</b>
<b>Aufrufen von Net.Data mit Java-Servlets und JavaBeans</b>	<b>121</b>
Net.Data-Servlets	121
Informationen zu Net.Data-Servlets	121
Definieren von Servlets	122
Ausführen von Net.Data-Servlets	122
Net.Data-JavaBeans	128
Informationen zu Net.Data-JavaBeans	128
Definieren und Ausführen der Net.Data-JavaBeans	130
<b>Net.Data-Caching</b>	<b>133</b>
Informationen zum Web-Caching	133
Informationen zum Net.Data-Caching	134
Terminologie für Net.Data-Caching	134
Konzepte des Net.Data-Caching	135
Einschränkungen des Net.Data-Caching	136
Schnittstellen des Net.Data-Caching	136
Planen für den Cache-Manager	138
Cache-Fehler	138
Cache-Kennungen	138
Konfigurieren des Cache-Managers und der Net.Data-Caches	139
Definieren des Cache-Managers	139
Definieren eines Cache	141
Starten / Stoppen des Cache-Managers	146
Starten des Cache-Managers	146
Stoppen des Cache-Managers	147
Caching von Web-Seiten	147
Caching einer Seite	148
Erweitertes Caching: Dynamisches Ermitteln der Notwendigkeit zur Zwischenspeicherung	149
Der Befehl CACHEADM	151

	Das Cache-Protokoll . . . . .	153
	Konfigurieren des Protokolls . . . . .	153
	Format des Cache-Protokolls . . . . .	154
	<b>Optimieren der Leistung . . . . .</b>	<b>157</b>
	Optimieren der Leistung mit den Web-Server-APIs . . . . .	157
	Optimieren der Leistung mit FastCGI . . . . .	159
	Optimieren der Leistung durch Verbindungsverwaltung . . . . .	160
	Informationen zur Direktverbindung . . . . .	161
	Vorteile der Direktverbindung . . . . .	162
	Einsatzmöglichkeiten der Direktverbindung . . . . .	162
	Starten von Connection Manager . . . . .	162
	Verarbeitungsablauf für Net.Data und Direktverbindung . . . . .	163
	Optimieren der Leistung mit Net.Data-Caching . . . . .	164
	Net.Data-Fehlerprotokollierung: Überlegungen zur Leistung . . . . .	165
	Optimieren mathematischer Funktionen . . . . .	165
	<b>Protokollieren von Net.Data-Fehlernachrichten . . . . .</b>	<b>167</b>
	Planen der Protokollüberwachung . . . . .	167
	Steuern der Protokollstufe in den Protokolldateien . . . . .	168
	Arten nicht protokollierter Nachrichten . . . . .	168
	Größe und Archivierung der Protokolldatei . . . . .	168
	Protokolldateiformat . . . . .	169
	<b>Anhang A. Net.Data für AIX . . . . .</b>	<b>171</b>
	Laden gemeinsam benutzter Bibliotheken für Sprachumgebungen . . . . .	171
	Leistungsverbesserung in der REXX-Umgebung . . . . .	172
	<b>Anhang B. Net.Data-SmartGuides . . . . .</b>	<b>173</b>
	Einführung . . . . .	174
	Ausführen der SmartGuides . . . . .	174
	<b>Anhang C. Verwenden von Plug-Ins für NetObjects Fusion (NOF) mit Net.Data-Servlets . . . . .</b>	<b>177</b>
	Informationen zum Plug-In für NetObjects Fusion . . . . .	177
	Installieren des Plug-In für NetObjects Fusion . . . . .	178
	Konfigurieren des Net.Data-Plug-In für NetObjects Fusion . . . . .	178
	Ändern der Plug-In-Merkmale . . . . .	178
	Veröffentlichen von Servlets mit dem NOF-Plug-In . . . . .	182
	<b>Anhang D. Net.Data-Beispielmakro . . . . .</b>	<b>183</b>
	<b>Anhang E. Bemerkungen . . . . .</b>	<b>187</b>
	Marken . . . . .	188
	<b>Glossar . . . . .</b>	<b>189</b>
	<b>Index . . . . .</b>	<b>191</b>





---

# Vorwort

Vielen Dank, daß Sie sich für Net.Data Version 2, dem Entwicklungshilfsprogramm von IBM zum Erstellen dynamischer Web-Seiten 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.

In Net.Data Version 2 wurde die Leistung merklich gesteigert, und außerdem wurden neue Funktionen aufgenommen, mit denen Sie Ihre Internet-Geschäftslösungen mühelos konzipieren und einsetzen können.

---

## Informationen zu Net.Data

Mit Net.Data von IBM können Sie unter Verwendung von Daten aus relationalen und nichtrelationalen Datenbankverwaltungssystemen (DBMS - Database Management Systems), wie DB2, IMS und ODBC-fähige Datenbanken, sowie unter Verwendung von Anwendungen, die in Programmiersprachen wie Java, JavaScript, Perl, C, C++ und REXX geschrieben wurden, dynamische Web-Seiten erstellen.

Net.Data ist ein Makroumwandler, der als Middleware auf einem Web-Server ausgeführt wird. Sie können Net.Data-Anwendungsprogramme, sogenannte Makros, schreiben, die von Net.Data interpretiert und für die Erstellung dynamischer Web-Seiten mit angepaßtem Inhalt auf Grundlage der Eingabe vom Benutzer, des aktuellen Status Ihrer Datenbanken, vorhandener Geschäftslogik und anderer Faktoren, die Sie in den Makroentwurf aufnehmen, verwendet werden.

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:

- Zusammenfassen von Geschäftslogik in Perl-Prozeduren, C- und C++-Anwendungen bzw. REXX-Programmen
- Zugreifen auf Datenbanken wie DB2

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 unterstützt dem Industriestandard entsprechende Schnittstellen wie HTTP (HyperText Transfer Protocol) und CGI (Common Gateway Interface). HTTP wird zwischen dem Browser und dem Web-Server und CGI zwischen dem Web-Server und Net.Data verwendet. Diese Unterstützung ermöglicht Ihnen die Auswahl Ihres bevorzugten Browsers bzw. Web-Servers zur Verwendung mit Net.Data. Die Net.Data-Produktfamilie bietet für die Betriebssysteme OS/400, OS/390, Windows NT, AIX, OS/2, HP-UX, Sun Solaris und SCO ähnliche Funktionen. Net.Data unterstützt außerdem FastCGI und die wesentlichen Web-Server-APIs (Application Programming Interfaces - Anwendungsprogrammierschnittstellen) für mehrere Betriebssysteme.

Außerdem wurden in Net.Data Version 2 andere Leistungserweiterungen implementiert, um Ihre Anwendungsanforderungen zu erfüllen, unter anderem:

- Zwischenspeichern von HTML-Seiten
- Aufrufen von Net.Data ohne Makrodatei (Direktanforderung)
- Minimieren von überflüssigem Leerraum in generierten Web-Seiten
- Optimierte mathematische Funktionen

Net.Data Version 2 enthält darüber hinaus eine Anzahl von Funktionserweiterungen:

- Zu den Erweiterungen der Sprachumgebung gehört die Fähigkeit, mindestens eine Ergebnismenge aus gespeicherten Prozeduren in den SQL- oder ODBC-Sprachumgebungen abzurufen.
- Zu den Erweiterungen der Makrosprache zählen unter anderem:
  - Fähigkeit, Kommentare an einer beliebigen Stelle zu platzieren
  - Verschachtelte IF-Blöcke
  - WHILE-Blöcke
  - Fähigkeit, eine einzelne Ergebnismenge oder mehrere Ergebnismengen von einer gespeicherten Prozedur zu empfangen
  - DBCS-fähige Zeichenfolge- und Wortfunktionen
  - Unterstützung des SQL-Datentyps „Decimal“ in Parameterlisten für gespeicherte Prozeduren
  - Unterstützung von Cookies
  - Unterstützung von dynamisch generierten E-Mail-Nachrichten

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

Für einfachen Zugriff auf Daten von Ihrer Datenbank stellt Net.Data eine Vielzahl von Hilfsprogrammen bereit, darunter Plug-Ins für NetObjects Fusion und SmartGuides für auf Java basierende Entwicklungen. Diese Hilfsprogramme arbeiten mit den Net.Data-Java-Servlets in der Java-Umgebung und ermöglichen Ihnen das Erstellen von Anwendungen, die betriebssystemübergreifend portierbar sind. Plug-Ins für NetObjects Fusion ermöglichen Ihnen die Verwendung des Web-Entwicklungshilfsprogramms von NetObjects Fusion zum Erstellen komplexer Anwendungen mit dynamischen Daten relationaler Datenquellen.

Net.Data-SmartGuides stellen ein grafisches Hilfsprogramm bereit, das Sie durch das Erstellen grundlegender Net.Data-Makros führt.

---

## Informationen zu diesem Handbuch

In diesem Handbuch werden Verwaltungs- und Programmierungskonzepte für Net.Data sowie das Konfigurieren von Net.Data und seiner Komponenten, sein Sicherheitsplan und seine verbesserte Leistung 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 über das IMS-Web auf DB2-Datenbanken und IMS-Transaktionen zugreifen, sowie die Verwendung von Java, REXX, Perl und anderen Programmiersprachen zum Zugriff auf Ihre Daten.

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 neueste Version dieses Handbuchs können über die 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. Betriebssystemspezifische Unterschiede, Net.Data-Nachrichten und andere Informationen werden im Handbuch *Net.Data Reference* beschrieben.

Als Grundlage zum Verständnis der in diesem Handbuch erläuterten Konzepte müssen Sie wissen, wie ein Web-Server funktioniert, einfache SQL-Anweisungen verstehen und HTML-Befehle, einschließlich HTML-Formularbefehle, kennen. Sie sollten sich zudem mit den Informationen in den Handbüchern *Net.Data Reference* und *Net.Data Language Environment Reference* vertraut machen.

## | 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.



# Übersicht über Net.Data

Mit HTML allein können Sie statische Web-Seiten erstellen, das heißt Seiten, die sich nicht ändern, sofern Sie sie nicht editieren. Zum Stellen von dynamischen Daten und Anwendungen ins Web (z. B. aktuelle Verkaufsstatistiken) 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*. Mit Hilfe von Net.Data-Makros können Sie Logik, Variablen, Funktionsaufrufe und Hilfsprogramme zur Berichterstellung verwenden. Ein Makro ist eine Textdatei mit Net.Data-Makrosprachkonstrukten, HTML-Befehlen und Sprachumgebungsanweisungen wie SQL und Perl. Net.Data verarbeitet die Makrodatei und erstellt die HTML-Ausgabe. 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 zeigt die Beziehung zwischen Net.Data und dem Web-Server und den unterstützten Daten sowie Programmiersprachenumgebungen.

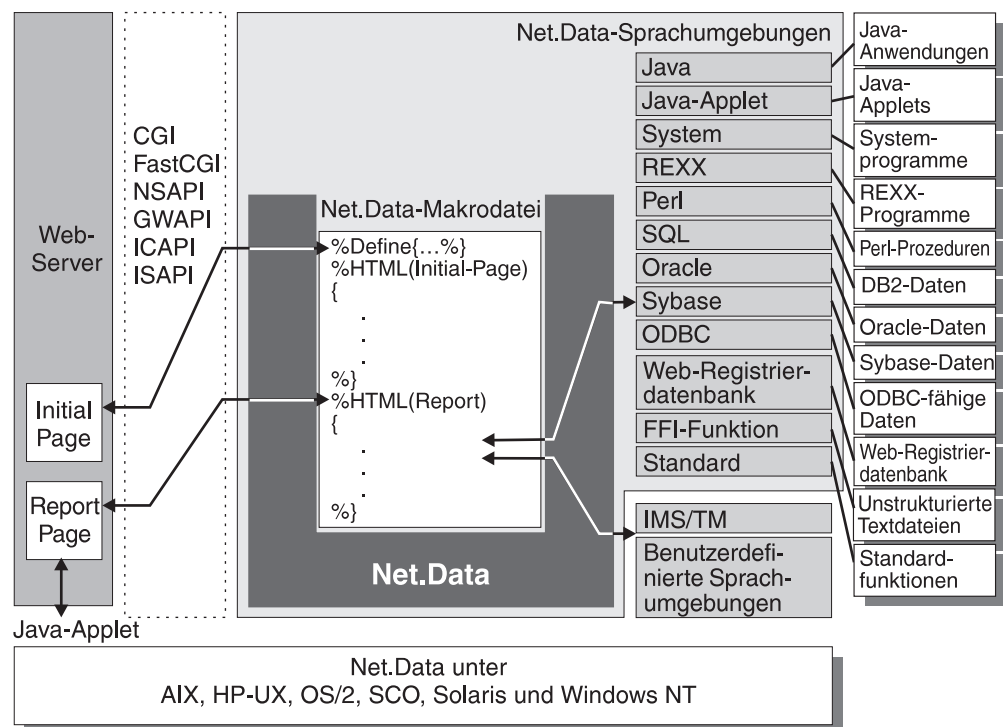


Abbildung 1. Die Beziehung zwischen Net.Data, dem Web-Server und unterstützten Daten sowie Programmquellen

Der Web-Server ruft Net.Data als CGI- oder FastCGI-Prozeß oder den Thread einer Web-Server-API auf, indem er Net.Data als DLL oder gemeinsam benutzte Bibliothek aufruft, wenn eine URL-Adresse empfangen wird, die die Net.Data-Services anfordert. Die URL-Adresse enthält Net.Data-spezifische Informationen, und zwar entweder die zu verarbeitende Makrodatei 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, wo sie mit dem Browser angezeigt wird.

Net.Data ist eine gute Wahl für das Erstellen dynamischer Web-Seiten, 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 bereit, die auf DB2-Datenbanken zugreifen, IMS-Transaktionen über das IMS-Web ausführen oder REXX, Perl und andere Sprachen für Ihre Anwendungen verwenden.

Ein weiterer wichtiger Vorteil ist, daß Net.Data den Zugriff auf viele verschiedene Datenbankquellen unterstützt. Dies ermöglicht Web-Entwicklern die Arbeit mit Daten aus einer Vielzahl von Datenbanken, wie DB2, IMS, Oracle, Sybase und anderer ODBC-fähiger Datenquellen. Weitere Informationen zu den von Net.Data bereitgestellten Sprachumgebungen finden Sie im Handbuch *Net.Data Language Environment Reference*.

Die Net.Data-Web-Anwendungsumgebung bietet die folgenden Funktionen:

#### **Interpretierte Makrosprache**

Die Net.Data-Makrosprache ist eine interpretierte 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 gibt Programmierern einen hohen Grad 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 Rechenoperationen für eine Zeichenfolge aus, die eine gültige Zahl darstellt, einschließlich jener in Exponentialschreibweise. Die Variablen der Makrosprache werden im Abschnitt „Net.Data-Makrovariablen“ auf Seite 76 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)
- Angeben von Zugriffsrechten

Sie können Net.Data mit den folgenden Hilfsprogrammen konfigurieren:

- Texteditor

Bearbeiten Sie die INI-Datei 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 INI-Datei 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 für eine Konfigurationsaufgabe mit einer bestimmten Methode anfangen, sollten Sie die Verwendung dieser Methode für optimale Ergebnisse fortsetzen.

**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	Plattformen:			
	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 Verwaltungshilfsprogramme, 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.

- „Anpassen der Net.Data-Initialisierungsdatei“ auf Seite 13
- „Konfigurieren von Net.Data für FastCGI“ auf Seite 31
- „Konfigurieren von Net.Data zur Verwendung mit den Web-Server-APIs“ auf Seite 35
- „Konfigurieren von Net.Data mit Net.Data Administration Tool“ auf Seite 38
- „Angabe von Zugriffsrechten auf Net.Data-Dateien“ auf Seite 52

---

## 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 der Konfigurationsvariablen steuern verschiedene Aspekte der Net.Data-Operation, wie die Verschlüsselung von Zeichendaten in Web-Seiten, ob Zeichenfolge- und Wortfunktionen DBCS-fähig sind, und den Namen des DB2-Exemplars zum Zugriff auf Datenbankdaten. Die Einstellungen geben auch an, wie Net.Data die Verbindung zu Sprachumgebungen, zu Datenbanken, zur Verbindungsverwaltung und zum Cache herstellt und mit diesen kommuniziert. Darüber hinaus geben sie an, ob die Fehlerprotokollierung aktiviert ist.

Die Sprachumgebungsanweisungen definieren die Net.Data-Sprachumgebungen, die verfügbar sind, und geben spezielle Eingabe- und Ausgabeparameterwerte an, die mit den Sprachumgebungen ausgetauscht werden. Die Pfadanweisungen geben die Verzeichnispfade zu Dateien an, die Net.Data verwendet, wie 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 auf diese Datei hat. Weitere Informationen hierzu finden Sie im Abschnitt „Angaben von Zugriffsrechten auf Net.Data-Dateien“ auf Seite 52.

---

## 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“

„Die Konfigurationsdatei für den Cache-Manager“ auf Seite 9

„Gemeinsame Abschnitte der Initialisierungs-, Steuer- und Makrodateien von Net.Data“ auf Seite 10

### Die Konfigurationsdatei für Direktverbindungen

Eine Direktverbindung bietet Verbindungsverwaltung unter den Betriebssystemen Windows NT, OS/2 und UNIX, um die Leistung durch Beseitigen des Systemaufwands beim Start zu verbessern. Die Konfigurationsdatei für die Net.Data-Direktverbindung enthält Informationen zu einer oder mehreren benannten Clientes. Eine Clientte ist ein Langzeitprozeß, der eine Verbindung zu einer Datenbank verwaltet, oder eine Java-Anwendung, die Net.Data-Makroaufrufe von mehreren Benutzern überdauert. Eine Clientte ist nach ihrem Start so lange vorhanden, bis die Net.Data-Direktverbindung beendet wird. Es können mehrere Clientes mit einer einzelnen Datenbank verbunden sein.

Sie geben als Teil der Clientte-Informationen in der Konfigurationsdatei einen Clientte-Namen, eindeutige Anschlüsse und die Mindest- und Höchstanzahl von Prozessen an. Bei Datenbank-Clienttes können Sie auch für jeden Clientte-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, die Connection Manager startet, Zugriffsrechte auf diese Datei hat. Weitere Informationen hierzu finden Sie im Abschnitt „Angaben von Zugriffsrechten auf Net.Data-Dateien“ auf Seite 52.

## 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 im Abschnitt „Net.Data-Caching“ auf Seite 133 beschrieben. Das Konfigurieren des Cache-Managers wird im Abschnitt „Konfigurieren des Cache-Managers und der Net.Data-Caches“ auf Seite 139 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 Speicher- und Platzbedarf, 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 im Abschnitt „Net.Data-Caching“ auf Seite 133.

**Hinweis zu Berechtigungen:** Stellen Sie sicher, daß die Benutzer-ID, die den Cache-Manager startet, Zugriffsrechte auf diese Datei hat. Weitere Informationen hierzu finden Sie im Abschnitt „Angaben von Zugriffsrechten auf Net.Data-Dateien“ auf Seite 52.

## 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 die Makrodatei*

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 wahlweise Anschluß- und Maschinenvariablen 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 wahlweise Anschluß- und Maschinenvariablen angeben. Diese Werte müssen mit den Werten in der INI-Datei übereinstimmen, sofern verwendet.

Die folgenden Ausschnitte veranschaulichen die Beziehung zwischen einer Makrodatei, einer Initialisierungsdatei und einer Konfigurationsdatei für Direktverbindungen. Von der Makrodatei 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 Cliette.

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 zeigt einen Ausschnitt der Makrodatei, die die Anweisung @DTW\_ASSIGN enthält, die definiert, welche Cliette zum Zugriff auf eine Datenbank verwendet werden soll.

```
< *****>
< ** This is an HTML comment                **>
< ** Access the SAMPLE database using        **>
< ** cliette DTW_SQL:SAMPLE                  **>
< *****>
@DTW_ASSIGN (DATABASE, " SAMPLE ")
@insert_customer
(customer_name, customer_street, customer_city, customer_state,
customer_country, customer_zip, customer_credit, customer_expiry)

< *****>
< ** This is an HTML comment                **>
< ** Process the CELDIAL database using      **>
< ** the cliette DTW_SQL:CELDIAL            **>
< *****>
@DTW_ASSIGN (DATABASE, " CELDIAL ")
@insert_customer
(customer_name, customer_street, customer_city, customer_state,
customer_country, customer_zip, customer_credit, customer_expiry)
```

Abbildung 2. Ausschnitt einer Net.Data-Makrodatei

Beachten Sie, daß die Konfigurationsvariable DATABASE durch die Anweisung ENVIRONMENT der INI-Datei ersetzt wird, um den Cliette-Namen zu generieren. Dies ermöglicht den Zugriff auf mehrere Datenbanken von der gleichen Makrodatei.

Abb. 3 auf Seite 12 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)
CLLETTE "DTW_SQL:${DATABASE}"
```

Abbildung 3. Ausschnitt einer Net.Data-Initialisierungsdatei

Abb. 4 zeigt einen Ausschnitt der Konfigurationsdatei für Direktverbindungen, die die Cliette-Definitionen für DTW\_SQL: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.
#####
CLLETTE 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
}

CLLETTE 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 14
- „Anpassen der Pfadkonfigurationsanweisungen“ auf Seite 18
- „Umgebungsanweisungen“ auf Seite 22

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

```
1 DTW_CM_PORT 7128
2 DTW_INST_DIR c:\db2www
3 DTW_LOG_DIR c:\db2www\logs
4 DB2INSTANCE DB2
5 MACRO_PATH c:\DB2WWW\Macro
6 HTML_PATH c:\www\html
7 INCLUDE_PATH c:\db2www\Macro
8 EXEC_PATH c:\db2www\Macro
9 FFI_PATH c:\pub\ffi;pub\ffi\data
10 ENVIRONMENT (DTW_SQL) [DLL-Pfad] [Parameterliste]
11 ENVIRONMENT (DTW_SYB) [DLL-Pfad] [Parameterliste]
12 ENVIRONMENT (DTW_ORA) [DLL-Pfad] [Parameterliste]
13 ENVIRONMENT (DTW_ODBC) [DLL-Pfad] [Parameterliste]
14 ENVIRONMENT (DTW_DEFAULT) [DLL-Pfad] [Parameterliste]
15 ENVIRONMENT (DTW_APPLET) [DLL-Pfad] [Parameterliste]
16 ENVIRONMENT (DTW_REXX) [DLL-Pfad] [Parameterliste]
17 ENVIRONMENT (DTW_PERL) [DLL-Pfad] [Parameterliste]
18 ENVIRONMENT (DTW_SYSTEM) [DLL-Pfad] [Parameterliste]
19 ENVIRONMENT (DTW_FILE) [DLL-Pfad] [Parameterliste]
20 ENVIRONMENT (DTW_WEBREG) [DLL-Pfad] [Parameterliste]
21 ENVIRONMENT (DTW_JAVAPPS) [Parameterliste]
22 ENVIRONMENT (DTW_DLDPB) [Parameterliste]
23 ENVIRONMENT (USR_TEST) [Parameterliste]
```

- Die Zeilen 1 - 4 definieren Konfigurationsvariablen.
- Die Zeilen 5 - 9 definieren Pfade zu Dateien, die zum Verarbeiten des Makros erforderlich sind.
- Die Zeilen 10 - 23 definieren die Anweisungen ENVIRONMENT, die verfügbar sind.

Abbildung 5. Die Net.Data-Initialisierungsdatei

In den folgenden Abschnitten wird beschrieben, wie Sie die Konfigurationsanweisungen in der INI-Datei 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 in Erstellung befindlichen Web-Seite. Mit Konfigurationsvariablenanweisungen können Sie zudem anwendungsspezifische Variablen definieren.

Die verwendeten Konfigurationsvariablen hängen von den Sprachumgebungen und den verwendeten 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[=]wertefolge`

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

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

- „Konfigurationsvariablen für den Cache-Manager“ auf Seite 15
- „DB2INSTANCE: Variable für DB2-Exemplar“ auf Seite 16
- „DTW\_CM\_PORT: Variable für Anschlußnummer bei Direktverbindung“ auf Seite 16
- „DTW\_INST\_DIR: Variable für Net.Data-Installationsverzeichnis“ auf Seite 17
- „DTW\_LOG\_DIR: Variable für Speicherposition des Fehlerprotokolls“ auf Seite 17
- „DTW\_MBMODE: Variable für Unterstützung der Landessprache“ auf Seite 17
- „DTW\_OPTIMIZE\_MATH: Variable für das Optimieren mathematischer Funktionen“ auf Seite 18
- „DTW\_SMTP\_SERVER: Variable für E-Mail-SMTP-Server“ auf Seite 18

## 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:

- **CACHE\_PORT** gibt die Anschlußnummer an, mit der Net.Data die Verbindung zum Cache-Manager herstellt.
- **CACHE\_MACHINE** gibt den TCP/IP-Host-Namen der lokalen oder fernen Maschine an.

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
<b>Bei Angabe des Cache-Anschlusses</b>	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.
<b>Bei fehlender Angabe des Cache-Anschlusses</b>	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.

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 im Abschnitt „Net.Data-Caching“ auf Seite 133.

### **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:**

`CACHE_PORT anschlußnummer`

#### **Parameter:**

*anschlußnummer*

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

### **CACHE\_MACHINE: 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:**

`CACHE_MACHINE 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, das unter den Betriebssystemen Windows NT, OS/2 und UNIX ausgeführt wird.

Bei DB2 unter den Betriebssystemen OS/2, Windows NT und UNIX muß DB2INSTANCE als eine 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 exemplarname`

### **DTW\_CM\_PORT: Variable für Anschlußnummer bei Direktverbindung**

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

#### **Syntax:**

`DTW_CM_PORT anschlußnummer`

Dabei gibt *anschlußnummer* die für Direktverbindungen verwendete eindeutige Anschlußnummer an.

### **DTW\_INST\_DIR: Variable für Net.Data-Installationsverzeichnis**

Mit der Variable DTW\_INST\_DIR in der Net.Data-INI-Datei werden bestimmte Dateien während der Net.Data-Ausführung lokalisiert. Sie setzen diese Variable während der Installation auf das Verzeichnis (inst\_verz), 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 mit der Konfigurationsvariablen DTW\_LOG\_DIR gespeichert sind. Wenn in der Makrodatei 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_verz\logs\netdata.logs`. Informationen zum Protokollieren von Fehlermeldungen mit Net.Data und zur Variablen DTW\_LOG\_LEVEL finden Sie im Abschnitt „Protokollieren von Net.Data-Fehlermeldungen“ auf Seite 167.

**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 in der Makrodatei auf ERROR bzw. WARNING gesetzt ist.

#### **Syntax:**

`DTW_LOG_DIR \inst_verz\pfad`

**Beispiel:** Konfiguration der Initialisierungsdatei

`DTW_LOG_DIR \inst_verz\protokolle\`

### **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 einer Net.Data-Makrodatei entsprechend einstellen.

#### **Syntax:**

`DTW_MBMODE [=] NO|YES`

**Beispiel:** Aktivieren der Unterstützung in der Landessprache

`DTW_MBMODE = YES`

**Gehen Sie wie folgt vor, um die Einstellung der Initialisierungsdatei in einer Makrodatei zu überschreiben:**

- Fügen Sie der Initialisierungsdatei die Variable DTW\_MBMODE als Parameter von IN für die Anweisung ENVIRONMENT mit Angabe von DEFAULT hinzu, wie im folgenden Beispiel gezeigt wird:

```
ENVIRONMENT (DTW_DEFAULT) DTWFUNC.DLL  
(IN DTW_MBMODE, OUT RETURN_CODE )
```

- Setzen Sie die Konfigurationsvariable DTW\_MBMODE in einer Makrodatei auf den für die Anwendung erforderlichen Wert.

### **DTW\_OPTIMIZE\_MATH: Variable für das Optimieren mathematischer Funktionen**

Optimiert die Leistung mathematischer Funktionen. Wenn DTW\_OPTIMIZE\_MATH auf YES gesetzt ist, verwendet Net.Data mathematische C-Formatierung, wodurch die Funktionen schneller ausgeführt werden. Das Ausgabeformat weicht jedoch von dem Ausgabeformat ab, das erstellt wird, wenn diese Variable nicht angegeben wird.

- Vor allem abschließende Nullen nach Dezimalzeichen werden nicht angezeigt.
- Genauigkeit bedeutet immer die Anzahl signifikanter Ziffern.

Wenn DTW\_OPTIMIZE\_MATH auf NO gesetzt ist, verwendet Net.Data mathematische REXX-Formatierung. Funktionen werden langsamer ausgeführt, liefern jedoch Ausgabeformate, die mit der von früheren Versionen von Net.Data generierten Ausgabe konsistent sind. Der Standardwert ist NO.

#### **Syntax:**

DTW\_OPTIMIZE\_MATH NO|YES

#### ***Gehen Sie wie folgt vor, um die Einstellung der Initialisierungsdatei in einer Makrodatei zu überschreiben:***

Setzen Sie die Konfigurationsvariable DTW\_OPTIMIZE\_MATH mit DTW\_ASSIGN in einer Makrodatei auf den für die Anwendung erforderlichen Wert.

### **DTW\_SMTP\_SERVER: Variable für E-Mail-SMTP-Server**

Gibt den SMTP-Server an, der zum Senden von E-Mail-Nachrichten verwendet werden soll. Der Wert dieser Variable 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 ist *server\_name* der Host-Name bzw. die IP-Adresse des SMTP-Servers, der zum Senden von E-Mail-Nachrichten verwendet wird.

#### **Beispiel:**

DTW\_SMTP\_SERVER = "meinserver"

## **Anpassen der Pfadkonfigurationsanweisungen**

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

- MACRO\_PATH
- EXEC\_PATH
- INCLUDE\_PATH
- FFI\_PATH
- HTML\_PATH

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

### Aktualisierungsrichtlinien:

Mehrere allgemeine Richtlinien gelten für alle Pfadanweisungen.

- Jedes angegebene Verzeichnis wird durch ein Semikolon (;) begrenzt.
- Schrägstriche (/) und umgekehrte Schrägstriche (\) werden auf gleiche Weise behandelt.
- Jede Pfadanweisung kann mehrere Pfade angeben, mit Ausnahme der Anweisung HTML\_PATH, für die nur ein Pfad 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.

**Hinweis:** Net.Data durchsucht alle angegebenen Verzeichnisse, aber nicht die Unterverzeichnisse. Wenn sich zum Beispiel Net.Data-Makros in den folgenden Verzeichnissen befinden, müssen Sie jedes Unterverzeichnis in der Pfadanweisung angeben:

```
/usr/test/client  
/usr/test/assoc  
/usr/test/partner
```

Ihre Anweisung MACRO\_PATH könnte in etwa so aussehen:

```
MACRO_PATH [=] /usr/test/client;usr/test/assoc;usr/test/partner
```

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-Makrodateien durchsucht. Zum Beispiel wird durch Angabe des folgenden URL 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 [=] pfad1;pfad2;...;pfadn
```

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 im Abschnitt „Aufrufen von Net.Data“ auf Seite 59.

**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/SYSADM/macros` gefunden wird, ist der vollständig qualifizierte Pfad `/u/SYSADM/macros/query.d2w`.

## EXEC\_PATH

Die Konfigurationsanweisung `EXEC_PATH` gibt ein oder mehrere Verzeichnisse an, die von `Net.Data` nach einem externen Programm durchsucht werden, 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 [=] pfad1;pfad2;...;pfadn
```

**Beispiel:** Das folgende Beispiel zeigt die Anweisung `EXEC_PATH` in der Initialisierungsdatei und die `EXEC`-Anweisung in der Makrodatei, die 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`.

## 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 Kopfname zu erstellen.



### Syntax:

```
INCLUDE_PATH [=] pfad1;pfad2;...;pfadn
```

**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 Kopffdatei angibt.

Net.Data-Initialisierungsdatei:

```
INCLUDE_PATH = /u/SYSADM/includes;/usr/lpp/netdata/includes;
```

Net.Data-Makro:

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

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

**Beispiel 2:** Das folgende Beispiel zeigt die Anweisung `INCLUDE_PATH` und eine Kopffdatei, die durch einen Unterverzeichnisnamen vollständig qualifiziert ist.

Net.Data-Initialisierungsdatei:

```
INCLUDE_PATH = /u/SYSADM/includes;/usr/lpp/netdata/includes;
```

Net.Data-Makro:

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

Die Kopffdatei wird in den Verzeichnissen `/u/SYSADM/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 Kopffdatei `/usr/lpp/netdata/includes/OE/oeheader.inc`.

### FFI\_PATH

Die Konfigurationsanweisung `FFI_PATH` gibt ein oder mehrere Verzeichnisse an, in dem bzw. 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 [=] pfad1;pfad2;...;pfadn
```

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

Net.Data-Initialisierungsdatei:

```
FFI_PATH = /u/SYSADM/ffi;/usr/lpp/netdata/ffi;
```

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

## HTML\_PATH

Die Konfigurationsanweisung HTML\_PATH gibt an, in welches Verzeichnis Net.Data große Objekte (LOBs) schreibt. Dieser Wert kann durch das Konfigurieren der Initialisierungsdatei geändert werden. Für diese Pfadanweisung ist nur ein Verzeichnispfad zulässig.

### Syntax:

HTML\_PATH [=] pfad

**Beispiel:** Das folgende Beispiel zeigt die Anweisung HTML\_PATH in der Initialisierungsdatei.

Net.Data-Initialisierungsdatei:

HTML\_PATH = /pub/htm

Wenn eine Abfrage ein LOB zurückgibt, sichert Net.Data es im angegebenen Verzeichnis in der Konfigurationsanweisung HTML\_PATH.

**Hinweis zur Leistung:** Berücksichtigen Sie bei der Verwendung von LOBs die Systemeinschränkungen, denn LOBs können Ressourcen schnell aufbrauchen. Weitere Informationen hierzu finden Sie im Abschnitt „Verwenden großer Objekte“ auf Seite 115.

## 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 und die Schnittstelle für Sprachumgebungen werden im Handbuch *Net.Data Language Environment Reference* beschrieben.

Für den Aufruf der Sprachumgebung erfordert Net.Data, daß in der Net.Data-Initialisierungsdatei eine Anweisung ENVIRONMENT für eine Sprachumgebung vorhanden ist.

Net.Data gibt mehrere Variablen an, die sich darauf auswirken, wie Net.Data-Sprachumgebungen Aufrufe an in FUNCTION-Blöcken definierte Funktionen interpretieren. Die Einstellungen dieser Variablen müssen an eine Sprachumgebung übergeben werden, um wirksam zu werden.

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(art) bibliotheksname (parameterliste, ...)  
[CLIETTE "cliette_name"]
```

**Parameter:**

- *art*

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.

- *bibliotheksname*

Der Name der DLL oder gemeinsam benutzten Bibliothek mit den Schnittstellen für Sprachumgebungen, die Net.Data aufruft. Unter OS/2 wird der DLL-Name ohne die Erweiterung *.dll* angegeben. Unter AIX wird der Name des gemeinsam benutzten Objekts mit der Erweiterung *.o* angegeben.

- *parameterliste*

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

Die Parameter werden nach den in der FUNCTION-Blockdefinition angegebenen Parametern in das Feld *parm\_data\_array* der Struktur *dtw\_lei* übergeben. (Informationen zu diesen Strukturen finden Sie im Handbuch *Net.Data Language Environment Reference*.)

Sie müssen diese Parameter als Konfigurationsvariablen oder als Variablen in Ihrer Makrodatei 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.

- *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 der Datenbank- und Java-Anwendung 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 "*typ:db\_name*"

**Parameter:**

*typ* Die der Cliette zugeordnete Sprachumgebung der Datenbank. Eine Liste der gültigen Typen finden Sie auf Seite 50.

*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 INI-Datei verarbeitet, werden die Sprachumgebungs-DLLs bzw. gemeinsam benutzten Bibliotheken nicht geladen. Net.Data lädt eine Sprachumgebungs-DLL bei der ersten Ausführung einer Funktion, die diese Sprachumgebung angibt. Die DLL 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)  ( 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 )
```

Jede Anweisung ENVIRONMENT muß auf einer separaten Zeile stehen.

## Konfigurieren der Direktverbindung

Eine Direktverbindung verwaltet Datenbank- und Java-Anwendungsverbindungen zur Leistungssteigerung von Net.Data unter den Betriebssystemen Windows NT, OS/2 und UNIX. Durch die Verwendung von Connection Manager und Cliettes, d. h. Prozesse, die offene Verbindungen verwalten, beseitigt die Direktverbindung den Systemaufwand für die Verbindungsherstellung zu einer Datenbank bzw. für das Starten einer virtuellen Java-Maschine beim Systemstart.

Eine 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 Cliette. Informationen zur Direktverbindung finden Sie im Abschnitt „Optimieren der Leistung durch Verbindungsverwaltung“ auf Seite 160.

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 25 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 typ:db_name`

**Parameter:**

*typ* Der Name, der eine Sprachumgebung einer Cliette zuordnet. Eine Liste der gültigen Typen finden Sie auf Seite 50.

*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. Ermitteln Sie Werte für `MIN_PROCESS` und `MAX_PROCESS`. `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_anzahl`

`MAX_PROCESS=max_anzahl`

**Parameter:**

*min\_anzahl* 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\_anzahl* Die maximale Anzahl von Cliettes, die gleichzeitig ausgeführt werden können. Für diese Anzahl von Cliettes müssen Sie über genügend eindeutige Anschlußnummern verfügen.

3. Ermitteln Sie die Anschlußnummern, 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 häufig auftretender 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 ist xxx die Kennung der Datenbankart. Die Namen der gültigen ausführbaren Dateien finden Sie in Tabelle 4:

Tabelle 4. 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 ist *db\_name* der Name der Datenbank, der die Cliette zugeordnet ist, zum Beispiel MYDATABASE.

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 25 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:

`CLLETTE DTW_JAVAPPS`

2. Ermitteln Sie Werte für MIN\_PROCESS und MAX\_PROCESS. 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_anzahl`  
`MAX_PROCESS=max_anzahl`

#### **Parameter:**

*min\_anzahl*

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

*max\_anzahl*

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

3. Ermitteln Sie die Anschlußnummern, 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 für den Cliette-Typ in der Initialisierungsdatei übereinstimmen, auf den in der Anweisung ENVIRONMENT 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 INI-Datei, aber nicht in der Konfigurationsdatei für Direktverbindungen verwenden.

Die Variable DATABASE wird in der Net.Data-Makrodatei definiert. Wenn in der Makrodatei 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 Clientette die Verarbeitung beendet. Wenn eine Clientette 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 Clientette-Typ für verschiedene benannte Abschnitte verwenden. Beispielsweise verwenden alle für DB2-Datenbanken relevanten Abschnitte in der Konfigurationsdatei denselben Clientette-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 Clientette von Connection Manager an. Steht der gewünschte Clientette-Typ nicht zur Verfügung, gibt Connection Manager die Nachricht aus, daß keine Clientette 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-Dienst zu starten:***

Unter Windows NT können Sie angeben, daß Connection Manager nicht von der Befehlszeile aus gestartet werden soll, sondern als Windows NT-Dienst. Die Ausführung von Connection Manager als ein Windows NT-Dienst 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 von Net.Data für FastCGI

Mit FastCGI kann Net.Data im FastCGI-Modus unter Apache Web Server und Domino Go Webserver, dem Nachfolgeprodukt von IBM Internet Connection Secure Server (ICSS), ausgeführt werden. Der FastCGI-Modus verbindet die Leistungsstärke anderer Web-API-Programme mit der Zuverlässigkeit von CGI-BIN-Programmen (getrennter Speicherbereich).

### **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 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_verz
-processes prozeßanzahl
-initial-env LIBPATH=bibliothekspfad
-initial-env ORACLE_HOME=oracle_pfad
-initial-env ORACLE_SID=oracle_exemplar
-initial-env SYBASE=sybase_pfad
-initial-env DSQUERY=sybase_exemplar
-initial-env DB2INSTANCE=db2_exemplar
-initial-env RXQUEUE_OWNER_PID=REXX_leistungsvariable
-initial-env LANG=länderspezifisch
```
- Deklarieren Sie das FastCGI-Modul:  

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

## Für Domino Go Webserver unter AIX: Aktualisieren Sie die Dateien

httpd.conf und fcgi.conf:

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

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

- Deklarieren die Anwendung in der Datei fcgi.conf:

```
Local {  
  Exec inst_verz  
  Role Responder  
  URL /fcgi-bin/db2www  
  BindPath /tmp/db2www.ibm  
  NumProcesses prozeßanzahl  
  Environ LIBPATH=bibliothekspfad  
  Environ ORACLE_HOME=oracle_pfad  
  Environ ORACLE_SID=oracle_exemplar  
  Environ SYBASE=sybase_pfad  
  Environ DSQUERY=sybase_exemplar  
  Environ DB2INSTANCE=db2_exemplar  
  Environ RXQUEUE_OWNER_PID=REXX_leistungsvariable  
  Environ LANG=länderspezifisch  
}
```

### Parameter:

*inst\_verz* Der Pfad und Verzeichnisname für die ausführbaren Dateien von Net.Data.

### Für Apache:

AppClass /u/meinverz/apache/fcgi-bin/db2www

### Für Domino Go Webserver:

Exec /u/meinverz/http/fcgi-bin/db2www

**Role Responder** Nur für Domino Go Webserver erforderliches Schlüsselwort.

**URL** Nur für Domino Go Webserver erforderliches Schlüsselwort und 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 Pfadanweisung. Der Pfad des von Net.Data und FastCGI verwendeten eindeutigen UNIX-Socket.

*prozeßanzahl* 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 im Abschnitt „Optimieren der Leistung mit FastCGI“ auf Seite 159.

**Für Apache:**

-processes 7

**Für ICS oder Domino Go Webserver:**

NumProcesses 7

*bibliothekspfad* Die Anweisungen LIBPATH (gemeinsam benutzte Bibliotheken oder DLL), die in jeder Anweisung ENVIRONMENT in der Net.Data-INI-Datei deklariert sind.

**Für Apache:**

-initial-env LIBPATH=/u/meinverz/apache/lib:/u/meinverz/apache:/usr/lib

**Für Domino Go Webserver:**

Environ LIBPATH=/u/meinverz/http/lib:/u/meinverz/http:/usr/lib

*oracle\_pfad* 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\_exemplar* 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\_pfad* 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\_exemplar* 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\_exemplar* Bei Verwendung von DB2 erforderlich. Das Exemplar der DB2-Datenbank.

**Für Apache:**

-initial-env DB2INSTANCE=wwwinst

**Für Domino Go Webserver:**

Environ DB2INSTANCE=wwwinst

*REXX\_leistungsvariable* Bei Verwendung von REXX unter AIX erforderlich. Die Leistungsvariable wird mit FastCGI und REXX unter dem Betriebssystem AIX verwendet. Der Standardwert ist 0. Deklarieren Sie diese Variable bei anderen Produkten und Betriebssystemen in der Net.Data-Makrodatei. Weitere Informationen zu dieser Variablen finden Sie im Anhang „Net.Data for AIX“ des Handbuchs *Net.Data Reference*.

**Für Apache:**

-initial-env RXQUEUE\_OWNER\_PID=0

**Für Domino Go Webserver:**

Environ RXQUEUE\_OWNER\_PID=0

*länderspezifisch* 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 fgi-bin als einen neuen Prozeduraliasnamen in der Datei „srm.conf“ hinzu: `ScripAlias /fcgi-bin/  
/u/meinverz/apache/fci-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/datname.erw/block/  
[?name=wert&...]">beliebiger text</A>`
4. Ändern Sie die Endbenutzerdokumentation für URL-Aufrufe von Net.Data mit FCGI-BIN anstelle von CGI-BIN. Beispiel:  
`http://server/fcgi-bin/db2www/datname.erw/block/[?name=wert&...]`

## 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 im Abschnitt „Optimieren der Leistung mit den Web-Server-APIs“ auf Seite 157 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. Wenn Sie erweitern, erwägen Sie möglicherweise 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/netdata/icapi-lib/db2www: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.dateityp`  
  
Dabei ist *dateityp* bei Windows NT und OS/2 `.dll` und bei 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 stattdessen `dtwisapi.dateityp`. 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.dateityp`  
  
Dabei ist *dateityp* bei Windows NT und OS/2 `.dll` und bei 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="<pfad>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 der Makrodatei ist. Sie können eine beliebige Kombination aus drei Zeichen angeben.  
type=magnus-internal/d2w exts=d2w

4. Versetzen Sie die Net.Data-Makrodateien aus dem Verzeichnis netdata/macro in das Hauptdokumentverzeichnis des Servers:

/netscape/server/docs/

5. Fügen Sie der Anweisung MACRO\_PATH in der Initialisierungsdatei das Hauptdokumentverzeichnis des Servers hinzu. Durch diese Änderung wird Net.Data mitgeteilt, an welcher Position nach den Dateien 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 Hauptdokumentverzeichnis 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) unter den Betriebssystemen Windows NT, AIX und OS/2. Mit diesem Hilfsprogramm können Sie die folgenden Funktionen ausführen:

- „Starten von Administration Tool“
- „Konfigurieren von Pfadanweisungen“ auf Seite 39
- „Konfigurieren von Anschlüssen“ auf Seite 41
- „Konfigurieren von Cliettes“ auf Seite 42
- „Konfigurieren von Sprachumgebungen“ auf Seite 47
- „Definieren von Konfigurationsvariablen“ auf Seite 51

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

## Vorbereitung

1. Planen Sie die Konfiguration der Sprachumgebungen, 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 sich `classes.zip` nach der Installation von JDK in Ihrer Anweisung `CLASSPATH` befindet.
4. Wenn Sie den mit DB2 Universal Database gelieferten IBM JDBC-Treiber installiert haben, 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_verz\connect\verwaltungsverzeichnis`, wobei `inst_verz` das für Net.Data während der Installation angegebene Verzeichnis und `verwaltungsverzeichnis` das Verzeichnis ist, in dem sich die Dateien von Administration Tool befinden.

**Für AIX:** `/usr/lpp/internet/db2www/db2.v2/verwaltungsverzeichnis`, wobei `verwaltungsverzeichnis` 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 Tool** aus.

## Für AIX:

Wechseln Sie in das Net.Data-Installationsverzeichnis (inst\_verz). Geben Sie in der Befehlszeile `ndadmin` ein, um das Hilfsprogramm 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 im Abschnitt „Anpassen der Pfadkonfigurationsanweisungen“ auf Seite 18 beschrieben. Abb. 7 zeigt die Seite **Pfad**.

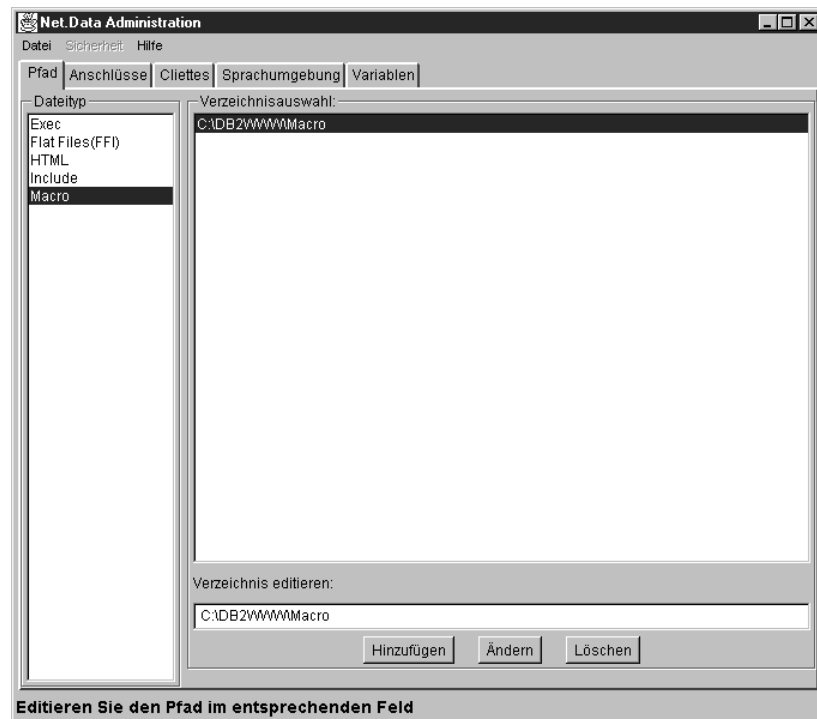


Abbildung 7. Die Seite "Pfad" von Net.Data Administration Tool. Auf dieser Seite können Sie Pfadanweisungen hinzufügen, ändern oder löschen.

**Konfigurationshinweis:** Der Dateityp HTML kann nur einen 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 ausführbare Datei (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**.

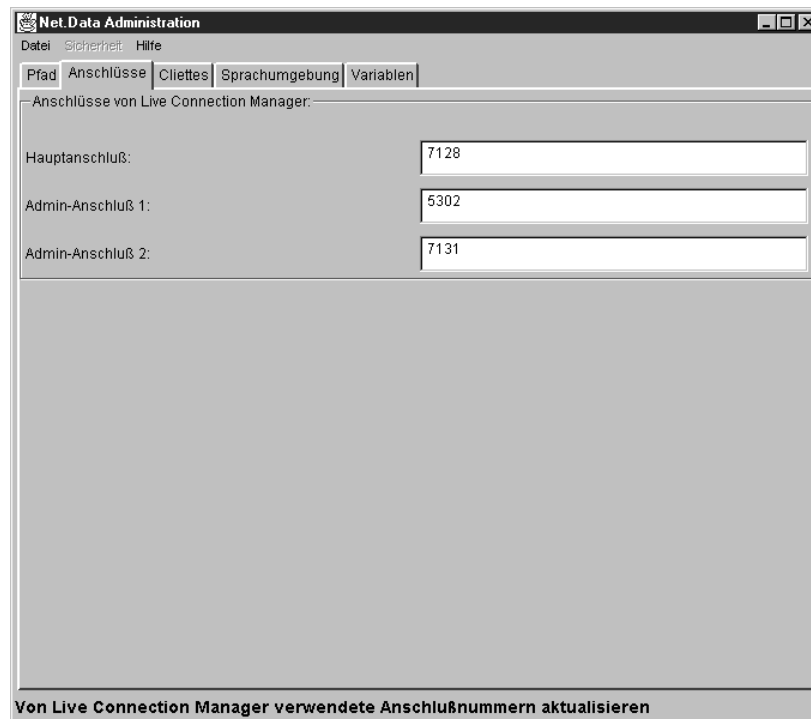


Abbildung 8. Die Seite "Anschlüsse" von Net.Data Administration Tool. Auf dieser Seite können Sie Anschlüsse angeben.

### **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 Clettes

Auf der Seite **Clette** können Sie Datenbank-Clettes für Direktverbindungen hinzufügen, ändern oder löschen, und Sie können ferner Benutzer-IDs und Kennwörter von Datenbanken und Administratoren für Datenbank-Clettes verwalten. Weitere Informationen zu Clettes finden Sie im Abschnitt „Optimieren der Leistung durch Verbindungsverwaltung“ auf Seite 160. Abb. 9 zeigt die Seite **Clette**.

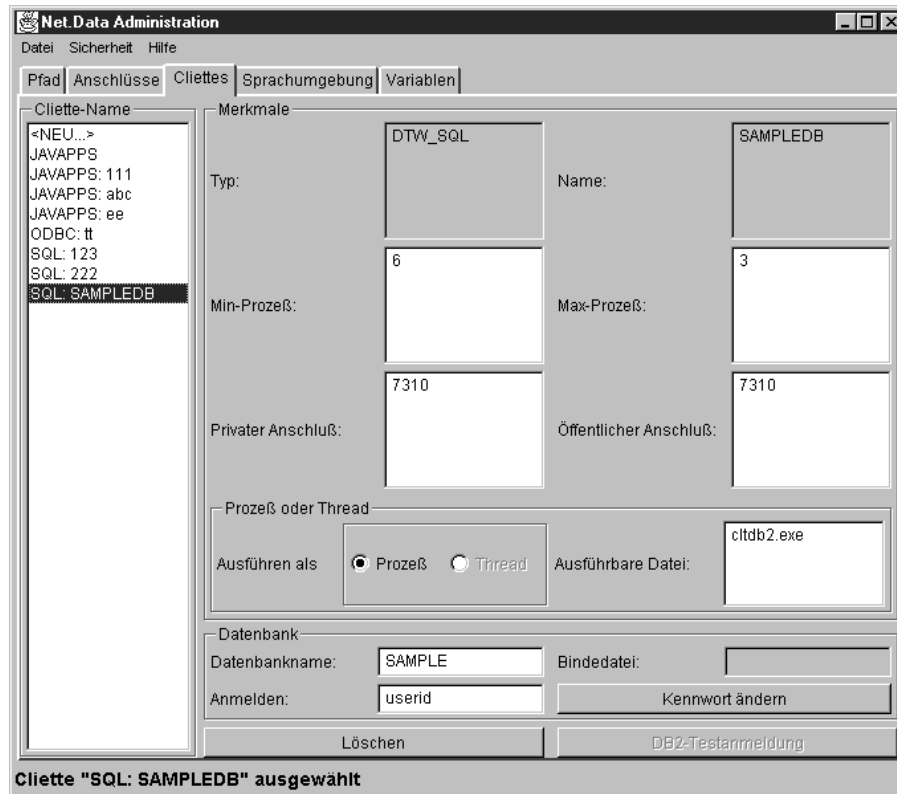


Abbildung 9. Die Seite "Clette" von Net.Data Administration Tool. Auf dieser Seite können Sie Clettes hinzufügen, ändern und löschen.

**Gehen Sie wie folgt vor, um eine Cliette hinzuzufügen:**

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

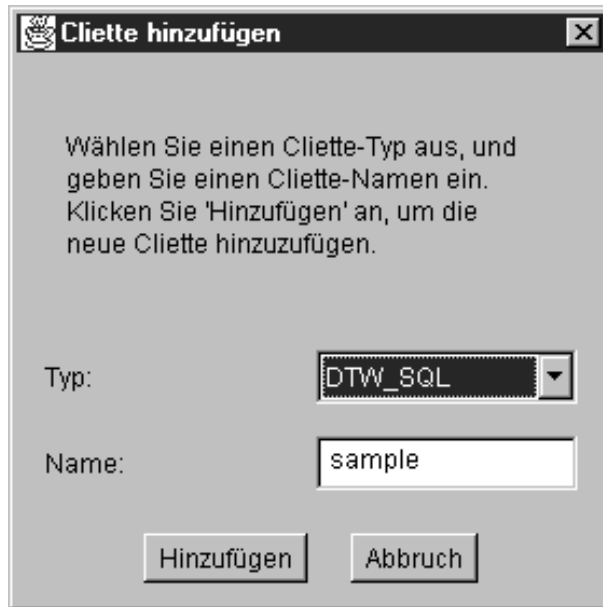


Abbildung 10. Das Fenster "Cliette hinzufügen" von Net.Data Administration Tool. Auf dieser Seite können Sie Cliettes hinzufügen.

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 **Cliette** 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 der Sprachumgebungsart entspricht. Net.Data gibt die erforderlichen Angaben in dieses Feld ein, wenn Sie eine neue Cliette hinzufügen. Die Auswahlmöglichkeiten werden in der Liste **Cliette-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 im Abschnitt „Konfigurieren der Direktverbindung“ auf Seite 25.
  - 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 im Abschnitt „Konfigurieren der Direktverbindung“ auf Seite 25.
  - 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 wie erforderlich die Werte für die Auswahlgruppe **Datenbank**:
  - 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** angegebene 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 **Cliette** 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 **Cliette** 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 <pfad>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 **Cliette** von Administration Tool die Menüoption **Sicherheit -> Verschlüsselung ein** aus. Das Bestätigungsfenster **Verschlüsselung ein** wird geöffnet.
3. Klicken Sie **Ja** an, um den Vorgang fortzusetzen. Das Fenster **Verschlüsselungskennwort** wird geöffnet.
4. Geben Sie das Kennwort für die Berechtigung, um mit Cliettes zu arbeiten, 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 **Cliette** von Administration Tool die Menüoption **Sicherheit -> Verschlüsselung aus** aus. Das Bestätigungsfenster **Verschlüsselung aus** 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 <pfad>dtwcm.cnf wiederherstellen.

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

1. Wählen Sie auf der Seite **Cliette** von Administration Tool die Menüoption **Sicherheit -> Verschlüsselungskennwort ändern** aus. Das Bestätigungsfenster **Verschlüsselungskennwort ändern** 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 **Cliette** 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 im Abschnitt „Umgebungskonfigurationsanweisungen“ auf Seite 22 ausführlich behandelt. Abb. 11 zeigt die Seite **Sprachumgebung**.

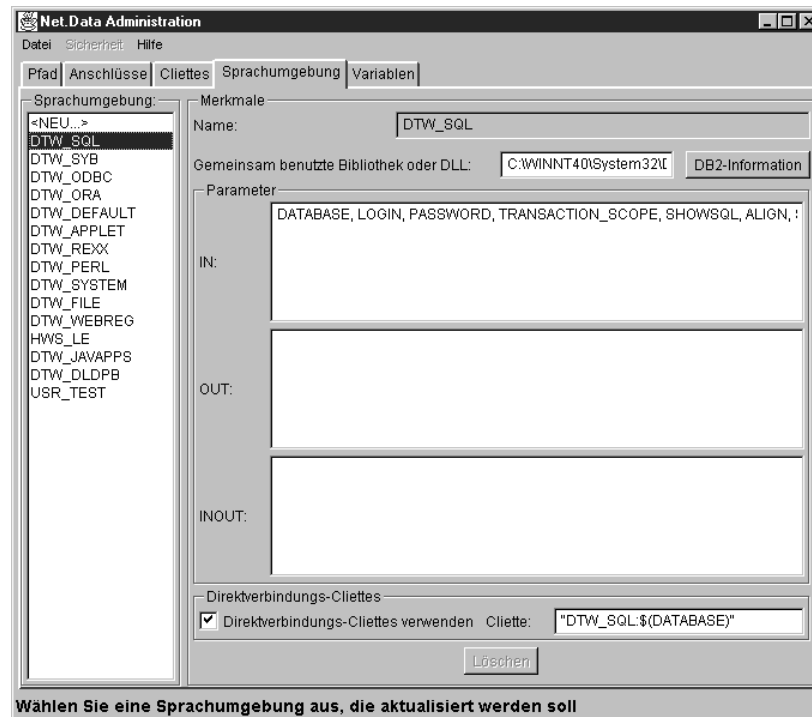


Abbildung 11. Die Seite "Sprachumgebung" von Net.Data Administration Tool. Auf dieser Seite können Sie Sprachumgebungen angeben.

### 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 im Feld ein, und klicken Sie den Knopf **Hinzufügen** an. Das Fenster **Sprachumgebung hinzufügen** wird geöffnet.

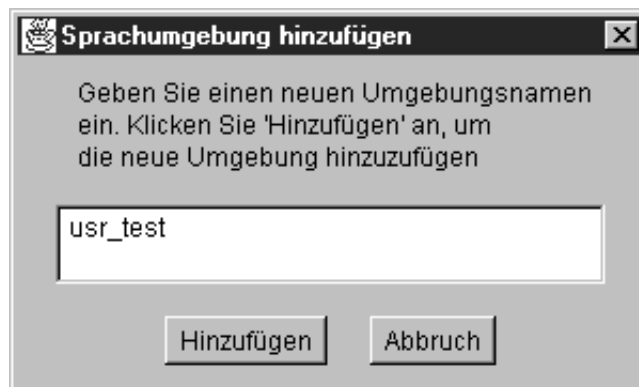


Abbildung 12. Das Fenster "Sprachumgebung hinzufügen" von Net.Data Administration Tool. Auf dieser Seite können Sie eine neue Sprachumgebung angeben.

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 Clientette werden in der Auswahlgruppe **Merkmale** angezeigt.
3. Ändern Sie die Merkmale in der Auswahlgruppe **Merkmale** wie in Abb. 12 gezeigt wie erforderlich:
  - a. Geben Sie im Feld **Name** den Namen der Sprachumgebung an. Dieser Name entspricht der zum Definieren einer Clientette verwendeten Sprachumgebungsart. Sie können diesen Wert ändern, indem Sie in der Liste **Sprachumgebung** einen anderen Namen doppelt anklicken. Weitere Informationen zu Sprachumgebungsarten finden Sie im Abschnitt „Umgebungs-konfigurationsanweisungen“ auf Seite 22.
  - 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 auf Seite 49 gezeigt.

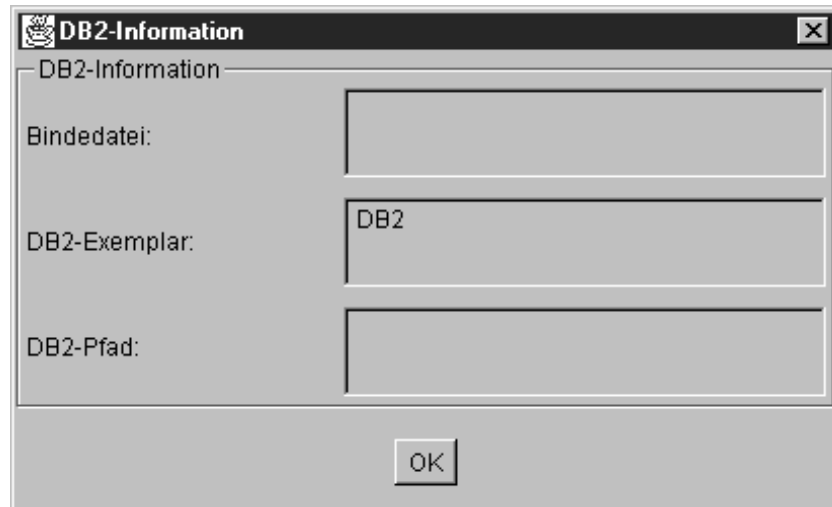


Abbildung 13. Das Fenster "DB2-Information" von Net.Data Administration Tool. Auf dieser Seite können Sie spezifische DB2-Datenbankinformationen angeben.

Geben Sie die Werte für die DB2-Umgebungsvariablen an:

- 1) Geben Sie im 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:**

*art:name*

Dabei gilt folgendes:

*art* Die Sprachumgebung für die Cliette. Sie kann einen der folgenden Werte annehmen:

**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 Fehlermeldungen auswählen. Abb. 14 zeigt die Seite **Variablen**.

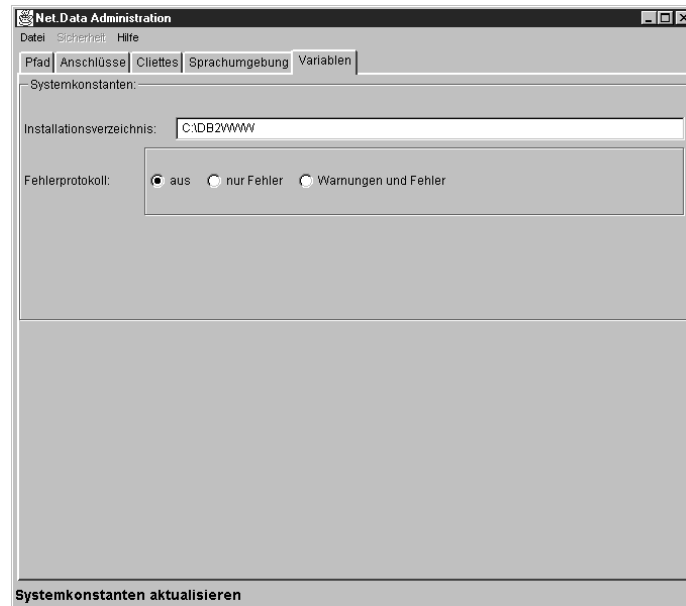


Abbildung 14. Die Seite "Variablen" von Net.Data Administration Tool. Auf dieser Seite können Sie Initialisierungsvariablen angeben.

### **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 **Verzeichnis 'Install'** den Pfad für das Verzeichnis ein, in dem die Protokolldatei gespeichert werden soll. Der Standardwert ist `\inst_verz\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 Fehlermeldungen 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.

---

## Angeben von Zugriffsrechten auf Net.Data-Dateien

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 von Net.Data verwendeten Dateien verfügen. Dies bedeutet, daß diese Dateien sich 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-ID, unter der Net.Data ausgeführt wird, über die folgenden Berechtigungen verfügt:

- Lesen der Net.Data-Initialisierungsdatei `db2www.ini`
- Ausführen von ausführbaren Net.Data-Dateien und -DLL-Dateien sowie Durchsuchen der Verzeichnisse in den Pfaden zu den ausführbaren Dateien und DLL-Dateien
- Lesen der entsprechenden Net.Data-Makrodateien und Durchsuchen der entsprechenden, durch die Pfadkonfigurationsanweisung `MACRO_PATH` angegebenen Verzeichnisse
- Ausführen der entsprechenden Dateien und Durchsuchen der entsprechenden, durch die Pfadkonfigurationsanweisung `EXEC_PATH` angegebenen Verzeichnisse
- Lesen der entsprechenden Dateien und Durchsuchen der entsprechenden, durch die Pfadkonfigurationsanweisung `INCLUDE_PATH` angegebenen Verzeichnisse
- Lesen und Schreiben der entsprechenden Dateien und Durchsuchen der entsprechenden, durch die Pfadkonfigurationsanweisung `FFI_PATH` angegebenen Verzeichnisse
- Lesen der Konfigurationsdatei für Direktverbindungen namens `DTWCN.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

Sie müssen entscheiden, welche Sicherheitsstufe für Ihre Datenbestände angebracht ist. In diesem Kapitel werden Methoden beschrieben, mit denen Sie Ihre Datenbestände sichern können, 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 56
- „Verwenden der Authentifizierung“ auf Seite 56
- „Verwenden der Berechtigung“ auf Seite 56
- „Verwenden von Net.Data-Mechanismen“ auf Seite 57

Außerdem bietet Net.Data Kennwortverschlüsselung für Datenbank-Cliettes. Weitere Informationen hierzu finden Sie im Abschnitt „Konfigurieren von Cliettes“ auf Seite 42.

---

### Verwenden von Firewalls

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

Firewalls führen folgende Funktionen aus:

- 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 des Firewall

Net.Data kann mit einem Firewall-Produkt verwendet werden, das in Ihrer Umgebung ausgeführt wird.

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 einen Firewall konfiguriert haben, mit dem 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 gesicherten Intranet als auch vom öffentlichen Internet abgegrenzt wird. Die Firewall-Software wird für die Erstellung eines ersten Firewall zwischen Web-Server und öffentlichem Internet sowie einem zweiten Firewall zwischen Web-Server und gesichertem Intranet, auf dem sich der DB2-Server befindet, verwendet. Sie finden diese Konfiguration in Abb. 15.

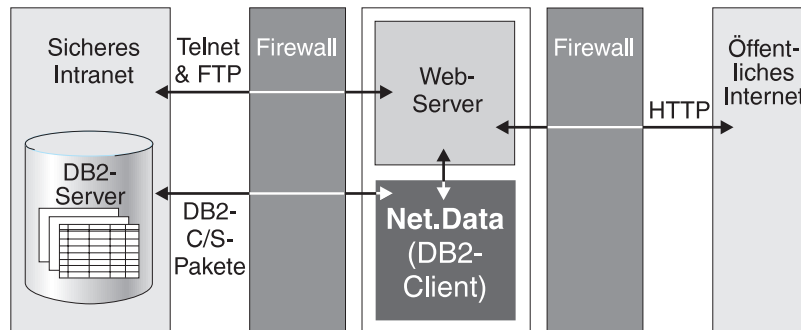


Abbildung 15. Konfiguration mit hoher Sicherheit

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 Intranet auf den DB2-Server zugreifen kann. Gehen Sie hierzu wie folgt vor:
  - Installieren Sie Client Application Enabler (CAE) auf der Web-Server-Maschine.
  - Konfigurieren Sie den Firewall so, daß eine DB2-Übertragung durch den Firewall möglich ist. Eine Möglichkeit besteht darin, Regeln zur Paketfilterung hinzuzufügen, die Anforderungen von DB2-Clients von Net.Data und Bestätigungspakete vom DB2-Server an Net.Data durchlassen.
- Erlauben Sie FTP- und Telnet-Zugriffe zwischen dem Web-Server und dem gesicherten 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 ausgehende TCP-Bestätigungspakete vom Web-Server an jeden beliebigen Host des öffentlichen Internet 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 Datenstations-Plattform. Diese Konfiguration ist einfacher als die oben beschriebene, bietet jedoch trotzdem einen Datenbankschutz. Abb. 16 zeigt diese Konfiguration.

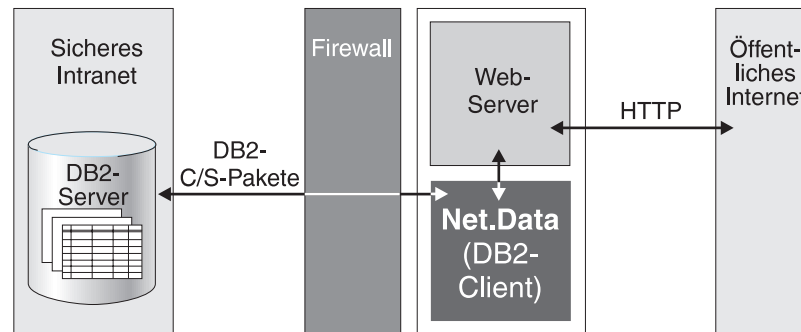


Abbildung 16. Konfiguration mit mittlerer Sicherheit

Hierzu müssen Sie CAE auf dem Web-Server installieren, damit Net.Data Übertragungen zum DB2-Server durchführen kann. Der Firewall muß so konfiguriert sein, daß Anforderungen von DB2-Clients 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 Intranet installiert. Dadurch sind sie nicht gegen externe Manipulation geschützt. Für diese Konfiguration benötigt der Firewall keine Regeln zur Paketfilterung. Abb. 17 zeigt diese Konfiguration.

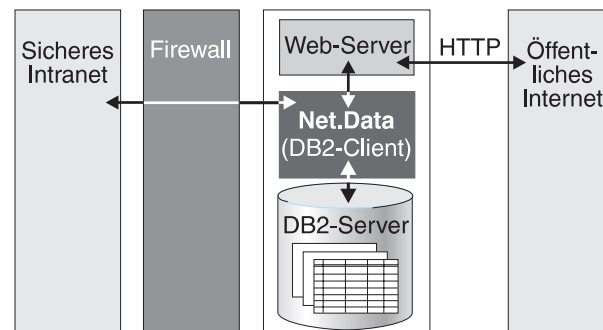


Abbildung 17. Konfiguration mit niedriger Sicherheit

---

## 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 höher und Lotus Domino Go Webserver, 4.6.1 oder höher unterstützen SSL.

---

## Verwenden der Authentifizierung

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.

Net.Data unterstützt zwei Arten von Authentifizierung: 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.

---

## Verwenden der Berechtigung

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 im Abschnitt „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

Außer den oben beschriebenen Methoden können Sie andere von Net.Data bereitgestellte Mechanismen, wie Pfadanweisungen und verdeckte Variablen verwenden, sowie Methoden, die HTML-Formulare oder SQL-Anweisungen verwenden.

Net.Data wertet die Einstellungen der Pfadkonfigurationsanweisungen aus, um die Speicherposition der Dateien und ausführbaren Programme zu ermitteln, die von Net.Data-Makrodateien verwendet werden. Diese Pfadanweisungen geben mindestens ein Verzeichnis an, das Net.Data durchsucht, wenn es versucht, Makrodateien, ausführbare Dateien sowie Kopfdateien zu lokalisieren. Durch die selektive Aufnahme von Verzeichnissen in diese Pfadanweisungen können Sie explizit die Dateien steuern, auf die Benutzer über Browser zugreifen können.

Zusätzliche Informationen zu Pfadanweisungen finden Sie im Abschnitt „Konfigurieren von Net.Data“ auf Seite 5.

Mit verdeckten 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 hierzu finden Sie im Abschnitt „Verdeckte Variablen“ auf Seite 84.

Sie können auch die folgenden Methoden für die Erstellung eines Schutzplans verwenden:

- Erstellen Sie mit Hilfe von Net.Data Ihren eigenen Schutzplan. Beispielsweise könnten Sie anhand eines HTML-Formulars Informationen zur Gültigkeitsprüfung von einem Benutzer anfordern und diese Informationen dann anhand von Daten aus einer Datenbank oder durch ein externes Programm, das von einem Net.Data-Makro aufgerufen wird, überprüfen lassen.
- Schützen Sie Ihre Datenbestände durch die SQL-Anweisungen, die Sie den Benutzern zum Senden an die Datenbank zur Verfügung stellen, begrenzen Sie zum Beispiel SELECT-Anweisungen auf zwei Tabellen.

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

<http://www-genome.wi.mit.edu/WWW/faqs/www-security-faq.html>



---

## Aufrufen von Net.Data

Sie können Net.Data zur Verwendung mit CGI oder FastCGI oder mit einer Web-Server-API wie GWAPI (Lotus Domino Go Webserver), ICAPI (Internet Connection Server), NSAPI (Netscape Server) und ISAPI (Microsoft Internet Server) konfigurieren. Die zum Aufrufen von Net.Data verwendete Syntax hängt von der jeweiligen Net.Data-Konfiguration ab. Informationen zum Konfigurieren von Net.Data finden Sie in „Konfigurieren von Net.Data“ auf Seite 5.

In diesem Kapitel wird das Aufrufen von Net.Data mit CGI erläutert. Unter „Optimieren der Leistung mit den Web-Server-APIs“ auf Seite 157 finden Sie Informationen zum Aufrufen von Net.Data im API-Modus.

Sie können auch angeben, daß Net.Data eine Makrodatei oder eine einzelne SQL-Anweisung, gespeicherte Prozedur oder Funktion ausführen soll. Diese Aufrufarten werden als Makroanforderung bzw. Direktanforderung bezeichnet.

**Makroanforderung** Ruft Net.Data durch Angabe einer Makrodatei auf, die in der Net.Data-Makrosprache geschrieben ist.

**Direktanforderung** Ruft Net.Data durch folgende Angaben auf:

- Der Name einer Sprachumgebung
- Eine SQL-Anweisung oder der Name eines Programms 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 nun eine Direktanforderung an die Datenbank absetzen. Eine Direktanforderung hat keine komplexe Net.Data-Anwendungslogik, die eine Net.Data-Makrodatei 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 auf Seite 60 verdeutlicht die Unterschiede zwischen einer Makroanforderung und einer Direktanforderung. Eine Makroanforderung gibt immer eine Makrodatei innerhalb der URL-Adresse für die Anforderung an und kann auch Formulardaten verwenden. Eine Direktanforderung gibt nie eine Makrodatei innerhalb der URL-Adresse an, kann jedoch weiterhin Formulardaten verwenden.

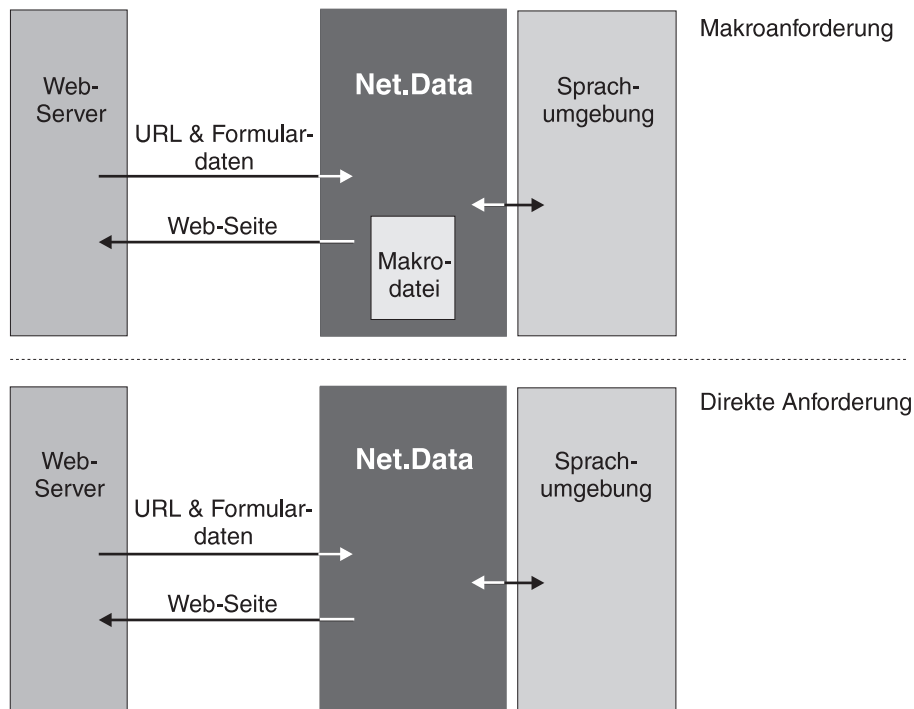


Abbildung 18. Makroanforderung im Vergleich zu Direktanforderung

Die Syntax zum Aufrufen von Net.Data hängt von der Net.Data-Konfiguration und der Art Ihrer Anforderung ab. Bei sowohl Makro- als auch Direktanforderungen kann Net.Data von einem Web-Browser über eine HTML-Programmverbindung (Link), ein HTML-Formular oder eine URL-Adresse aufgerufen werden. Der Web-Server ruft Net.Data über CGI, FastCGI oder eine der Web-Server-APIs auf.

Bei Makroanforderungen gilt folgendes: Der Name der Net.Data-Makrodatei und der Name des auszuführenden HTML-Blocks innerhalb des Net.Data-Makros werden in der Programmverbindung (Link), im Formular bzw. in der URL-Adresse angegeben.

Bei Direktanforderungen werden der Name der Net.Data-Sprachumgebung, die SQL-Anweisung bzw. der Name des Programms und andere zusätzliche erforderlicher Parameterwerte mit Hilfe einer von Net.Data definierten Syntax in der URL-Adresse angegeben.

In diesem Kapitel werden beide Aufrufmethoden beschrieben:

- „Aufrufen von Net.Data mit einer Makrodatei (Makroanforderung)“ auf Seite 61
- „Aufrufen von Net.Data ohne Makrodatei (Direktanforderung)“ auf Seite 63



## Aufrufen von Net.Data mit einer Makrodatei (Makroanforderung)

In diesem Abschnitt wird gezeigt, wie Sie Net.Data durch Angabe einer Makrodatei aufrufen können. Beim Aufruf über eine Makroanforderung können Sie Net.Data mit einer URL-Adresse, einem HTML-Formular bzw. einer HTML-Programmverbindung (Link) aufrufen.

Die folgenden Beispiele zeigen die unterschiedlichen Möglichkeiten zum Aufrufen von Net.Data.

- HTML-Programmverbindung (Link):

```
<A HREF="http://server/cgi-bin/db2www/datname.erw/block/  
[?name=wert&...]">beliebiger text</A>
```

- HTML-Formular:

```
<FORM METHOD=methode ACTION="http://server/cgi-bin/db2www/  
datname/block/[?name=wert&...]">beliebiger  
text</FORM>
```

- URL:

```
http://server/cgi-bin/db2www/datname/block/[?name=wert&...]
```

### Parameter:

<i>server</i>	Gibt den Namen des Web-Servers an. Wenn es sich bei dem Server um den lokalen Server handelt, kann der Server-Name übergangen und ein relativer URL verwendet werden.
<i>datname</i>	Gibt den Namen und die Erweiterung der Net.Data-Makrodatei an, wobei der Dateiname für den relativen Verzeichnispfad in dem von MACRO_PATH angegebenen Verzeichnis steht.
<i>block</i>	Gibt den Namen des HTML-Blocks in der angegebenen Net.Data-Makrodatei an.
<i>methode</i>	Gibt die für das Formular verwendete HTML-Methode an. Hierfür wird METHOD=POST empfohlen.
<b>?name=wert&amp;...</b>	Gibt einen oder mehrere wahlfreie Parameter an, die an Net.Data weitergegeben werden.

## HTML-Programmverbindungen (Links)

Sie können auf einer Web-Seite eine Programmverbindung (Link) erstellen, die zur Ausführung eines HTML-Blocks führt, indem Sie den HTML-Befehl `<a>` in der Makrodatei verwenden. Sie erzielen dies durch Verwendung des Attributs `HREF` zur Angabe des Makros und HTML-Blocks und durch Aufnahme von Text oder sogar eines Bilds in den Programmverbindungsbehl. Dadurch wird der Text bzw. das Bild als „Detailpunkt“ angegeben, wenn die Web-Seite im Browser angezeigt wird. Wenn ein Benutzer über einen Browser den Text bzw. das Bild anklickt, führt Net.Data den HTML-Block im Makro aus.

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>
```

Der Link ruft das folgende Makro auf:

```
%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. Informationen zum Anpassen Ihrer Berichte mit einem REPORT-Block finden Sie in „REPORT-Blöcke“ auf Seite 109.

Im allgemeinen beginnt jeder Block eines Net.Data-Makros mit %block\_name{ und endet mit %}. Weitere Informationen zur Syntax der Net.Data-Makrosprache finden Sie im Handbuch *Net.Data Reference*.

## HTML-Formulare

Sie können die Ausführung Ihrer Net.Data-Makros mit HTML-Formularen dynamisch anpassen. Formulare ermöglichen Benutzern die Eingabe 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 61 auf, indem es Benutzern ermöglicht, in einem Browser mit Hilfe eines einfachen HTML-Formulars die Produktart auszuwählen, für den Informationen angezeigt werden sollen.

```
<H1>Hardware Query Form</H1>
<HR>
<FORM METHOD=POST ACTION="/cgi-bin/db2www/equip1st.d2w/report">
<P>What type of hardware do you want to see?
<MENU>
<LI><INPUT TYPE="RADIO" NAME="hardware" VALUE="MON" checked> Monitors
<LI><INPUT TYPE="RADIO" NAME="hardware" VALUE="PNT"> Pointing devices
<LI><INPUT TYPE="RADIO" NAME="hardware" VALUE="PRT"> Printers
<LI><INPUT TYPE="RADIO" NAME="hardware" 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-Berichtsblock im Makro `equip1st.d2w` aus:

```
%DEFINE DATABASE="MNS97"

%HTML(input){%
%}
  %FUNCTION(DTW_SQL) myQuery(){
SELECT MODNO, COST, DESCRIP FROM EQPTABLE
WHERE TYPE='${hardware}'
  %REPORT{
<H3>Here is the list you requested</H3>
  %ROW{
<HR>
$(N1): $(V1), $(N2): $(V2)
<P>$(N3): $(V3)
%}
%}
%}

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

Im obigen Beispiel wird der Wert für `TYPE=${hardware}` 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 Makrodatei (Direktanforderung)

In diesem Abschnitt wird gezeigt, wie Sie Net.Data ohne Angabe einer Makrodatei aufrufen können. Beim Aufruf mit Direktanforderungen müssen Sie den Namen der Net.Data-Sprachumgebung, die SQL-Anweisung bzw. ein auszuführendes Programm und andere zusätzlich erforderliche Parameterwerte mit Hilfe einer von Net.Data definierten Syntax in der URL-Adresse angeben.

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 keine Makrodatei zu lesen und zu verarbeiten braucht. Die von Net.Data bereitgestellten Sprachumgebungen SQL, ODBC, Oracle, Sybase, Java, System, Perl und REXX unterstützen Direktanforderungsaufrufe. Sie können Net.Data mit einer URL-Adresse, einem HTML-Formular bzw. einer Programmverbindung (Link) aufrufen.

Eine Direktanforderung ruft Net.Data durch Übergeben der Parameter im Paarabschnitt (`NAME=VALUE`) 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?direktanforderung">beliebiger text</A>
```

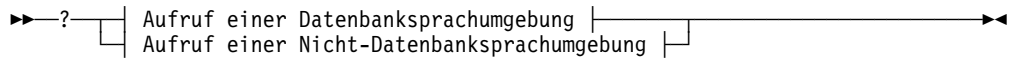
Dabei steht *direktanforderung* für die Direktanforderungssyntax. Die folgende HTML-Programmverbindung (Link) enthält zum Beispiel eine Direktanforderung:

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

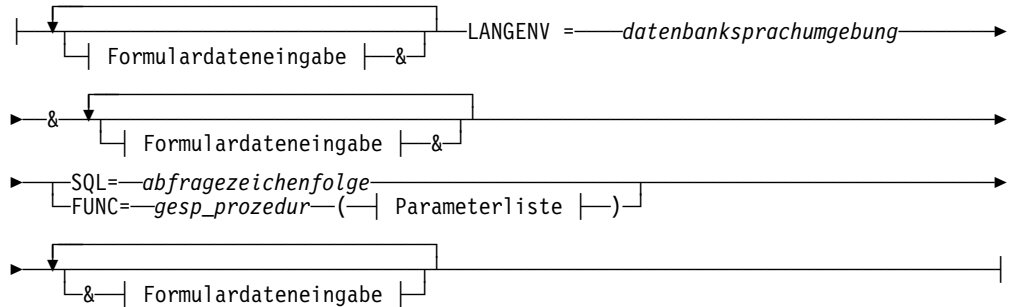
## Syntax für Direktanforderungen

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

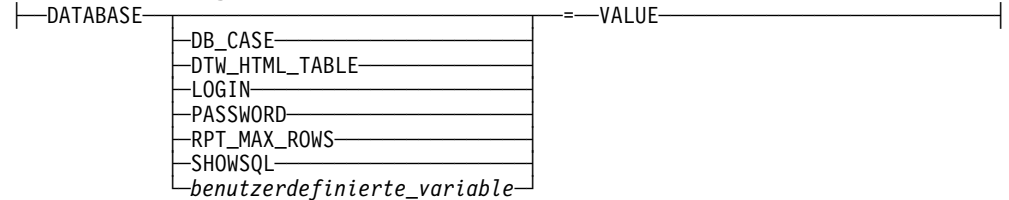
### Syntax



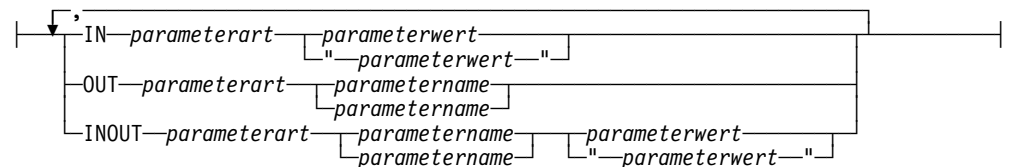
### Aufruf einer Datenbanksprachumgebung:



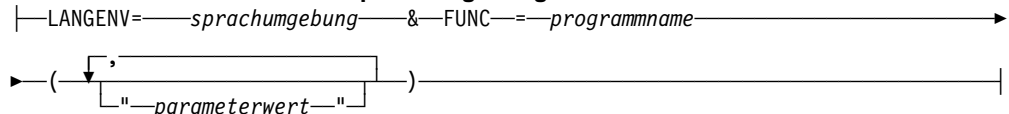
### Formulardateneingabe:



### Parameterliste:



### Aufruf einer Nicht-Datenbanksprachumgebung:



### Parameter

#### Aufruf einer Datenbanksprachumgebung

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

#### Formulardateneingabe

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 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.

*benutzerdefinierte\_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.

*datenbanksprachumgebung*

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.

*abfragezeichenfolge*

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.

### *gesp\_prozedur*

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

### *parameterart*

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

### *parametername*

Gibt einen gültigen Parameternamen an.

### *parameterwert*

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**

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

## **LANGENV**

Gibt die Zielsprachumgebung für die Ausführung der Funktion an.

### *sprachumgebung*

Gibt den Namen der Nicht-Datenbanksprachumgebung an:

- DTW\_PERL
- DTW\_REXX
- DTW\_SYSTEM

## **FUNC**

Gibt an, daß die Direktanforderung die Ausführung eines Programms angibt.

### *programmname*

Gibt das Programm mit der auszuführenden Funktion an.

### *parameterwert*

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)

**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)">
beliebiger Text</A>
```

**Beispiel 2:** Eine Programmverbindung (Link), die die Perl-Sprachumgebung wie im vorherigen Beispiel aufruft, jedoch eine Zeichenfolge mit URL-verschlüsselten 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)">beliebiger Text</A>
```

**Hinweis:** Sie müssen bestimmte Zeichen, wie Leerzeichen und doppelte Anführungszeichen, innerhalb von URL-Adressen verschlüsseln. In diesem Beispiel müssen die doppelten Anführungszeichen und Leerzeichen innerhalb des Parameterwerts als %22 bzw. das Pluszeichen (+) verschlüsselt werden. Eine Liste aller Zeichen, die innerhalb von URL-Adressen verschlüsselt werden müssen, finden Sie in der Beschreibung zur Funktion DTW\_URLESCSEQ im Handbuch *Net.Data Reference*.

### HTML-Formulare

**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>
```

**Hinweis:** Sie können Variablensubstitutionen in Direktanforderungsaufrufen verwenden, indem Sie ein durch ein Net.Data-Makro erstelltes HTML-Formular einsetzen.

## URL-Adresse

**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
```

**Hinweis:** Stellen Sie zum Verweisen auf einen Dateinamen mit Pfad, der nicht in der Pfadkonfigurationsanweisung EXEC angegeben ist, den vollständig qualifizierten Pfad zusammen mit dem Dateinamen für den Wert *funktionsname* bereit.

**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
```



## 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. in der Makrodatei definiert sind
- Formatieren der Verarbeitungsausgabe in HTML und Rückgabe an den Web-Browser

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

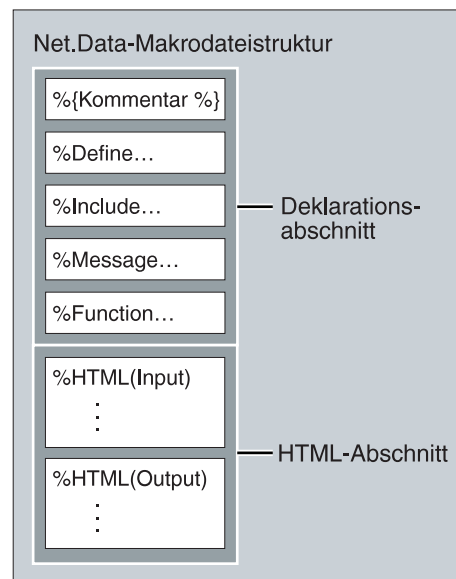


Abbildung 19. Makrodateistruktur

- Der *Deklarationsabschnitt* enthält die Definitionen von Variablen und Funktionen in der Makrodatei.
- Der *HTML-Abschnitt* enthält HTML-Blöcke, die aus HTML-Anweisungen bestehen, die das Layout der Web-Seite festlegen.

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

**Hinweis zu Berechtigungen:** Stellen Sie sicher, daß der Web-Server Zugriffsrechte auf diese Datei hat. Weitere Informationen hierzu finden Sie im Abschnitt „Angaben von Zugriffsrechten auf Net.Data-Dateien“ auf Seite 52.

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

- „Aufbau einer Net.Data-Makrodatei“
- „Net.Data-Makrovariablen“ auf Seite 76
- „Net.Data-Funktionen“ auf Seite 90
- „Generieren von HTML in einem Makro“ auf Seite 107
- „Bedingte Logik und Schleifen in einer Makrodatei“ auf Seite 111
- „Verwenden großer Objekte“ auf Seite 115

---

## Aufbau einer Net.Data-Makrodatei

Die Makrodatei besteht aus zwei Abschnitten:

- Der Deklarationsabschnitt, der die im HTML-Abschnitt 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 auf Dateien hat, auf die in EXEC- und INCLUDE-Anweisungen verwiesen wird. Weitere Informationen hierzu finden Sie im Abschnitt „Angaben von Zugriffsrechten auf Net.Data-Dateien“ auf Seite 52.

- Der HTML-Abschnitt 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 der Makrodatei an. Die HTML-Blöcke werden im Abschnitt „HTML-Blöcke“ auf Seite 74 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.COMD, das die vom Benutzer eingegebenen Daten wiederholt. 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 gängige HTML-Befehle, 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 oder einer Web-Server-API darauf zu. Net.Data erfordert zwei Parameter: den Namen des zu verarbeitenden Makros und den anzuzeigenden HTML-Block in diesem Makro.

Wenn die Makrodatei 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 in der Makrodatei 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, die einen Funktionsaufruf an ein externes REXX-Programm und einen Funktionsaufruf an eine in der Makrodatei enthaltene Funktion definieren.

```
%{ ***** FUNCTION Block *****%}
%FUNCTION(DTW_REXX) rexx1 (IN input) returns(result) { <-- Diese Funktion
                                                         akzeptiert einen
                                                         Parameter und gibt
                                                         ein Ergebnis zurück,
                                                         das den zugeordneten
                                                         Funktionsaufruf
                                                         ersetzt.

                                                         %EXEC{ompsamp.cmd %} <-- Diese Funktion führt das externe REXX-Programm
                                                         "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 eine 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 liefert ebenfalls eine Variable namens result. Der Inhalt der Variablen result im REXX-Befehl tritt an die Stelle der im OUTPUT-Block enthaltenen aufrufenden Funktion @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 wiederholt die an sie übergebenen Daten. Sie können den Ergebnistext nach Belieben formatieren, indem Sie den anfordernden Funktionsaufruf @rexx1() zwischen normale HTML-Befehle (wie z. B. <b> oder <em>) 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 Block, today, verweist ebenfalls auf ein REXX-Programm. In diesem Fall befindet sich das gesamte REXX-Programm (eine ganze Zeile) selbst innerhalb der Funktionsdeklaration. Ein externes Programm ist nicht erforderlich. Inline-Programme sind für REXX- und Perl-Funktionen zulässig, da sie interpretierte 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 dienen zur Definition des Web-Seiten-Layouts, zur Angabe von Variablen und zum Aufrufen von Funktionen mit Hilfe von HTML-Blöcken, die als Eingangs- und Endpunkte für das Makro verwendet werden.

In der Net.Data-Aufrufsanforderung wird immer ein HTML-Block angegeben, und jedes Makro muß mindestens einen HTML-Block enthalten.

Der erste HTML-Block in der Beispielmakrodatei heißt INPUT. Der INPUT-Block 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 TYPE="text" SIZE="30">
                                von anderer Stelle des Makros
                                verwiesen werden. Sie wird mit der
                                Benutzereingabe initialisiert.

<p>
<INPUT TYPE="submit" VALUE="Enter">

<hr>
<p>
[
<a href="/">Home page</a>]
</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. Sie können einen beliebigen Namen vergeben. 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, die im folgenden noch näher beschrieben wird. 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 dieses Feld eingegebenen Daten in der Variablen *input\_data* an den HTML-Block OUTPUT übergeben.

Sie können über einen relativen Verweis auf HTML-Blöcke in anderen Makrodateien zugreifen, wenn sich diese Makrodateien auf demselben Web-Server befinden. So greift beispielsweise der ACTION-Parameter ACTION="../othermacro.d2w/main" auf den HTML-Block main in der Makrodatei othermacro.d2w zu. Auch hier werden alle in das Formular eingegebenen Daten in der Variablen *input\_data* an das andere 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>
```

Zum Empfang von Eingabedaten müssen keine Umgebungsvariablen definiert werden, wie es bei den meisten CGI-Programmen der Fall ist. Net.Data verarbeitet Umgebungsvariablen für Sie. Sie müssen die Variablennamen lediglich angeben. Der nächste HTML-Block des Beispiels ist der Block OUTPUT. Er enthält HTML-Code und Net.Data-Makroanweisungen, die die Verarbeitung der Ausgabe von der INPUT-Blockanforderung 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, der das Argument "input_data" übergibt.
<p>
<hr>
<p>
[
<a href="/">Home page</a> '
<a href="input">Previous page</a>]
%}
```

Ebenso wie der INPUT-Block besteht 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 legt die Anzahl und die Arten der übergebenen Variablen fest.

---

## 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 zum ersten Mal auf sie verwiesen wird
  - Parametervariablen, die zu einer Definition im FUNCTION-Block gehören und die nur innerhalb eines FUNCTION-Blocks verwendet werden können
  - Variablen, die von Net.Data verfügbar gemacht werden und Formulardaten bzw. jeweils aus Name und Wert bestehenden URL-Datenpaaren entsprechen
  - Variablen, die einer Net.Data-Tabelle zugeordnet sind und auf die nur innerhalb eines ROW-Blocks oder eines REPORT-Blocks verwiesen werden kann

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 der Makrodatei aus verwiesen werden kann. Kennungen mit globalem Geltungsbereich sind:

  - Integrierte Net.Data-Funktionen
  - Formulardaten
  - URL-Daten
  - Variablen, die innerhalb eines HTML-Blocks verfügbar gemacht werden
- Makrodatei

Eine Kennung hat diesen Geltungsbereich, 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.) Eine Kennung mit dem Geltungsbereich Makrodatei ist von der Stelle, an der sie deklariert wird, bis zum Ende der Makrodatei sichtbar.



- FUNCTION-Block bzw. MACRO\_FUNCTION-Block

Eine Kennung besitzt den Geltungsbereich FUNCTION-Block, wenn sie in der Parameterliste der Funktionsdefinition deklariert wird. Wenn eine Kennung mit dem gleichen Namen bereits außerhalb der Funktionsdefinition vorhanden ist, verwendet Net.Data die Kennung aus der Funktionsparameterliste innerhalb des Funktionsblocks. Net.Data verwendet oder ändert die Kennung bzw. ihren Wert von außerhalb des Funktionsblocks nicht.

Eine Kennung besitzt nicht den Geltungsbereich FUNCTION-Block, wenn sie außerhalb der Funktion deklariert oder initialisiert wurde und nicht in der Funktionsparameterliste deklariert wird. Wenn die Kennung innerhalb eines Funktionsblocks verwendet wird, behält sie den Wert bei, der ihr vor dem Funktionsaufruf zugeordnet wurde. Wird sie innerhalb des Funktionsblocks aktualisiert, behält die Kennung den neuen Wert nach dem Funktionsaufruf bei.

Eine Kennung besitzt den Geltungsbereich FUNCTION-Block, wenn sie weder in der Parameterliste deklariert noch vor dem Funktionsaufruf deklariert oder initialisiert wird. Auf sie kann außerhalb des Funktionsblocks nicht verwiesen werden.

- REPORT-Block

Eine Kennung besitzt den Geltungsbereich REPORT-Block, wenn auf sie nur innerhalb eines REPORT-Blocks verwiesen werden kann (zum Beispiel Tabellenspaltennamen N1, N2, ..., Nn). Nur solche Variablen, die Net.Data implizit als Teil der Tabellenverarbeitung definiert, können den Geltungsbereich REPORT-Block haben. Alle anderen verfügbar gemachten Variablen besitzen den Geltungsbereich FUNCTION-Block.

- ROW-Block

Eine Kennung besitzt den Geltungsbereich ROW-Block, wenn auf sie nur innerhalb eines ROW-Blocks verwiesen werden kann (zum Beispiel Tabellenwertnamen V1, V2, ..., Vn). Nur solche Variablen, die Net.Data implizit als Teil der Tabellenverarbeitung definiert, können den Geltungsbereich ROW-Block haben. Alle anderen verfügbar gemachten Variablen besitzen den Geltungsbereich FUNCTION-Block.

Wenn auf eine Variable verwiesen wird, wird sie durch den Wert der Kennung ersetzt. Wenn ein Verweis auf eine Variable keinen zugeordneten Wert besitzt oder ein Funktionsaufruf keinen Wert zurückgibt, wird der Verweis durch eine leere Zeichenfolge ersetzt.

Die folgenden Abschnitte beschreiben die Definition von Variablen und die Methoden, auf sie zu verweisen, und erläutern die verschiedenen Arten von Variablen und ihre Verwendung.

## Definieren von Variablen

Variablen können in einem Net.Data-Makro auf drei Arten definiert werden:

- **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 der Anweisung ist für Net.Data spezifisch:

```
%DEFINE variable_name="variable value"
```

```
%DEFINE variable_name={ variable value on multiple  
                        lines of text %}
```

*variable\_name* ist der Name, den Sie der Variablen geben. Variablen müssen mit einem Buchstaben oder Unterstreichungszeichen beginnen und können jedes beliebige alphanumerische Zeichen oder ein Unterstreichungszeichen enthalten. Alle Variablennamen sind von der Groß-/Kleinschreibung abhängig, mit Ausnahme von *N\_columnName* und *V\_columnName*, bei denen es sich um Tabellenvariablen handelt.

Sollen Anführungszeichen in einer Zeichenfolge verwendet werden, setzen Sie jeweils zwei Anführungszeichen hintereinander. Zwei aufeinanderfolgende Anführungszeichen allein entsprechen einer leeren Zeichenfolge. Beispiel:

```
%DEFINE HI="say ""hello"""
```

Die Variable HI enthält den Wert say "hello".

```
%DEFINE reply="hello"
```

Die Variable reply enthält den Wert hello.

```
%DEFINE empty=""
```

Die Variable empty ist Null.

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 und INPUT**

Sie können die Befehle SELECT und INPUT in einem HTML-Formular verwenden. Im folgenden Beispiel werden Standardbefehle für HTML-Formulare zum Definieren einer Variablen verwendet:

```
<INPUT NAME="variable_name" TYPE=...>
```

oder

```
<SELECT NAME="variable_name">
```

*variable\_name* ist der Name, den Sie der Variablen geben, und der Wert der Variablen wird durch die im Formular empfangene Eingabe bestimmt. Im

Abschnitt „HTML-Formulare“ auf Seite 62 finden Sie ein Beispiel dafür, wie diese Art der Variablendefinition in einem Net.Data-Makro verwendet wird.

Ein Variablenwert, der durch einen Befehl INPUT oder SELECT empfangen wird, überschreibt einen mit einer Anweisung DEFINE in einem Net.Data-Makro definierten Variablenwert.

- **URL-Daten**

Sie können Net.Data-Makros über URL-Anforderungen aufrufen und Variablen, wie zum Beispiel eine Benutzer-ID, in die URL-Angabe einfügen, die an Net.Data gesendet werden sollen. 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 den Befehl INPUT empfängt. Net.Data empfängt und verarbeitet die Daten ebenso 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.

Wenn Sie Variablen als Teil Ihres HTML-Texts verwenden wollen, fügen Sie Verweise auf Variablen in Ihre HTML-Blocks ein. 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 Verbindung (Link) erstellen:

```
<A href="$(homeURL)">Home page</A>
```

Verweise auf Variablen können in allen Abschnitten eines Net.Data-Makros eingefügt werden. Wenn die Variable zu dem Zeitpunkt, zu dem auf sie verwiesen wird, noch nicht definiert ist, gibt Net.Data eine leere Zeichenfolge zurück. Net.Data definiert die Variable nicht.

**Einschränkung:** Rückbezügliche Verweise (bzw. Zyklen) sind nicht zulässig. Die folgenden Anweisungen DEFINE verursachen beispielsweise einen Fehler, wenn auf die Variable verwiesen wird und die endgültigen Werte berechnet werden:

```
%DEFINE a="$(b)"  
%DEFINE b="$(a)"
```

## Variablenarten

Sie können die folgenden Arten von Variablenverweisen in Ihren Makrodateien verwenden:

- „Bedingungsvariablen“
- „Umgebungsvariablen“ auf Seite 81
- „Ausführbare Variablen“ auf Seite 81
- „Verdeckte Variablen“ auf Seite 84
- „Listenvariablen“ auf Seite 85
- „Tabellenvariablen“ auf Seite 86
- „Zusätzliche Variablen“ auf Seite 87
- „Variablen zur Tabellenverarbeitung“ auf Seite 88
- „Berichtsvariablen“ auf Seite 89
- „Sprachumgebungsvariablen“ auf Seite 89

Wenn Sie Variablen, die in spezieller Weise durch Net.Data definiert sind (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.

### 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
```

Im Abschnitt „Listenvariablen“ auf Seite 85 finden Sie ein Beispiel für die Verwendung von Bedingungsvariablen mit Listenvariablen.

## Umgebungsvariablen

Sie können auf Net.Data-Umgebungsvariablen verweisen, die in dem Prozeß existieren, unter dem Net.Data ausgeführt wird.

Die Syntax zur Definition von Umgebungsvariablen sieht folgendermaßen aus:

```
%define var=%ENVVAR
```

Dabei ist *var* der Name der Variable, 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
```

Im Handbuch *Net.Data Reference* finden Sie weitere Informationen zur Anweisung `ENVVAR`.

## 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 zu Sprachkonstrukten 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, liefert `Net.Data` den Wert:

```
Today is Tue 11-07-1999
```

Wenn `Net.Data` eine ausführbare Variable in einer Makrodatei 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 hierzu finden Sie unter „`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 (Library List) 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 sie 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 HTML-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 Ihre HTML-Quelle 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 `$$(variable)` wird verwiesen, anschließend werden sie definiert.
2. Verwenden Sie in dem HTML-Block, in dem auf die Variablen verwiesen wird, für einen Variablenverweis zwei Dollarzeichen anstelle eines einzelnen Dollarzeichens. Zum Beispiel `$$(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 Klauseln WHERE oder HAVING 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 Werttrennzeichen ein. Die meisten Abfragen verwenden boolesche oder mathematische Operatoren (z. B. AND, OR oder >) als Werttrennzeichen. 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="$(cond1)">Sao Paola<BR>  
<INPUT TYPE="checkbox" NAME="conditions" VALUE="$(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 Feldgruppe identischer Datensätze oder Zeilen sowie eine Feldgruppe von Spaltennamen, die die Felder in jeder Zeile beschreiben. Eine Tabelle wird in einem Net.Data-Makro wie in der folgenden Anweisung definiert:

```
%DEFINE myTable=%TABLE(30)
```

Die Zahl hinter TABLE gibt die maximale Anzahl der Zeilen an, die diese Tabelle enthalten kann. Wenn Sie eine Tabelle ohne Zeilenbegrenzung angeben wollen, müssen Sie den Standardwert bzw. ALL angeben:

```
%DEFINE myTable2=%TABLE  
%DEFINE myTable3=%TABLE(ALL)
```

Bei der Definition einer Tabelle hat die Tabelle null Zeilen und null Spalten, und es wird kein Speicher zugeordnet. Die einzige Möglichkeit, eine Tabelle mit Werten zu füllen, besteht darin, sie als Parameter OUT oder INOUT an eine Funktion zu übergeben. Die Sprachumgebung DTW\_SQL fügt die Ergebnisse einer Anweisung SELECT automatisch in eine Tabelle ein.

Bei allen anderen Sprachumgebungen, wie zum Beispiel DTW\_REXX oder DTW\_PERL, werden Tabellenwerte nicht automatisch gesetzt. Statt dessen werden die einzelnen Elemente der Tabelle für den systemeigenen Sprach-Interpreter als Ausgabeparameter zur Verfügung gestellt und müssen dann durch die Prozedur bzw. das Programm, das ausgeführt wird, gesetzt werden. Einzelheiten hierzu finden Sie in den Beschreibungen der Sprachumgebungen im Handbuch *Language Environment Reference*.

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 verwiesen werden. Nähere Informationen hierzu finden Sie unter „Variablen zur Tabellenverarbeitung“ auf Seite 88. 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.

Die Spaltennamen und Feldwerte in einer Tabelle werden als Feldgruppenelemente mit dem Ursprung 1 und nicht wie der Standardkonvention der Sprachen C und C++ entsprechend mit dem Ursprung 0 für den Anfang von Feldgruppen 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 über die Ergebnismenge einer Datenbankabfrage
- Bestimmen von Informationen zu Dateiadressen und Datumsangaben

Zusätzliche Variablen können entweder einen von Net.Data bestimmten vordefinierten Wert 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 der Makrodatei als Variablenverweise verwendet und liefern Informationen über den aktuellen Status von Dateien, Datumsangaben oder den Status eines Funktionsaufrufs. Um beispielsweise den Namen der aktuellen Datei abzurufen, könnten Sie folgende Variable verwenden:

```
<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 der Makrodatei 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"
```

Eine Liste verschiedener gültiger Variablen mit Syntax und Beispielen finden Sie im Kapitel über Variablen im Handbuch *Net.Data Reference*.

## 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, und ermöglichen Ihnen, auf Werte aus den Ergebnismengen von SQL-Abfragen oder Funktionsaufrufen nach Spalte, Zeile oder Feld, die bzw. das verarbeitet wird, zu verweisen. Sie können außerdem 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 HTML-Ausgabe auf den aktuellen Spaltennamen verweisen.

Variablen zur Tabellenverarbeitung werden als Variablenverweise innerhalb der Makrodatei verwendet. Zum Beispiel können Sie den Namen der aktuellen Spalte, die verarbeitet wird, folgendermaßen abrufen:

```
<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)
```

Eine Liste gültiger Variablen zur Tabellenverarbeitung mit Syntax und Beispielen finden Sie im Kapitel über Variablen im Handbuch *Net.Data Reference*.

## Berichtsvariablen

Net.Data zeigt HTML-Ausgaben, die durch die Makrodatei 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 HTML-Ausgabe und bei der Verwendung der HTML-Ausgabe mit Standardberichten und Net.Data-Tabellen. Diese Variablen müssen vor ihrer Verwendung mit Hilfe einer Anweisung `DEFINE` 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 Variable `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)
%}
```

Eine Liste gültiger Berichtsvariablen mit Syntax und Beispielen finden Sie im Kapitel über Variablen im Handbuch *Net.Data Reference*.

## Sprachumgebungsvariablen

Diese Variablen werden mit Sprachumgebungen verwendet und beeinflussen die Verarbeitung einer Anforderung durch die Sprachumgebung. Diese Variablen müssen vor ihrer Verwendung in einem Variablenverweis mit Hilfe einer Anweisung `DEFINE` oder der Funktion `@DTW_ASSIGN()` definiert werden. Setzen Sie Sprachumgebungsvariablen, oder verweisen Sie auf sie in den entsprechenden Net.Data-Makroblöcken.

Mit diesen Variablen können Sie Operationen durchführen wie das Herstellen von Verbindungen zu Datenbanken, das Bereitstellen alternativen Texts für Java-Applets und das Aktivieren der Sprachenunterstützung.

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, wie die Datenbank definiert wird, auf die zugegriffen werden soll.

```
%DEFINE DATABASE="CELDIAL"
```

Eine Liste gültiger Sprachumgebungsvariablen mit Syntax und Beispielen finden Sie im Kapitel über Variablen im Handbuch *Net.Data Reference*.

## Net.Data-Funktionen

Net.Data stellt eine Vielzahl nützlicher Funktionen für Ihre Web-Anwendungen zur Verfügung. Außerdem lassen sich problemlos auch eigene Funktionen schreiben. Abb. 20 zeigt, wie die Net.Data-Funktionen und die Makrodatei miteinander interagieren.

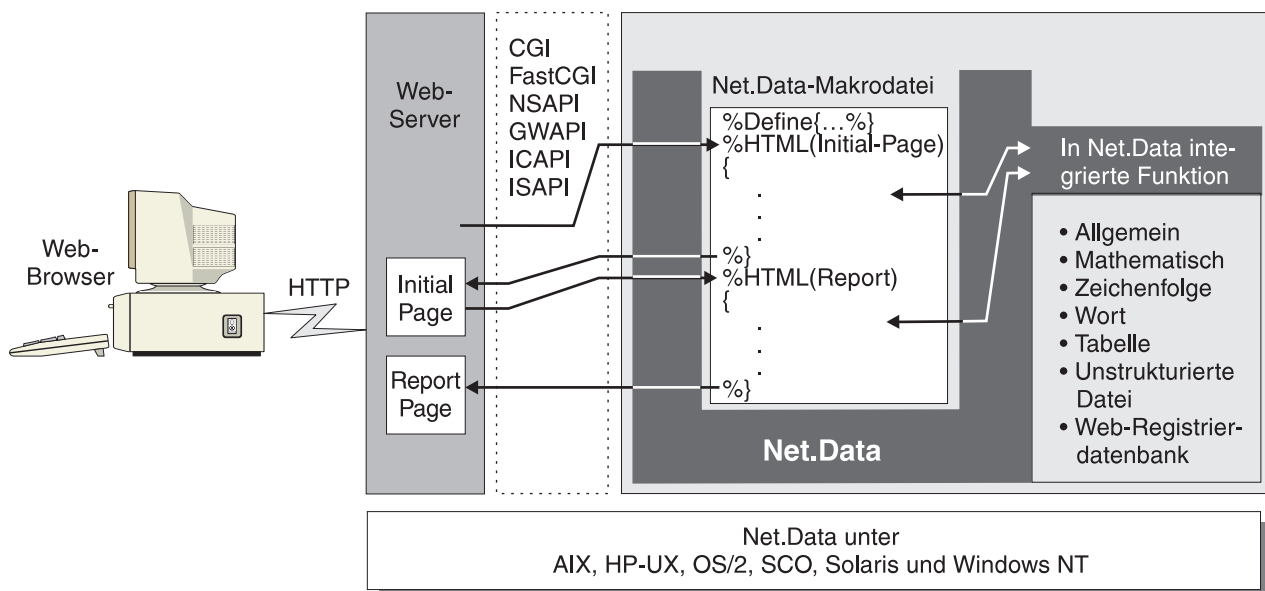


Abbildung 20. In Net.Data integrierte Funktionen

## Definieren von Funktionen

Sie können eigene Funktionen definieren oder die in Net.Data integrierten Funktionen verwenden. Zum Definieren eigener Funktionen für die Makrodatei (die auch als benutzerdefinierte Funktionen bezeichnet werden), verwenden Sie einen FUNCTION-Block oder MACRO\_FUNCTION-Block.

**FUNCTION-Block** Definiert eine Unterroutine, die von einem Net.Data-Makro aufgerufen wird und entweder von der Sprachumgebung verarbeitet wird oder ein externes Programm aufruft.

**MACRO\_FUNCTION-Block** Definiert eine Unterroutine, die von einem Net.Data-Makro aufgerufen wird und von Net.Data, d. h. nicht von einer anderen Sprachumgebung, verarbeitet werden muß. Die Anweisungen im Block müssen Quellenanweisungen der Net.Data-Makrosprache sein.

Der MACRO\_FUNCTION-Block ist eine Alternative zum FUNCTION-Block, die die Leistung verbessern kann. Der MACRO\_FUNCTION-Block wird nur von Net.Data verarbeitet und ruft keine Sprachumgebung auf. Einzelheiten zu diesen beiden Konstrukten finden Sie im Handbuch *Net.Data Reference*.

Die Syntax sieht wie folgt aus:

### FUNCTION-Block:

```
%FUNCTION(art) funktionsname(syntax parameter, ...)
[RETURNS(rückkehrvariable)] {
    ausführbare-anweisungen
    report-block
    message-block
%}
```

### MACRO\_FUNCTION-Block:

```
%MACRO_FUNCTION funktionsname(syntax parameter, ...) {
    ausführbare-anweisungen
%}
```

Dabei gilt folgendes:

*art*

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. Mit Net.Data werden verschiedene Standardsprachumgebungen bereitgestellt.

*funktionsname*

Gibt den Namen des FUNCTION- oder MACRO\_FUNCTION-Blocks an. Sie können den FUNCTION-Block oder MACRO\_FUNCTION-Block über einen Funktionsaufruf an einer anderen Stelle in der Makrodatei ausführen. Der Funktionsaufruf verweist auf den *funktionsnamen* mit einem vorangestellten kommerziellen A (@). Nähere Informationen finden Sie unter „Aufrufen von Funktionen“ auf Seite 94.

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 eine Funktion aufruft, werden alle FUNCTION-Blöcke gleichen Namens bzw. MACRO\_FUNCTION-Blöcke gleichen Namens 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 einen FUNCTION-Block bzw. einen MACRO\_FUNCTION-Block übergeben, von ihm empfangen oder sowohl übergeben als auch empfangen wird. Die Syntaxart gilt für alle nachfolgenden Parameter in der Parameterliste, bis sie durch eine andere Syntaxart geändert wird. Die Standardart ist IN.

*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 des FUNCTION-Blocks bzw. MACRO\_FUNCTION-Blocks ungültig.

*rückkehrvariable* Geben sie diesen Parameter nach dem Schlüsselwort RETURNS an, um einen speziellen OUT-Parameter zu definieren. Der Wert der Rückkehrvariablen wird dem Funktionsaufruf zugeordnet und ersetzt den Funktionsaufruf bei der Net.Data-Makroverarbeitung. Wenn Sie die Klausel RETURNS nicht angeben, ist der Wert des Funktionsaufrufs einer der folgenden Werte:

- NULL, falls der Rückkehrcode aus dem Aufruf an die Sprachumgebung Null ist
- Der Wert des Rückkehrcodes, wenn der Rückkehrcode ungleich Null ist

*ausführbare-anweisungen* Die Gruppe der Sprachanweisungen, die an die angegebene Sprachumgebung zur Verarbeitung übergeben wird, nachdem die Variablen ersetzt und die Funktionen verarbeitet wurden. *ausführbare-anweisungen* können Net.Data-Variablenverweise und Net.Data-Funktionsaufrufe enthalten. Vor der Übergabe der ausführbaren Anweisungen an die Sprachumgebung ersetzt Net.Data diese Variablenverweise bzw. Funktionsaufrufe durch die tatsächlichen Werte. *ausführbare-anweisungen* 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 im Abschnitt „Ausführbare Variablen“ auf Seite 81.

Bei MACRO\_FUNCTION-Blöcken sind die ausführbaren Anweisungen eine Kombination aus HTML-Anweisungen und Net.Data-Makrosprachkonstrukten. In diesem Fall spielt die Sprachumgebung keine Rolle, da Net.Data als Sprachprozessor fungiert und die ausführbaren Anweisungen auswertet und ausführt.

*report-block* Definiert den REPORT-Block zur Behandlung der Ausgabe des FUNCTION-Blocks. Siehe Abschnitt „REPORT-Blöcke“ auf Seite 109.

*message-block* Definiert den MESSAGE-Block, der alle vom FUNCTION-Block zurückgemeldeten Nachrichten verarbeitet. Siehe Abschnitt „MESSAGE-Blöcke“ auf Seite 105.

Definieren Sie Funktionen in der äußersten Net.Data-Makroebene, bevor diese im Net.Data-Makro aufgerufen werden.

## Verwenden von Sonderzeichen in Sprachanweisungen

Wenn Zeichen, die der Syntax der Net.Data-Sprachkonstrukte entsprechen, im Abschnitt für die Sprachanweisungen eines FUNCTION-Blocks als Teil eines syntaktisch gültigen, eingebetteten Programmcodes (wie für REXX oder Perl) verwendet werden, können sie als Net.Data-Sprachkonstrukte fehlinterpretiert 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 enthält als Teil ihrer Perl-Sprachanweisungen zum Beispiel 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:

- o Mit der Anweisung %EXEC:

```
%function(DTW_PERL) func() {  
    %EXEC{ func.pr1 %}  
%}
```

- o Mit einem Variablenverweis zur Angabe der Zeichen %{:

```
%define percent_openbrace = "%{"
```

```
%function(DTW_PERL) func() {  
    ...  
    for $num_words (sort bynumber keys $(percent_openbrace) $Rtitles{$num} )) {  
        &make_links($Rtitles{$num}){$num_words});  
    }  
    ...  
%}
```

## Aufrufen von Funktionen

Eine Funktion wird von einem Net.Data-Makro mit Hilfe des kommerziellen A (@) gefolgt von einem FUNCTION-Blocknamen bzw. einem MACRO\_FUNCTION-Blocknamen aufgerufen:

```
@funktionsname([ argument,... ])
```

*funktionsname*

Dies ist der Name des FUNCTION-Blocks

bzw. MACRO\_FUNCTION-Blocks, der aufgerufen werden soll. 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 definierten Variable, eine Literalzeichenfolge, ein Variablenverweis oder ein Funktionsaufruf. Die Argumente in einem Funktionsaufruf werden mit den Parametern eines FUNCTION-Blocks bzw. MACRO\_FUNCTION-Blocks abgeglichen, und jedem Parameter wird bei der Verarbeitung des FUNCTION-Blocks bzw. des MACRO\_FUNCTION-Blocks der Wert des entsprechenden Arguments zugewiesen. Die Argumente müssen dieselbe Anzahl und Art wie die zugehörigen Parameter aufweisen.

Net.Data verarbeitet den FUNCTION-Block, MACRO\_FUNCTION-Block oder integrierte Funktionen, die einem Funktionsaufruf zugeordnet sind, in der folgenden Reihenfolge:

1. Net.Data verarbeitet die Variablenverweise und Funktionsaufrufe im Abschnitt mit den ausführbaren Anweisungen des FUNCTION-Blocks. Net.Data ersetzt alle Variablenverweise durch die aktuellen Werte der Variablen und führt alle Funktionsaufrufe aus und ersetzt sie durch den Rückkehrwert des jeweiligen Funktionsaufrufs. Die Variablenverweise und Funktionsaufrufe werden in der Reihenfolge verarbeitet, in der sie angegeben sind. Net.Data verarbeitet während dieses Schritts keine integrierten Funktionen oder MACRO\_FUNCTION-Blöcke.
2. Der systemeigene Sprachprozessor verarbeitet den Abschnitt mit den ausführbaren Anweisungen. Bei FUNCTION-Blöcken entspricht der Prozessor der für den FUNCTION-Block definierten Sprachumgebung wie zum Beispiel SQL, REXX oder Perl. Bei MACRO\_FUNCTION-Blöcken fungiert Net.Data als Sprachprozessor und führt die ausführbaren Anweisungen aus. Integrierte Funktionen enthalten keine ausführbaren Anweisungen. Net.Data verarbeitet die integrierten Funktionen nach Funktionsnamen.  
  
Net.Data übergibt die Parameter der Funktion an den systemeigenen Sprachprozessor. Net.Data übergibt Werte an den systemeigenen Sprachprozessor nur für IN- und INOUT-Parameter und akzeptiert vom systemeigenen Sprachprozessor zurückgegebene Werte nur für OUT- und INOUT-Parameter.
3. Net.Data setzt die impliziten Variablen RETURN\_CODE und DTW\_DEFAULT\_MESSAGE entsprechend dem Rückkehrcode und der zurückgegebenen Nachricht des Sprachprozessors. Für MACRO\_FUNCTION-Blöcke setzt Net.Data diese Variablen nicht.
4. Wenn der FUNCTION-Block einen REPORT-Block enthält oder angibt, daß ein Standardbericht generiert werden soll, verarbeitet Net.Data den Berichtsabschnitt unter Verwendung der neuen Werte aller angegebenen Ausgabeparameter. Für integrierte Funktionen generiert Net.Data keine Berichte.
5. Wenn der FUNCTION-Block einen lokalen MESSAGE-Block enthält, verarbeitet Net.Data den MESSAGE-Block. Net.Data verarbeitet den globalen MESSAGE-Block, wenn eine der folgenden Bedingungen zutrifft:
  - Ein globaler MESSAGE-Block ist angegeben, und der Rückkehrcode wird nicht von einem lokalen MESSAGE-Block bearbeitet.
  - Eine integrierte Funktion wird aufgerufen.

Net.Data verarbeitet keine MESSAGE-Blöcke oder MACRO\_FUNCTION-Blöcke.

6. Net.Data ersetzt den Funktionsaufruf durch den zurückgegebenen Wert der Funktion. Bei FUNCTION-Blöcken ist dieser Wert einer der folgenden:

**RETURNS-Parameterwert** Ersetzt den FUNCTION-Block mit einem Schlüsselwort RETURNS.

**Leere Zeichenfolge ("" )** Ersetzt den FUNCTION-Block ohne Schlüsselwort RETURNS, wenn der Wert für RETURN\_CODE gleich Null ist.

**RETURN\_CODE** Ersetzt den FUNCTION-Block ohne Schlüsselwort RETURNS, wenn der Wert für RETURN\_CODE *ungleich* Null ist.

Bei MACRO\_FUNCTION-Blöcken ersetzen die Ergebnisse der Verarbeitung des Abschnitts mit den ausführbaren Anweisungen den Funktionsaufruf.

Bei integrierten Funktionen hängt der Wert vom Format der integrierten Funktion ab.

## Aufrufen gespeicherter Prozeduren

Eine gespeicherte Prozedur ist ein kompiliertes Programm, das auf dem lokalen oder fernen DB2-Server gespeichert ist und SQL-Anweisungen ausführen kann. In Net.Data werden gespeicherte Prozeduren von Net.Data-Funktionen mit Hilfe der SQL-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 erhöhen, indem Sie kompilierte SQL-Anweisungen auf dem Datenbank-Server speichern.

In diesem Abschnitt werden folgende Themen behandelt:

- „Syntax für gespeicherte Prozeduren“ auf Seite 97
- „Aufrufen einer gespeicherten Prozedur“ auf Seite 98
- „Übergeben von Parametern“ auf Seite 99
- „Verarbeiten von Ergebnismengen“ auf Seite 99

## 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) funktionsname ([IN datentyp arg1, INOUT datentyp arg2,  
    OUT tabellenname...])  
    CALL gespeicherte_prozedur [(ergebnismenge....)]  
[%REPORT(ergebnismenge...)]
```

Dabei gilt folgendes:

### *funktionsname*

Ist der Name der Net.Data-Funktion, die den Aufruf der gespeicherten Prozedur initialisiert.

### *gespeicherte\_prozedur*

Ist der Name der gespeicherten Prozedur.

### *datentyp*

Ist einer der von Net.Data unterstützten Datentypen der Datenbank wie in Tabelle 5 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.

### *tabellenname*

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 ergebnismenge übereinstimmen.

### *ergebnismenge*

Ein Name, der eine von der gespeicherten Prozedur zurückgegebene Ergebnismenge einem REPORT-Block zuordnet. Wenn dieser Parameter angegeben ist, muß er mit dem zugehörigen Parameternamen für tabellenname übereinstimmen.

*Tabelle 5. Datentypen für gespeicherte Prozeduren*

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

## Aufrufen einer gespeicherten Prozedur

Eine gespeicherte Prozedur wird wie folgt aufgerufen:

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

```
%function (dtw_sql) funktionsname()
```

2. Geben Sie wahlfrei IN-, INOUT- oder OUT-Parameter für die gespeicherte Prozedur an, einschließlich eines Tabellennamens 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).

```
%function (dtw_sql) funktionsname (IN datentyp  
arg1, INOUT datentyp arg2, OUT tabellenname...)
```

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

```
CALL gespeicherte_prozedur
```

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
```

Beispiel:

```
%function (dtw_sql) mystoredproc (IN CHAR(30) arg1) {  
    CALL myproc  
    %report {  
        ...  
        %row { .... %}  
        ...  
    %}  
%}
```

5. Wenn die gespeicherte Prozedur mehrere Ergebnismengen generiert, haben Sie folgende Möglichkeiten:

- Geben Sie einen oder mehrere Namen für Ergebnismengen in der Anweisung CALL an.

```
CALL gespeicherte_prozedur (ergebnismenge1, ergebnismenge2, ...)
```

- Geben Sie wahlfrei einen oder mehrere REPORT-Blöcke an, um zu definieren, wie Net.Data die Ergebnismengen anzeigen soll.

```
%REPORT(ergebnismenge1)
```

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. Ein Parameter wird an eine gespeicherte Prozedur übergeben, indem der Parametername mit dem Schlüsselwort IN angegeben wird. Wenn eine gespeicherte Prozedur den Parameter aktualisiert, müssen Sie den Parameter für den zurückzugebenden Wert mit dem Schlüsselwort INOUT bzw. OUT übergeben. Der Datentyp, der für einen Parameter angegeben wird, muß mit dem übereinstimmen, den die gespeicherte Prozedur erwartet.

### 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 angeben, ob eine oder mehrere Ergebnismengen von einer gespeicherten Prozedur zurückgegeben werden. Die Ergebnismengen können in Net.Data-Tabellen zur weiteren Verarbeitung innerhalb Ihres Makros gespeichert oder mit Hilfe eines REPORT-Blocks angezeigt werden. Sie müssen jeder durch eine gespeicherte Prozedur generierten Ergebnismenge einen Namen zuordnen. Dies geschieht durch die Angabe von Parametern in der SQL-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 im Standardformat für Net.Data-Berichte anzeigen, indem Sie keinen REPORT-Block für die Ergebnismenge definieren.
- Sie können eine Ergebnismenge einem REPORT-Block zuweisen, damit Net.Data die Ergebnismenge in einem Bericht anzeigt. In diesem Fall können Sie mit Net.Data-Variablen, HTML- 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 in der Makrodatei 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.

Wenn eine gespeicherte Prozedur mehrere Ergebnismengen generiert, müssen Sie in der SQL-Anweisung CALL einen Parameter für jede Ergebnismenge angeben.

**Wenn nur eine Ergebnismenge zurückgegeben und diese in einem Standardbericht angezeigt werden soll:**

Verwenden Sie die folgende Syntax:

```
%function (dtw_sql) funktionsname () {  
    CALL gespeicherte_prozedur ()  
%}
```

Beispiel:

```
%function (dtw_sql) mystoredproc() {  
    CALL myproc  
%}
```

**Wenn nur eine Ergebnismenge zurückgegeben und ein REPORT-Block für die Verarbeitung der Anzeige angegeben werden soll:**

Verwenden Sie die folgende Syntax:

```
%function (dtw_sql) funktionsname () {  
    CALL gespeicherte_prozedur  
    %REPORT (ergebnismenge) {  
        ...  
    %}  
%}
```

Beispiel:

```
%function (dtw_sql) mystoredproc () {  
    CALL myproc  
    %REPORT {  
        ...  
        %row { .... %}  
        ...  
    %}  
%}
```

Alternativ kann die folgende Syntax verwendet werden:

```
%function (dtw_sql) funktionsname () {  
    CALL gespeicherte_prozedur (ergebnismenge)  
  
    %REPORT (ergebnismenge) {  
        ...  
    %}  
%}
```

Beispiel:

```
%function (dtw_sql) mystoredproc () {  
    CALL myproc (mytable1)  
    %REPORT (mytable1) {  
        ...  
        %row { .... %}  
        ...  
    %}  
%}
```



**Wenn nur eine Ergebnismenge in einer Net.Data-Tabelle zur weiteren Verarbeitung gespeichert werden soll:**

Verwenden Sie die folgende Syntax:

```
%function (dtw_sql) funktionsname (OUT tabellenname) {  
    CALL gespeicherte_prozedur (tabellenname)  
%}
```

Beispiel:

```
%define DTW_DEFAULT_REPORT = "N0"
```

```
%function (dtw_sql) mystoredproc (OUT mytable1) {  
    CALL myproc (mytable1)  
%}
```

Beachten Sie, daß die Variable DTW\_DEFAULT\_REPORT auf den Wert N0 gesetzt ist, so daß kein Standardbericht für die Ergebnismenge generiert wird.

**Wenn mehrere Ergebnismengen zurückgegeben und diese im Standardberichtsformat angezeigt werden sollen:**

Verwenden Sie die folgende Syntax:

```
%function (dtw_sql) funktionsname () {  
    CALL gespeicherte_prozedur (tabellenname1, tabellenname2)  
%}
```

Beispiel:

```
%function (dtw_sql) mystoredproc () {  
    CALL myproc (mytable1, mytable2)  
%}
```

**Wenn mehrere Ergebnismengen zurückgegeben und die Ergebnismengen in Net.Data-Tabellen zur weiteren Verarbeitung gespeichert werden sollenn:**

Verwenden Sie die folgende Syntax:

```
%function (dtw_sql) funktionsname (OUT tabellenname1, tabellenname2) {  
    CALL gespeicherte_prozedur (ergebnismenge1, ergebnismenge2)  
%}
```

Beispiel:

```
%define DTW_DEFAULT_REPORT = "N0"  
  
%function (dtw_sql) mystoredproc (OUT mytable1 mytable2) {  
    CALL myproc (mytable1, mytable2)  
%}
```

Beachten Sie, daß die Variable DTW\_DEFAULT\_REPORT auf den Wert N0 gesetzt ist, so daß kein Standardbericht für die Ergebnismengen generiert wird.

**Wenn mehrere Ergebnismengen zurückgegeben und REPORT-Blöcke für die Verarbeitung der Anzeige angegeben werden sollen:**

Jeder Ergebnismenge wird ein eigener REPORT-Block zugeordnet. Verwenden Sie die folgende Syntax:

```
%function (dtw_sql) funktionsname () {  
    CALL gespeicherte_prozedur (ergebnismenge1, ergebnismenge2)  
    %REPORT (ergebnismenge1)  
    ...  
    %row { .... %}  
    ...  
    %}  
    %REPORT (ergebnismenge2)  
    ...  
    %row { .... %}  
    ...  
    %}  
%}
```

**Beispiel:**

```
%function (dtw_sql) mystoredproc () {  
    CALL myproc (mytable1, mytable2)  
  
    %REPORT(mytable1)  
    ...  
    %row { .... %}  
    ...  
    %}  
  
    %REPORT(mytable2)  
    ...  
    %row { .... %}  
    ...  
    %}  
%}
```

**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 Makroschreiber 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.

## **Richtlinien und Einschränkungen für die Rückgabe mehrerer Ergebnismengen**

Beachten Sie die folgenden Richtlinien und Einschränkungen für die Rückgabe mehrerer Ergebnismengen von einer gespeicherten Prozedur.

### **Richtlinien:**

- Geben Sie einen REPORT-Block pro Ergebnismenge an. Der für den REPORT-Block angegebene Parameter für den Tabellennamen muß mit dem entsprechenden Parameter für den Tabellennamen in der Anweisung CALL übereinstimmen.
- Geben Sie die REPORT-Blöcke für mehrere Ergebnismengen in der Reihenfolge an, in der die Ergebnismengen verarbeitet werden sollen.
- Wenn die Standardverarbeitung erfolgen soll und kein REPORT-Block für eine Ergebnismenge angegeben ist, definieren Sie DTW\_DEFAULT\_REPORT = "YES". Beim Aufbau der Web-Seite zeigt Net.Data die Standardberichte für Ergebnismengentabellen nach den Berichten für Ergebnismengen mit REPORT-Blöcken an.
- Wenn Net.Data Ergebnismengen ohne REPORT-Blöcke nicht anzeigen soll, setzen Sie DTW\_DEFAULT\_REPORT = "NO".
- Wenn die Variable DTW\_SET\_TABLE\_IN zusammen mit einer gespeicherten Prozedur verwendet wird, wird die erste Ergebnismenge, die von der gespeicherten Prozedur zurückgegeben wird, der DTW\_SET\_TABLE\_IN-Tabelle zugeordnet.

### **Einschränkungen:**

- Mehrere REPORT-Blöcke können nur in gespeicherten Prozeduren in der DB2-Sprachumgebung DTW\_SQL verwendet werden.
- Berichtsvariablen werden für eine vollständige Funktion definiert und beeinflussen die Verarbeitung aller REPORT-Blöcke und die Ergebnismengen, die sie verarbeiten. 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 in einer gespeicherten Prozedur verwendet werden können.
- Für ein und dieselbe Ergebnismenge können nicht mehrere REPORT-Blöcke angegeben werden.
- Die maximale Anzahl von Ergebnismengen für eine einzige gespeicherte Prozedur ist 32.

## 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. Net.Data verwendet den folgenden MESSAGE-Blockprozeß:

1. Net.Data setzt die Variable RETURN\_CODE, eine Sprachumgebungsvariable, für jeden Funktionsaufruf an einen FUNCTION-Block. RETURN\_CODE wird nicht bei einem Funktionsaufruf an einen MACRO\_FUNCTION-Block gesetzt.
2. Wenn eine Sprachumgebung einen Rückkehrcodewert an Net.Data zurückgibt, setzt Net.Data den Wert von RETURN\_CODE auf den Wert des Rückkehrcodes.
3. Nach Beendigung des Funktionsaufrufs bestimmt der MESSAGE-Block anhand des Werts von RETURN\_CODE die weitere Vorgehensweise.

Ein MESSAGE-Block besteht aus einer Reihe von Nachrichtenweisungen, 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, besitzt 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
```

---

## Generieren von HTML in einem Makro

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

### HTML-Blöcke

Die Net.Data-Makrodatei enthält HTML-Blöcke und die Funktionen in den HTML-Blöcken, die die HTML-Ausgabe für einen Web-Browser generieren. In einer Makrodatei muß mindestens ein HTML-Block definiert werden. Der Inhalt des HTML-Blocks steuert den übrigen Net.Data-Aufruf.

Jede gültige HTML-Anweisung kann in einem HTML-Block verwendet werden. Darüber hinaus können Sie in einem HTML-Block INCLUDE-Anweisungen, Funktionsaufrufe und Variablenverweise verwenden. Das folgende Beispiel zeigt eine gängige Verwendung von HTML-Blocks in einem Net.Data-Makro:

```
%DEFINE DATABASE="MNS96"
```

```
    %HTML(INPUT){
<H1>Hardware Query Form</H1>
<HR>
<FORM METHOD="POST" ACTION="/cgi-bin/db2www/equip1st.d2w/report">
<dl>
<dt>What hardware do you want to list?
<dd><input type="radio" name="hardware" value="MON" checked>Monitors
<dd><input type="radio" name="hardware" value="PNT">Pointing devices
<dd><input type="radio" name="hardware" value="PRT">Printers
<dd><input type="radio" name="hardware" value="SCN">Scanners
</dl>
<HR>
<input type="submit" value="Submit">
</FORM>
%}
```

```
%FUNCTION(DTW_SQL) myQuery() {
SELECT MODNO, COST, DESCRIP FROM EQPTABLE WHERE TYPE=$(hardware)
    %REPORT{
<B>Here is the list you requested:</B><BR>
    %ROW{
<HR>
$(N1): $(V1)    $(N2): $(V2)
<P>
$(V3)
%}
%}
%}

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

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/equip1st.d2w/input">  
List of hardware</a>
```

Wenn der Anwendungsbenutzer diese Verbindung anklickt, ruft der Web-Browser Net.Data auf, und Net.Data analysiert die Makrodatei. 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 als HTML-Anweisung interpretiert und zum Anzeigen an den Browser gesendet.

Wenn der Benutzer eine Auswahl getroffen und den Knopf zur Übergabe (Submit) gedrückt hat, führt Net.Data den ACTION-Abschnitt des HTML-Elements FORM aus, der einen Aufruf an den Block HTML(REPORT) des Net.Data-Makros definiert. Net.Data verarbeitet daraufhin den Block HTML(REPORT) ebenso wie zuvor den Block HTML(INPUT).

Anschließend verarbeitet Net.Data den Funktionsaufruf myQuery(), der wiederum den SQL-FUNCTION-Block aufruft. Nachdem der Variablenverweis \$(hardware) in der SQL-Anweisung durch den im INPUT-Formular empfangenen Wert ersetzt wurde, führt Net.Data die Abfrage aus. An dieser Stelle beginnt Net.Data wieder damit, den HTML-Bericht an den Browser zu senden, der die Ergebnisse der Abfrage gemäß den im REPORT-Block definierten HTML-Anweisungen anzeigt.

Wenn Net.Data die Verarbeitung des REPORT-Blocks abgeschlossen hat, kehrt Net.Data zum Block HTML(REPORT) zurück und beendet die Verarbeitung.

Net.Data verarbeitet bei jedem Aufruf nur einen HTML-Block. Allerdings können Sie mit Hilfe von HTML-Verbindungen und -Formularen auf einfache Weise dem Benutzer einer Anwendung einen weiteren Aufruf von Net.Data für einen anderen HTML-Block ermöglichen, wobei die Steuerung ganz bei Ihnen liegt.



## 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 HTML-Befehlen, Makrovariablenverweisen und Funktionsaufrufen angegeben werden kann. Wahlfrei kann im REPORT-Block ein Tabellenname angegeben werden. Wenn kein Tabellenname angegeben wird, verwendet Net.Data die Tabellendaten aus der ersten Ausgabetabelle in der Parameterliste des FUNCTION-Blocks. Wenn Sie keine Tabelle im FUNCTION-Block angeben, verwendet Net.Data die Standardtabellendaten.

Der REPORT-Block besteht aus drei Teilen, die jeweils wahlfrei sind:

- Kopfdaten, die HTML-Daten enthalten, die einmal vor den Tabellenzeilendaten angezeigt werden.
- ROW-Block, der HTML- und Tabellenvariablen enthält, die einmal für jede Zeile der Ergebnistabelle angezeigt werden.
- Fußzeilen, die Daten enthalten, die einmal nach den Tabellenzeilendaten angezeigt werden.

Wenn Sie keine Tabellenausgabe des ROW-Blocks anzeigen wollen, nehmen Sie keine Angaben im ROW-Block vor, oder übergehen Sie ihn ganz.

Innerhalb des REPORT-Blocks können Sie verschiedene von Net.Data bereitgestellte Variablen zum Zugriff auf die Daten in der Net.Data-Makroergbnistabelle verwenden. Diese Variablen sind im Abschnitt „Variablen zur Tabellenverarbeitung“ auf Seite 88 beschrieben. Weitere Einzelheiten finden Sie im Abschnitt über die Berichtsvariablen (Report Variables) im Handbuch *Net.Data Reference*.

Wenn Sie Kopf- und Fußzeilendaten zur Verfügung stellen wollen, geben Sie den Text vor oder nach dem ROW-Block an. Net.Data verarbeitet alle Angaben, die vor einem ROW-Block angetroffen werden, und behandelt sie als Kopfdaten und behandelt alle Angaben nach dem ROW-Block als Fußzeilendaten. Wie beim HTML-Block behandelt Net.Data alle Angaben im Kopfdatenblock, ROW-Block und Fußzeilendatenblock, die nicht als Makrosprachkonstrukte erkannt werden, als HTML und sendet die HTML-Daten an den Browser.

Sie können außerdem Funktionen und Variablen im REPORT-Block verwenden.

Wenn Net.Data einen Standardbericht mit vorformatiertem Text ausgeben soll, geben Sie den REPORT-Block nicht in der Makrodatei an. 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 den Wert 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{%}
```

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:

```
%FUNCTION(DTW_SQL) custlist() {  
    SELECT Name, Phone, Fax FROM CustomerTbl  
    %REPORT{  
<I>Phone Query Results:</I>  
<BR>  
=====
```

Name:	Doen, David
Phone:	422-245-1293
Fax:	422-245-7383

```
-----  
<BR>  
    %}  
    Total records retrieved: $(NUM_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. Ausgeben des Titels *Phone Query Results*: einmal am Anfang des Berichts
2. Einsetzen der Werte für Name, Phone und Fax in die Variablen V1, V2 und V3 für jede abgerufene Zeile
3. Zeichnen einer Linie nach jeder abgerufenen Zeile zur besseren Lesbarkeit
4. Ausgeben der Zeichenfolge *Total records retrieved*: und des Werts für NUM\_ROWS einmal am Ende des Berichts

Sie können mehrere REPORT-Blöcke für gespeicherte Prozeduren zur Rückgabe mehrerer Ergebnismengen verwenden. Mehrere REPORT-Blöcke werden für gespeicherte Prozeduren nur bei Verwendung der Sprachumgebung DTW\_SQL unterstützt. Im Abschnitt „Aufrufen gespeicherter Prozeduren“ auf Seite 96 finden Sie Informationen darüber, wie mehrere Ergebnismengen von einer gespeicherten Prozedur zurückgegeben werden können.

---

## Bedingte Logik und Schleifen in einer Makrodatei

Net.Data ermöglicht Ihnen, bedingte Logik und Schleifen in Ihr Net.Data-Makro mit Hilfe von IF- und WHILE-Blöcken zu integrieren.

- „Bedingte Logik“
- „Schleifenkonstrukte“ auf Seite 114

### Bedingte Logik

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 Sprachen 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 sie verschachteln. Die Syntax eines IF-Blocks wird im Kapitel über die Sprachkonstrukte im Handbuch *Net.Data Reference* dargestellt.

Die Syntaxregeln für einen IF-Block werden durch die Position des Blocks in der Makrodatei 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, daß 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
```

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
```

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 block
...
    %IF block
```

**Ausnahme:** Geben Sie keinen ROW-Block in einem IF-Block an, wenn sich der IF-Block in einem REPORT-Block befindet.

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 Zeichenfolgenvergleiche wie in den Bedingungen angegeben durchzuführen. Wenn jedoch die beiden folgenden Bedingungen zutreffen, führt Net.Data einen numerischen Vergleich durch:

- Die Bedingung ist eine Binäroperation (<, >, <=, >=, !=, ==).
- Wenn beide Ausdrücke in der Bedingung ganze Zahlen darstellen. Das heißt, die Ausdrücke sind Zeichenfolgen aus Ziffern, denen wahlfrei ein Zeichen '+' oder '-' vorangestellt sein kann. Die Zeichenfolge darf keine Nichtziffernzeichen außer '+' bzw. '-' enthalten.

**Beispiele für gültige Zeichenfolgen:**

```
+1234567890
-47
000812
92000
```

### Beispiele für ungültige Zeichenfolgen:

- 20           (enthält Leerzeichen)
- 234,000       (enthält ein Komma)
- 57.987       (enthält ein Dezimalzeichen)

Net.Data wertet den IF-Block zum Zeitpunkt der Ausführung des Blocks aus, der sich von dem Zeitpunkt, zu dem er ursprünglich von Net.Data gelesen wird, unterscheiden kann. Wenn Sie zum Beispiel einen IF-Block in einem REPORT-Block angeben, wertet Net.Data die Bedingungsliste des IF-Blocks nicht aus, wenn Net.Data die FUNCTION-Blockdefinition liest, die den REPORT-Block enthält, sondern erst, wenn Net.Data die Funktion aufruft und sie ausführt. Dies gilt sowohl für den Abschnitt mit der Bedingungsliste des IF-Blocks als auch für den Block der auszuführenden Anweisungen.

**Einschränkung:** Net.Data unterstützt keinen numerischen Vergleich von nichtganzzahligen Zahlen.

**Beispiel:** Dieses Beispiel zeigt eine Makrodatei, die IF-Blöcke in anderen Blöcken enthält:

```
%{ 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

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 Bedingungs-tests einen Block von Anweisungen auszuführen. Im Gegensatz zum IF-Block kann der Anweisungsblock auf der Grundlage des Ergebnisses des Bedingungs-tests beliebig oft ausgeführt werden.

Sie können WHILE-Blöcke innerhalb von HTML-Blöcken, REPORT-Blöcken, ROW-Blöcken, MACRO\_FUNCTION-Blöcken und HTML-IF-Blöcken angeben, und Sie können sie verschachteln. Die Syntax eines WHILE-Blocks wird im Kapitel über die Sprachkonstrukte im Handbuch *Net.Data Reference* dargestellt.

Net.Data verarbeitet den WHILE-Block in exakt der gleichen Weise wie auch der IF-Block verarbeitet wird, jedoch wird die Bedingungsliste bei jedem Durchlaufen der Schleife erneut ausgewertet. Und wie bei jedem bedingten Schleifenkonstrukt kann die Verarbeitung zu einer Endlosschleife führen, wenn die Bedingung nicht korrekt codiert wurde.

**Beispiel:** Dieses Beispiel zeigt eine Makrodatei 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 großer Objekte

Große Multimedia- und Dokumentobjekte sind Binärdateien wie BMP-, TIF-, GIF- und PostScript-Dateien, die zum Anzeigen oder Drucken verwendet werden. Sie können diese Dateien in DB2-Datenbanken auf den meisten Betriebssystemen speichern und über die SQL-Sprachumgebung für Ihre Web-Anwendung auf sie zugreifen.

Net.Data unterstützt die folgenden Arten großer Objekte (LOBs):

- Große Binärobjekte (BLOBs)
- Große Zeichenobjekte (CLOBs)
- Große Doppelbyte-Zeichenobjekte (DBLOBs)

LOBs enthalten in der Regel in den ersten Byte eine Dateikennung, die die Art der in der Datei enthaltenen Informationen angibt. Wenn Net.Data ein LOB erkennt, fügt Net.Data der temporären Datei und der Net.Data-Makrovariablen, die für den Namen der Datei steht, die entsprechende Erweiterung hinzu. Wenn Sie keinen REPORT-Block in der Makrodatei haben, hängt Net.Data die Erweiterung .txt an CLOB-Dateien an. Net.Data erkennt die folgenden LOB-Formate:

- Bitmap (.bmp)
- Graphical Image Format (.gif)
- Tag Image File Format (.tif)
- Postscript (.ps). Diese müssen als BLOBs, nicht als CLOBs gespeichert werden.

**Einschränkung:** Net.Data erkennt und unterstützt keine anderen Dateitypen. Außerdem unterstützt Net.Data die SQL-Anweisungen UPDATE und INSERT für große Objekte nicht.

**Hinweis zur Planung:** Wenn eine Abfrage ein LOB zurückgibt, sichert Net.Data diese Datei in dem durch die Konfigurationsanweisung HTML\_PATH angegebenen Verzeichnis. Bei der Verwendung von LOBs ist die jeweilige Systemausstattung zu berücksichtigen, da diese Objekte die Systemressourcen schnell aufbrauchen können. Einige LOBs, wie beispielsweise Audiodateien, erfordern spezielle Hardware und Software.

**Beispiel 1:** Zeigt ein Bild inline an

```
<IMG SRC="/tmplobs/filename">  
<A HREF="/tmplobs/filename">filename</A>
```

**Beispiel 2:** Im folgenden Beispiel muß der Anwendungsbenutzer den Dateinamen anklicken, um das Anzeigeprogramm aufzurufen, weil die Anwendung eine .WAV-Datei verwendet. Net.Data erkennt diesen Dateityp nicht, so daß eine Variable EXEC verwendet wird, um die Erweiterung an die Datei anzuhängen.

```

%DEFINE{
docroot="/usr/lpp/internet/server_root/html"
rename=%EXEC "rename ${docroot}${V3} ${docroot}${V3}.wav"
%}

%FUNCTION(DTW_SQL) queryData() {
SELECT Name, IDPhoto, Voice FROM RepProfile
  %REPORT{
<P>Here are the images you selected:<P>
  %ROW{
$(rename)
$(V1) Voice sample <IMG SRC="${V2}">
<A HREF="${V3}.wav">Voice sample</A><P>
%}
%}
%}

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

```

Die Funktion queryData gibt folgenden HTML-Text zurück:

```

<P>Here are the images you selected:<P>
Kinson Yamamoto
<IMG SRC="/tmplobs/p2345n1.gif">
<A HREF="/tmplobs/p2345n2.wav">Voice sample</A><P>
Merilee Lau
<IMG SRC="/tmplobs/p2345n3.gif">
<A HREF="/tmplobs/p2345n4.wav">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 einer Person in einer .GIF-Datei. Das Bild wird inline angezeigt. Net.Data fügt das temporäre LOB-Verzeichnis und die Erweiterung .GIF automatisch ein.
- V3 ist ein Beispiel für ein Stimmuster der Person in einer .WAV-Datei. Wenn Net.Data ein unbekanntes Dateiformat wie zum Beispiel eine .WAV-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 Variablen EXEC 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.

Nicht alle Web-Browser unterstützen Grafik und Ton. Zur Unterstützung der hier beschriebenen Funktionen wird möglicherweise spezielle Hardware und Software benötigt, wie zum Beispiel eine Audiokarte mit Treiber.



---

## Verwenden integrierter Funktionen

Net.Data verfügt über zahlreiche integrierte Funktionen, die die Entwicklung von Web-Seiten vereinfachen. Diese Funktionen sind von Net.Data bereits definiert, so daß Sie diese Funktionen nicht mehr in einem FUNCTION-Block definieren müssen. Rufen Sie diese Funktionen einfach in einem Makro an einer beliebigen Stelle auf, an der eine benutzerdefinierte Funktion aufgerufen werden kann.

Es gibt drei Methoden, mit denen integrierte Funktionen ihre Ergebnisse zurückgeben können. An dem zugehörigen Präfix können Sie erkennen, wie eine Funktion ihre Ergebnisse ausgibt:

- **DTW\_, DTWF\_ und DTWR\_:** Die Ergebnisse des Aufrufs werden in einem Ausgabeparameter zurückgegeben, bzw., es wird kein Ergebnis zurückgegeben. (**DTWF\_** ist das Präfix für Funktionen von unstrukturierten Textdateien. **DTWR\_** ist das Präfix für Web-Registrierdatenbankfunktionen.)
- **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 den Parametern zurückgegeben, die an die Funktion übergeben werden.

Einige integrierte Funktionen unterstützen nicht jede Art. Weitere Informationen finden Sie im Handbuch *Net.Data Reference*.

In der folgenden Liste wird eine allgemeine Übersicht über die integrierten Net.Data-Funktionen gegeben. Mit diesen Funktionen können Sie allgemeine, mathematische, Zeichenfolge-, Wort- bzw. Tabellenbearbeitungsfunktionen ausführen. Beschreibungen der einzelnen Funktionen mit Syntax und Beispielen finden Sie im Handbuch *Net.Data Reference*.

### Allgemeine Funktionen

Mit Hilfe dieser Gruppe von Funktionen können Sie Web-Seiten durch Ändern von Daten oder Zugreifen auf Systemservices entwickeln. Hiermit können Sie Umgebungsvariablen abfragen und einstellen, HTML-Escape-Codes verwenden, und andere nützliche Informationen vom System abrufen.

### 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 Ergebnissenauigkeit angegeben und Exponentialschreibweise verwendet werden.

### 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

Zeichenfolgewart einer anderen Variablen zuordnen und noch andere nützliche Funktionen ausführen.

### **Wortfunktionen**

Diese Funktionen können zum Bearbeiten von Wörtern in Zeichenfolgen verwendet werden. Die meisten dieser Funktionen arbeiten auf dieselbe Weise wie die 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.

### **Tabellenfunktionen**

Diese Funktionen können zum Generieren von Berichten oder Formularen mit Hilfe der Daten in einer Net.Data-Tabellenvariable verwendet werden. Tabellenvariablen enthalten einen Wertebereich mit den zugehörigen Spaltennamen. Sie bieten eine bequeme Möglichkeit, ganze Wertegruppen an eine Funktion weiterzugeben.

### **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.

### **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.

## 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 Hilfe der SYSTEM-Sprachumgebung können Sie außerdem ein Programm ausführen oder einen Befehl absetzen.

Bei 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 Bibliothek für dynamisches Verbinden (DLL) oder gemeinsam benutzte Bibliothek implementiert werden. Sie müssen der Net.Data-Initialisierungsdatei eine Anweisung ENVIRONMENT hinzufügen, um einer benutzerdefinierten Sprachumgebung eine DLL zuzuordnen. Beim ersten Funktionsaufruf für einen FUNCTION-Block, der den Sprachumgebungsnamen angibt, lädt Net.Data eine DLL und führt sie aus. Bei nachfolgenden Funktionsaufrufen für FUNCTION-Blöcke, die den gleichen Sprachumgebungsnamen angeben, führt Net.Data die DLL lediglich aus, da DLLs nur einmal geladen werden.

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

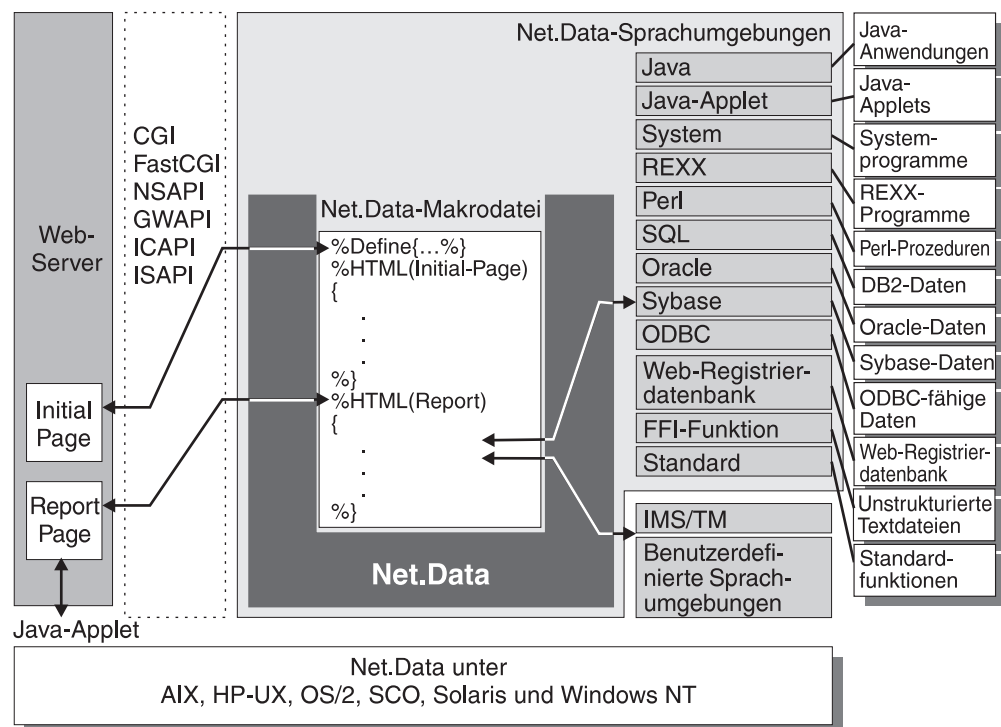


Abbildung 21. Die Net.Data-Sprachumgebungen

Ausführliche Informationen zu den von Net.Data bereitgestellten Sprachumgebungen und zum Erstellen einer benutzerdefinierten Sprachumgebung finden Sie im Handbuch *Net.Data Language Environment Reference*.



---

## Aufrufen von Net.Data mit Java-Servlets und JavaBeans

Net.Data stellt auf Java basierende Klassen bereit, mit denen Sie über Net.Data auf einem beliebigen Java-fähigen Betriebssystem, das von Net.Data unterstützt wird, Net.Data-Makrodateien 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 128

---

### 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-Makrodateien 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 Makrodateien. 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 C, „Verwenden von Plug-Ins für NetObjects Fusion (NOF) mit Net.Data-Servlets“ auf Seite 177.

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

- „Informationen zu Net.Data-Servlets“
- „Definieren von Servlets“ auf Seite 122
- „Ausführen von Net.Data-Servlets“ auf Seite 122

### 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 eine Makrodatei oder Direktanforderung von Net.Data Version 2 mit Hilfe einer systemeigenen DLL-Datei ausführen. Mit diesen Servlets können Sie eine vorhandene Makrodatei (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_verz>/servlets/NetDataServlets.jar`.

Net.Data wird mit zwei Servlets geliefert:

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

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

Nachfolgend einige Vorteile des Makro-Servlet:

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

Das Makro-Servlet ermöglicht zudem systemeigenen 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 Makrodatei (Direktanforderung)“ auf Seite 63. Für den Einsatz des Funktions-Servlet ist die Installation von Net.Data Version 2 erforderlich.

## **Definieren von Servlets**

Weitere Informationen zum Registrieren und Verwenden von Servlets finden Sie in der Dokumentation zu Ihrem Web-Server. Die Net.Data-Servlets befinden sich in der Net.Data-Datei `<inst_verz>/servlets/NetDataServlets.jar`. Möglicherweise ist es für Ihren Web-Server erforderlich, `<inst_verz>/servlets/NetDataServlets.jar` zu Ihrer Umgebungsvariablen CLASSPATH hinzuzufügen.

## **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 C, „Verwenden von Plug-Ins für NetObjects Fusion (NOF) mit Net.Data-Servlets“ auf Seite 177.

- „Ausführen des Makro-Servlet“ auf Seite 123
- „Ausführen des Funktions-Servlet“ auf Seite 125

## Ausführen des Makro-Servlet

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

### 1. URL:

```
http://myserver/servlet/com.ibm.netdata.servlets.MacroServlet
```

```
?MACRO=makrowert&BLOCK=blockwert&parmn=wertnn
```

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="makrowert">  
  <param name="BLOCK" value="blockwert">  
  <param name="parmn" value="wertnn">  
</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:

*makrowert* Der vollständig qualifizierte Pfad zu einer vorhandenen Net.Data-Makrodatei

*blockwert* Der Name des auszuführenden HTML-Blocks in der angegebenen Net.Data-Makrodatei. Der Standardwert ist report (wahlfrei).

*parmn* Zusätzliche, für die Makrodatei erforderliche Parameter wie  
<param name="field1" ...

*wertnn* Zusätzliche, für die Makrodatei 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:

```
http://myserver/servlet/com.ibm.netdata.servlets.MacroServlet  
?MACRO=makroname&BLOCK=blockwert&htmlpath=html_pfad&parmn=wertnn
```

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="makrowert">
  <param name="BLOCK" value="blockwert">
  <param name="htmlpath" value="html_pfad">
  <param name="parmn" value="wertnn">
</servlet>
```

Beispiel:

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

**Parameter INBUFLEN und OUTBUFLEN:** Wenn Ihre Eingabe in die Makrodatei größer ist als 1 KB, müssen Sie den Parameter INBUFLEN angeben. Wenn die Ergebnisse Ihrer Makrodatei 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:

```
http://myserver/servlet/com.ibm.netdata.servlets.MacroServlet
?MACRO=makroname&BLOCK=blockwert&INBUFLEN=eingabepuffergröße
&OUTBUFLEN=ausgabepuffergröße&parmn=wertnn
```

Beispiel:

```
http://myserver/servlet/com.ibm.netdata.servlets.MacroServlet?MACRO=my_macro
&BLOCK=my_blockINBUFLEN=3K&OUTBUFLEN=48K&field1=custno
```

- SSI:

```
<servlet code="com.ibm.netdata.servlets.MacroServlet">
  <param name="MACRO" value="makrowert">

  <param name="BLOCK" value="blockwert">

  <param name="INBUFLEN" value="eingabepuffergröße">

  <param name="OUTBUFLEN" value="ausgabepuffergröße">
  <param name="parmn" value="wertnn">
</servlet>
```

Beispiel:

```
<servlet code="com.ibm.netdata.servlets.MacroServlet">
<param name="MACRO" value="my_makro">
<param name="BLOCK" value="my_block">
<param name="INBUFLEN" value="3K">
<param name="OUTBUFLEN" value="48K">
<param name="field1" value="custno">
</servlet>
```



## Ausführen des Funktions-Servlet

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=sprachumgebungsname&FUNC=programmname&parmnn=wertnn
```

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=datenbanksprachumgebungsname&SQL=SQL_anweisung  
&DATABASE=datenbankname&parmnn=wertnn
```

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="sprachumgebungsname">  
  <param name="FUNC" value="programmname">  
  <param name="parmnn" value="wertnn">  
</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="sprachumgebungsname">
  <param name="SQL" value="SQL_anweisungsname">
  <param name="DATABASE" value="datenbankname">
  <param name="parmnn" value="wertnn">
</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:**

*sprachumgebungsname* 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.

*programmname* 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 *wertnn* Eingabeparameter für die Funktion ein.

*datenbankname* Der Name der Datenbank, der dem Parameter DATABASE zugeordnet ist

*SQL\_anweisungsname* 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 *wertnn* Eingabeparameter für die SQL-Anweisung bzw. gespeicherte Prozedur ein.

*parmnn* Zusätzliche, für die Makrodatei erforderliche Werte wie <param name="field1" ...

*wertnn* Zusätzliche, für die Makrodatei erforderliche Werte wie ... value="custno"

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

- URL:

```
http://myserver/servlet/com.ibm.netdata.servlets.FunctionServlet
?LANGENV=sprachumgebungsname&FUNC=programmname&htmlpath=html_pfad
&parmnn=wertnn
```

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="sprachumgebungsname">
  <param name="SQL" value="SQL_anweisungsname">
  <param name="htmlpath" value="html_pfad">
  <param name="parmnn" value="wertnn">
</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\_pfad* den Pfad zum HTML-Stammverzeichnis des Web-Servers an, zum Beispiel *htmlpath=e:\html*.

**Parameter INBUFLEN und OUTBUFLEN:** Wenn Ihre Eingabe in die Makrodatei größer ist als 1 KB, müssen Sie den Parameter INBUFLEN angeben. Wenn die Ergebnisse Ihrer Makrodatei 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:

```
http://myserver/servlet/com.ibm.netdata.servlets.FunctionServlet
?LANGENV=sprachumgebungsname&FUNC=programmname&INBUFLEN=eingabepuffergröße
&OUTBUFLEN=ausgabepuffergröße&parmnn=wertnn
```

Beispiel:

```
http://myserver/servlet/com.ibm.netdata.servlets.FunctionServlet?
LANGENV=DTW_REXX
&FUNC=my_rexxINBUFLEN=3K&OUTBUFLEN=48K&field1=custno
```

- SSI:

```
<servlet code="com.ibm.netdata.servlets.FunctionServlet">
  <param name="LANGENV" value="sprachumgebungsname">
  <param name="FUNC" value="programmname">
  <param name="INBUFLEN" value="eingabepuffergröße">

  <param name="OUTBUFLEN" value="ausgabepuffergröße">
  <param name="parmnn" value="wertnn">
</servlet>
```

Beispiel:

```
<servlet code="com.ibm.netdata.servlets.FunctionServlet">
<param name="LANGENV" value="DTW_REXX">
<param name="FUNC" value="my_rexx">
<param name="INBUFLEN" value="3K">
<param name="OUTBUFLEN" value="48K">
<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 systemeigenen DLL-Datei von Net.Data auf und füllt den Rückkehrcode und eine Zeichenfolge mit der Net.Data-Ausgabe (Ergebnisse) auf. Da JavaBeans eine systemeigene 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 im Format der Rückgabe Ihrer Makrodatei oder 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-Makrodateien
- 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“ auf Seite 130

## 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 des mitgelieferten Anpassungsprogramms zur Verarbeitung verbinden, die Ergebnisse einer Makrodatei bzw. 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_verz>/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 systemeigenen 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-Entwicklungshilfsprogramm wie Bean Machine definieren und ausführen können. Die Schritte zur Verwendung der Entwicklungshilfsprogramme sind generisch, d. h., Sie können das Hilfsprogramm Ihrer Wahl einsetzen.

- „Definieren des Makro-Bean“
- „Ausführen des Makro-Bean“ auf Seite 131
- „Definieren des SQL-Bean“ auf Seite 131
- „Ausführen des SQL-Bean“ auf Seite 132

### Definieren des Makro-Bean

Mit dem Net.Data-Makro-Bean (com.ibm.netdata.beans.NetDataMacro) können Sie Java zum Ausführen einer vorhandenen Makrodatei verwenden. Zur Verwendung dieses Bean müssen Sie entsprechende Net.Data-Merkmale angeben, damit es mit der Makrodatei arbeiten kann.

#### ***Gehen Sie wie folgt vor, um das Net.Data-Makro-JavaBean mit einem JavaBean-Entwicklungshilfsprogramm zu definieren:***

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

**Makro** Gibt den Namen der auszuführenden, vorhandenen Makrodatei an, zum Beispiel: `PsMakro.mac`

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

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

**Parameter** Gibt den Parameternamen und bei der Ausführung des Makros zu verwendende Werte an

#### **Syntax:**

`name1=wert1&nameN=wertN`

## Ausführen des Makro-Bean

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

- Wählen Sie zur Ausführung des Makros die von Ihrem JavaBean-Entwicklungshilfsprogramm bereitgestellte Ausführungsmaßnahme aus.
- Nach der Ausführung des Makros können Sie auf die folgenden Ausgabemerkmale verweisen:  
**RC**           Gibt den von Net.Data zurückgegebenen Rückkehrcode an  
**Ergebnisse**  
                  Gibt die von der Ausführung der Net.Data-Makrodatei zurückgegebenen Daten an

## Definieren des SQL-Bean

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

***Gehen Sie wie folgt vor, um das Net.Data-Makro-JavaBean mit einem JavaBean-Entwicklungshilfsprogramm zu definieren:***

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

### **Sprachumgebung**

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-Pfad**

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

### **Parameter**

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

### **Syntax:**

name1=wert1&nameN=wertN

## Ausführen des SQL-Bean

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

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

**RC**           Gibt den von Net.Data zurückgegebenen Rückkehrcode an

### **Ergebnisse**

Gibt die von der SQL-Anweisung zurückgegebenen Daten an



---

## Net.Data-Caching

Durch Caching werden die Antwortzeiten für den Anwendungsbenutzer verbessert. Net.Data speichert Ergebnisse von einer Anforderung lokal auf dem Web-Server, 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“
- „Informationen zum Net.Data-Caching“ auf Seite 134
- „Terminologie für Net.Data-Caching“ auf Seite 134
- „Konzepte des Net.Data-Caching“ auf Seite 135
- „Einschränkungen des Net.Data-Caching“ auf Seite 136
- „Schnittstellen des Net.Data-Caching“ auf Seite 136
- „Planen für den Cache-Manager“ auf Seite 138
- „Cache-Kennungen“ auf Seite 138
- „Konfigurieren des Cache-Managers und der Net.Data-Caches“ auf Seite 139
- „Starten / Stoppen des Cache-Managers“ auf Seite 146
- „Caching von Web-Seiten“ auf Seite 147
- „Der Befehl CACHEADM“ auf Seite 151
- „Das Cache-Protokoll“ auf Seite 153

---

## 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 Tondateien sowie Java-Applets lokal im Hauptspeicher oder auf dem Datenträger, 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 Zugriffszeit auf den Datenträger zu verringern, wenn Benutzer wiederholt die gleichen Seiten abrufen.
- Ein Datenbankverwaltungssystem speichert Datenelemente, die in der Regel auf dem Datenträger gespeichert sind, im Hauptspeicher zwischen, um die Zugriffszeit auf den Datenträger 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
- Der 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 eigne 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. 22 zeigt, daß Net.Data einen Cache-Manager zum Verwalten des Caching von HTML-Ausgabe von einer Makrodatei verwendet. Zu dieser Ausgabe können Daten aus einer Datenbank gehören.

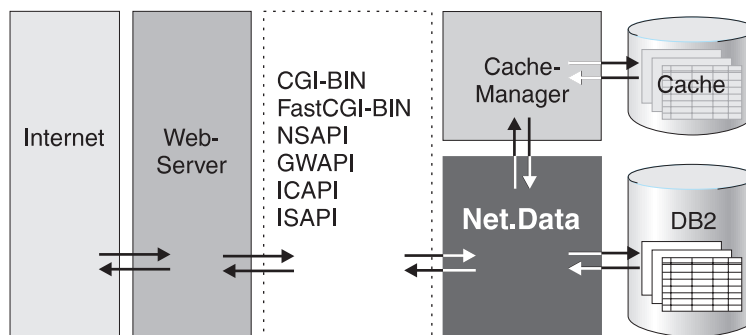


Abbildung 22. Net.Data-Caching

## 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 Dateiname lautet bei der Lieferung mit Net.Data `cachemgr.cnf`.

## Konzepte des Net.Data-Caching

Je nachdem, wie viele HTTP-Server sich auf Ihrem System befinden und ob jeder HTTP-Server (unter Verwendung separater Net.Data-Konfigurationsdateien) eine eigene Kopie von Net.Data ausführt, können alle Kopien von Net.Data einem bzw. mehreren Cache-Managern zugeordnet sein. Ein Cache-Manager kann eine Anzahl von Caches im Hauptspeicher unterstützen, wobei jeder Cache über eine Cache-Kennung, die sogenannte *Cache-ID*, verfügt. Abb. 23 zeigt einen Cache-Manager, der mit mehreren Makrodateien arbeitet und zwei Caches verwaltet.

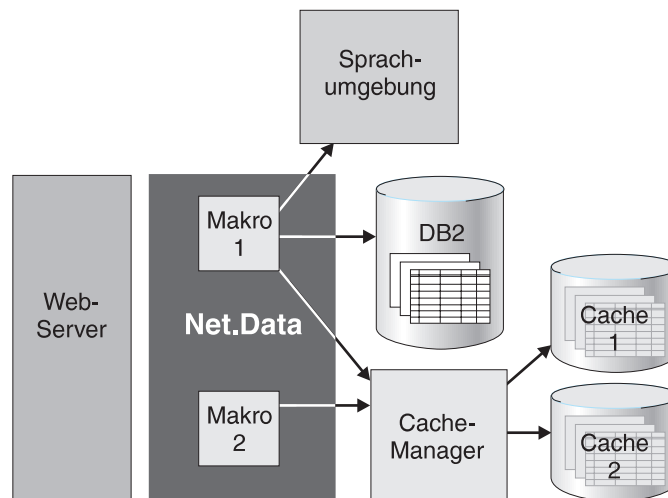


Abbildung 23. Cache-Manager arbeitet mit mehreren Makrodateien und Caches

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 Segment 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 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-Caching

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-Caching

Net.Data stellt eine Gruppe flexibler Schnittstellen bereit, mit denen Sie das Caching für Ihre Anwendung konfigurieren und definieren können. Die verschiedenen Optionen zur Verwendung der Net.Data-Caching-Funktionen werden in Tabelle 6 auf Seite 137 erläutert. Dort werden diese Funktionen zudem beschrieben.

Tabelle 6. 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 139
Cache-Konfigurationsoptionen	In einem einzigen Exemplar des Net.Data-Cache-Managers können Sie eine Anzahl von Caches zum Aufbewahren der zwischengespeicherten Elemente definieren. Jeder Cache verfügt über eine eigene Gruppe von Kenndaten, wie Größe und Speicherposition sowie die Cache-ID. Diese Kenndaten werden in der Zeilengruppe für den Cache in der Konfigurationsdatei für den Cache-Manager definiert. Jede Zeilengruppe wird durch die Cache-ID angegeben.	„Definieren eines Cache“ auf Seite 141
Net.Data-Initialisierungsoptionen	Wenn Net.Data und der entsprechende Cache-Manager auf separaten Systemen ausgeführt werden, dann geben Sie das Cache-Manager-System und die Anschlußnummer in der Net.Data-Initialisierungsdatei an.	„Konfigurationsvariablen für den Cache-Manager“ auf Seite 15
Integrierte Net.Data-Cache-Funktionen	Sie können den Inhalt eines Net.Data-Cache mit den integrierten Net.Data-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 <i>Net.Data Reference</i> .

---

## 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, das die Caching-Funktion enthält.
- Konfigurieren Sie den Cache-Manager. Siehe „Konfigurieren des Cache-Managers und der Net.Data-Caches“ auf Seite 139.
- Legen Sie fest, wie Sie die Net.Data-Anwendung einsetzen wollen.
- Überprüfen Sie die verschiedenen Net.Data-Cache-Protokolle, um zu ermitteln, ob Verbesserungen in der Verwendung des Cache und seiner Konfiguration vorgenommen werden müssen.

## Cache-Fehler

Der Cache-Manager speichert keine 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 zwischengespeichert, wenn:

- Net.Data einen Fehler feststellt, Net.Data die Makrodatei jedoch aufgrund einer Anweisung CONTINUE in einem Nachrichtenblock weiter ausführen muß und normal beendet
- Fehler außerhalb des Net.Data-Fehlerbestimmungsbereichs auftreten, 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 hinsichtlich aller Transaktionen für alle Caches an und wird zur Fehlerbehebung und Problemanalyse bereitgestellt.

Nachrichten werden standardmäßig über die Konsole angezeigt.

#### **Syntax:**

`log=pfad`

Dabei ist *pfad* 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 verwendet.

Dieser Wert muß mit der Anschlußnummer übereinstimmen, die in der Konfigurationsvariablen `CACHE_PORT` in der Net.Data-Initialisierungsdatei angegeben 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=anschlußnummer`

Dabei ist *anschlußnummer* 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=sekunden`

Dabei ist *sekunden* 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 Generierung der Protokolldatei verwaltet (vorhandene `.old`-Dateien werden überschrieben).

**Syntax:**

`wrap-log=yes|no`

Dabei gilt folgendes:

**yes**     Gibt Sie, daß Protokollumlauf verwendet werden soll.

**no**     Gibt Sie, 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=byte`

Dabei ist *byte* 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 *definitionen\_der\_ablaufverfolgungsmarkierungen* aufgenommen werden. Für sie gibt es keine Einstellungen.

Standardmäßig werden nur die Nachrichten beim Start und Stopp des Cache-Managers protokolliert.



**Syntax:**

`trace-flags=definitionen_der_ablaufverfolgungsmarkierungen`

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 Cache

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=pfadname`

Dabei gilt folgendes:

*pfadname* Ist der absolute oder relative Name des Pfads und Verzeichnisses, in dem die Cache-Seiten gespeichert werden.

2. Geben Sie an, ob der aktuelle Cache aktiv sein soll, wenn der Cache-Manager gestartet wird.

Nicht erforderlich. Der Standardwert ist **yes**. Wenn **no** gesetzt wird, wird der Cache im Cache-Manager definiert, jedoch nicht aktiviert. Er kann später mit dem Befehl **cacheadm** aktiviert werden.

**Syntax:**

`caching=yes|no`

Dabei gilt folgendes:

- yes**      Gibt an, daß der Cache aktiv sein soll, wenn der Cache-Manager gestartet wird.
- no**        Gibt an, daß der Cache nicht aktiv sein soll, wenn der Cache-Manager gestartet wird.

3. Geben Sie den maximalen Speicherbereich an, der im Dateisystem durch Seiten im aktuellen Cache belegt werden soll. Wenn der maximale Speicherbereich überschritten wird, löscht der Cache-Manager angefangen bei der ältesten Seite genügend Seiten, um den vom Cache belegten Gesamtspeicherbereich entsprechend zu begrenzen. Sie können das automatische Löschen von Einträgen inaktivieren, indem Sie diesen Wert erhöhen. Wenn jedoch der physische Dateisystemspeicherbereich überschritten wird, schlagen Versuche, dem Cache neue Seiten hinzuzufügen, fehl.

Nicht erforderlich. Der Standardwert ist 0 (kein Caching auf dem Datenträger).

**Syntax:**

`fssize=nnB|nnKB|nnM`

Dabei gilt folgendes:

- nnB**        Ist die Anzahl der Byte, zum Beispiel 5000B.
- nnKB**      Ist die Anzahl der Kilobyte, zum Beispiel 640KB.
- nnMB**      Ist die Anzahl der Megabyte, zum Beispiel 30MB.

4. Geben Sie den maximalen Speicherbereich an, der von allen Seiten in diesem Cache belegt werden soll. Wenn der maximale Speicherbereich überschritten wird, löscht der Cache-Manager angefangen bei der ältesten Seite genügend Seiten, um den vom Cache belegten Gesamtspeicherbereich entsprechend zu begrenzen. Sie können das automatische Löschen von Seiten inaktivieren, indem Sie diesen Wert erhöhen. Wenn jedoch der Prozeß **cachemgrd** zu viel Hauptspeicher in Anspruch nimmt, wird er eventuell vom Betriebssystem beendet.

Nicht erforderlich. Der Standardwert ist 1MB.

**Syntax:**

`mem-size=nnB|nnKB|nnMB`

Dabei gilt folgendes:

- nnB**        Ist die Anzahl der Byte, zum Beispiel 5000B.
- nnKB**      Ist die Anzahl der Kilobyte, zum Beispiel 640KB.
- nnMB**      Ist die Anzahl der 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 dem Datenträger) 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=zeitspanne

Dabei gilt folgendes:

**nnS** Ist die Anzahl der Sekunden, zum Beispiel 600S.

**nnM** Ist die Anzahl der Minuten, zum Beispiel 20M.

**nnH** Ist die Anzahl der Stunden, zum Beispiel 30H.

6. Geben Sie an, ob Cache-Seiten als abgelaufen markiert und eine Überprüfung der Gültigkeitsdauer ausgeführt werden soll.

Nicht erforderlich. Der Standardwert ist **yes** mit einer Standardgültigkeitsdauer von 60 Sekunden. Dieser Wert kann auch auf eine Zeitspanne gesetzt werden, indem Sie den Wert **yes** angeben und eine maximale Zeitspanne für ein im Cache gespeichertes Element festlegen. Wenn **no** gesetzt wird, werden Cache-Seiten nie als abgelaufen markiert, und es wird keine Überprüfung der Gültigkeitsdauer ausgeführt.

**Syntax:**

check-expiration=yes|nnS|nnM|nnH|no

Dabei gilt folgendes:

**yes** Gibt an, daß der Cache-Manager eine Überprüfung der Gültigkeitsdauer ausführt und daß Cache-Seiten als abgelaufen markiert werden.

**nnS** Ist die Anzahl der Sekunden, zum Beispiel 600S.

**nnM** Ist die Anzahl der Minuten, zum Beispiel 20M.

**nnH** Ist die Anzahl der Stunden, zum Beispiel 30H.

**no** Gibt an, daß der Cache-Manager keine Überprüfung der Gültigkeitsdauer ausführt und daß Cache-Seiten nicht als abgelaufen markiert werden.

7. Geben Sie den maximalen Speicherbereich an, den eine zwischengespeicherte Seite im Hauptspeicher-Cache belegen kann. Wenn eine Seite für den Hauptspeicher zu groß ist, wird der Datei-Cache überprüft. Wenn genügend Speicherbereich vorhanden ist, speichert der Cache-Manager die Cache-Seite im Datei-Cache. Wenn die Seite nicht in den Datei-Cache paßt, schlägt der Caching-Versuch fehl. Wenn die Seite kleiner als der Wert für **datum\_memory\_limit (cacheobj-memory-limit)** ist, der Cache jedoch nicht über genügend Speicherbereich verfügt, werden die ältesten Cache-Seiten aus dem Hauptspeicher-Cache gelöscht, um Speicher für die neue Seite freizugeben.

Nicht erforderlich. Der Standardwert ist 1KB.

**Syntax:**

datum-memory-limit (cacheobj-memory-limit)=nnB|nnKB|nnMB

Dabei gilt folgendes:

**nnB** Ist die Anzahl der Byte, zum Beispiel 5000B.

**nnKB** Ist die Anzahl der Kilobyte, zum Beispiel 640KB.

**nnMB** Ist die Anzahl der Megabyte, zum Beispiel 30MB.

8. Geben Sie den maximalen Speicherbereich an, den eine zwischengespeicherte Seite im Datei-Cache belegen kann. Wenn die Seite kleiner als der Wert für **datum\_disk\_limit** ist, im Datei-Cache jedoch kein Speicherbereich frei ist, werden die ältesten Cache-Seiten aus dem Datei-Cache gelöscht, um Speicher für die neue Seite freizugeben.

Nicht erforderlich; der Standardwert ist 1KB.

**Syntax:**

`datum-disk-limit (cacheobj-space-limit)=nnB|nnKB|nnMB`

Dabei gilt folgendes:

**nnB** Ist die Anzahl der Byte, zum Beispiel 5000B.

**nnKB** Ist die Anzahl der Kilobyte, zum Beispiel 640KB.

**nnMB** Ist die Anzahl der Megabyte, zum Beispiel 30MB.

9. Geben Sie die Zeitspanne zwischen der Erstellung von Datensätzen mit Statistikdaten an. Wenn 0 gesetzt wird, werden keine Datensätze mit Statistikdaten geschrieben.

Nicht erforderlich. Der Standardwert ist 0 (keine Statistikdaten).

**Syntax:**

`stat-interval = nnS|nnM|nnH`

Dabei gilt folgendes:

**nnS** Ist die Anzahl der Sekunden, zum Beispiel 600S.

**nnM** Ist die Anzahl der Minuten, zum Beispiel 1M.

**nnH** Ist die Anzahl der 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=pathname`

Dabei ist *pathname* der Pfad und Name der Statistikprotokollierungsdatei.

11. Geben Sie an, ob Zähler für Statistikdaten bei jedem Schreibvorgang 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=datname`

Dabei ist *datname* 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 eingeschaltet werden soll. Dieser Parameter wird ignoriert, außer wenn über den Parameter **tran-log** eine gültige die 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 Dateityp `.old` wird zugewiesen, und es wird ein neues Protokoll geöffnet. Es wird nur eine Generierung 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=byte`

Dabei ist *byte* 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/cache0.stats
  tran-log=/u/cached/logs/cache0.log
  tran-logging=yes
  wrap-tran-log=yes
  tran-log-size=100k
}
```

---

## Starten / Stoppen des Cache-Managers

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

- „Starten des Cache-Managers“
- „Stoppen des Cache-Managers“ auf Seite 147

## Starten des Cache-Managers

Der Cache-Manager-Dämon wird mit dem Befehl `cachemgrd` gestartet.

### Syntax:

► `cachemgrd -c konfigurationsdatei` ◀

### Parameter:

#### **cachemgrd**

Das Befehlsschlüsselwort

#### *konfigurationsdatei*

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`.

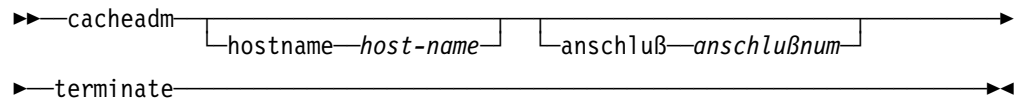
### Beispiel:

```
cachemgrd -c prkonfig.cfg
```

## Stoppen des Cache-Managers

Der Cache-Manager wird mit dem Befehl `cacheadm` gestoppt.

### Syntax:



### Parameter:

**cacheadm** Das Befehlsschlüsselwort

**host-name** 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.

**anschlußnum** 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 Speichern 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. 24 gezeigt.

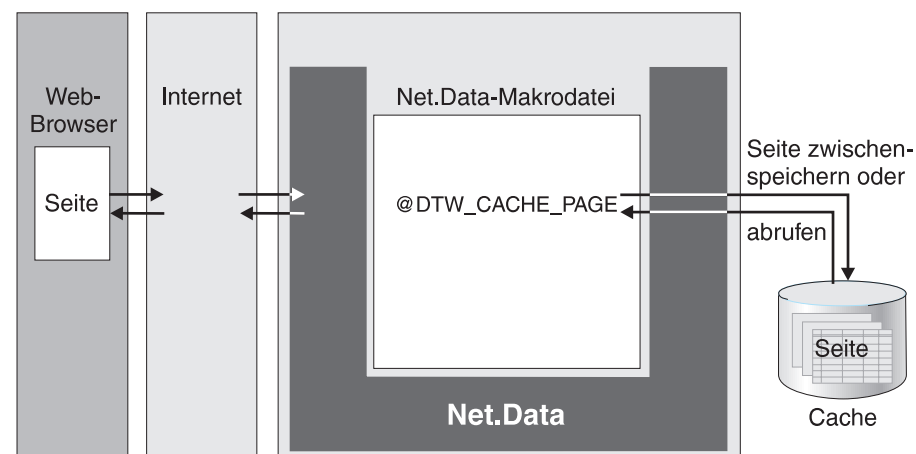


Abbildung 24. Funktion `DTW_CACHE_PAGE` leitet Caching ein

## 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 von 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 angegeben 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", id_der_zwischengespeicherten_Seite, "alter", 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.

#### **id\_der\_zwischengespeicherten\_Seite**

Eine Zeichenfolge mit einer Kennung, mit der die zwischengespeicherte Seite in einer nachfolgenden Cache-Anforderung über @DTW\_CACHE\_PAGE lokalisiert wird, zum Beispiel dem URL der Seite.

**alter** 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 *alter* 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 *alter* 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 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 zwischenspeichern, 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 als zwischenspeicherungswürdig betrachtet, wenn die Datenbank-tabelle 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.

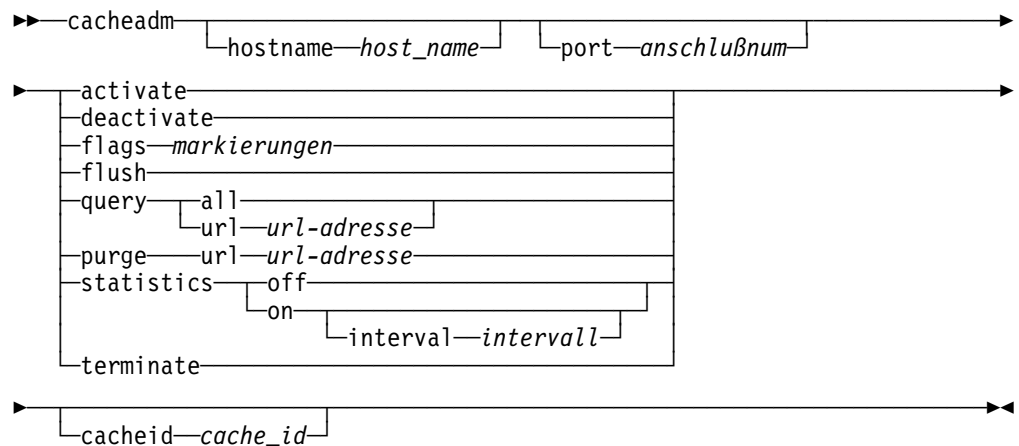
## Der Befehl CACHEADM

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 Zeichenfolge, 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 sind 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.

### **host\_name**

Gibt den Namen der Maschine an, auf der der Cache aktiv ist, sofern es sich nicht um die Maschine handelt, auf der der Befehl `cacheadm` abgesetzt wird, zum Beispiel: `meinhost`.

### **anschlußnum**

Gibt die Cache-Anschlußnummer an, sofern es sich nicht um die Nummer vom Standardwert (7175) handelt. Diese Nummer muß im System eindeutig sein.

### **purge**

Gibt eine bestimmte, aus dem Cache zu löschende Seite an. Wenn *url\_adresse* angegeben wird, löscht der Cache-Manager die Seite mit einem mit *url\_adresse* 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\_adresse* 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 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\_adresse**

Die URL-Adresse (URL - Universal Relative Location), die die Speicherposition der Datei auf dem Web-Server angibt, zum Beispiel:  
<http://www.ibm.com/meinverz/seite1>.

---

## **Das Cache-Protokoll**

Mehrere Arten von Statistikdaten für interne Operationen werden aufbewahrt und wahlweise 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 154

## **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/tt/jj hh:mm:ss id Initialization: interval n seconds*

Dabei gilt folgendes:

<i>mm/tt/jj</i>	Monat, Tag und Jahr des Beginns der Sammlung von Statistikdaten
<i>hh:mm:ss</i>	Stunde, Minute und Sekunde des Beginns der Sammlung von Statistikdaten
<i>id</i>	Name des dem Datensatz zugeordneten Cache
<i>n</i>	Sammlungsintervall

- Beendigungsdatensätze dokumentieren die Beendigung der Sammlung von Statistikdaten für einen bestimmten Cache. Diese Datensätze haben folgendes Format:

*mm/tt/jj hh:mm:ss id Termination*

Dabei gilt folgendes:

<i>mm/tt/jj</i>	Monat, Tag und Jahr des Endes der Sammlung von Statistikdaten
<i>hh:mm:ss</i>	Stunde, Minute und Sekunde des Endes der Sammlung von Statistikdaten
<i>id</i>	Name des dem Datensatz zugeordneten Cache

- Datensätze mit Statistikdaten sind eine durch Leerzeichen begrenzte Gruppe von Nummern, die die Aktivität im Cache anzeigen. Diese Datensätze haben folgendes Format:

*mm/tt/jj hh:mm:ss id statistikdaten*

Dabei gilt folgendes:

<i>mm/tt/jj</i>	Monat, Tag und Jahr der Erstellung der Sammlung von Statistikdaten
<i>hh:mm:ss</i>	Stunde, Minute und Sekunde der Erstellung der Sammlung von Statistikdaten
<i>id</i>	Name des dem Datensatz zugeordneten Cache

*<statistikdaten>*

Liste der durch Leerzeichen begrenzten und für diesen Cache gesammelten Statistikdaten wie in Tabelle 7 auf Seite 155 dargestellt:

Tabelle 7. 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 Cache belegter Speicher	Nein
14	Datenträgerbelegung	Von Objekten im Datenträgerbereich des Cache belegter Speicher	Nein
15	Verfügbarer Hauptspeicher	Für Objekte im Hauptspeicherbereich des Cache weiterhin verfügbarer Speicherbereich	Nein
16	Verfügbarer Plattenspeicherplatz	Für Objekte im Datenträgerbereich des Cache weiterhin verfügbarer Plattenspeicherplatz	Nein
17	Anzahl Hauptspeicherobjekte	Anzahl der Objekte im Hauptspeicherbereich des Cache	Nein
18	Anzahl Dateiobjekte	Anzahl der Objekte im Datenträgerbereich des Cache	Nein
19	Anzahl Sitzungen	Anzahl der momentan für den Cache aktiven Sitzungen	Nein





---

## Optimieren der Leistung

Das Optimieren der Leistung ist ein wichtiger Bestandteil der Systemoptimierung. In diesem Kapitel werden Strategien zum Erhöhen der Leistung diskutiert. Folgende Themen werden behandelt:

- „Optimieren der Leistung mit den Web-Server-APIs“
- „Optimieren der Leistung mit FastCGI“ auf Seite 159
- „Optimieren der Leistung durch Verbindungsverwaltung“ auf Seite 160
- „Optimieren der Leistung mit Net.Data-Caching“ auf Seite 164
- „Net.Data-Fehlerprotokollierung: Überlegungen zur Leistung“ auf Seite 165
- „Optimieren mathematischer Funktionen“ auf Seite 165

---

### Optimieren der Leistung mit den Web-Server-APIs

Sie können die Leistung optimieren, indem Sie Net.Data mit Web-Server-APIs anstelle von CGI aufrufen. Wenn Net.Data im Web-Server-API-Modus ausgeführt wird, wird Net.Data als Thread des Web-Server-Prozesses ausgeführt. Dadurch entfällt der beim Aufrufen von Net.Data als CGI-Prozeß entstehende Systemaufwand. Bei Verwendung von Web-Server-APIs wird Net.Data in Form von mehreren Threads des Server-Prozesses ausgeführt.

Der Web-Server ruft Net.Data standardmäßig als CGI-Programm auf, wobei jeder Net.Data-Prozeß in einem separaten Prozeß ausgeführt wird. Net.Data stellt zur Leistungsoptimierung Konfigurationsoptionen für die Web-Server-APIs bereit.

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 Sever

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

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

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 35.

**Hinweis:** Die Verwendung der Web-Server-APIs optimiert die Leistung ohne Anwendungsisolation. Da Net.Data in einem Multi-Thread-Modus ausgeführt wird, können Fehler bei benutzerdefinierten Sprachumgebungen, inkorrekten Aufrufen oder gar Datenbankausfällen zu Problemen mit dem Web-Server führen. Dies kann möglicherweise zum Absturz des Web-Servers führen. Berücksichtigen Sie bei Wahl zwischen einer der Web-Server-APIs oder CGI bzw. FastCGI, ob für Ihre Anwendung die Leistung oder die Anwendungsisolation wichtiger ist.

### 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 Sprachumgebungen für Oracle und Sybase eine Direktverbindung erforderlich.

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

#### Für ICAPI und GWAPI:

##### Syntax:

`http://server_name/CGI-BIN/db2www/makroname/html_block`

##### Parameter:

*server\_name*

Name des Servers

*makroname*

Der relative Pfad Ihrer Makrodatei im Verzeichnis, das mit MACRO\_PATH angegeben wird.

*html\_block*

Name des zu verarbeitenden HTML-Blocks in der Makrodatei

##### Beispiel:

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

#### Für ISAPI:

##### Syntax:

`http://server_name/server_HTML_stammverzeichnis/dll_name/  
makroname/  
html_block`

##### Parameter:

*server\_name*

Name des Servers

*server\_HTML\_stammverzeichnis*

Name des HTML-Stammverzeichnisses des Web-Servers

*dll\_name*

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

*makroname*

Der relative Pfad Ihrer Makrodatei im Verzeichnis, das mit MACRO\_PATH angegeben wird.

*html\_block*

Name des zu verarbeitenden HTML-Blocks in der Makrodatei

**Beispiel:**

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

**Für NSAPI:**

**Syntax:**

`http://server_name/makroname/html_block`

**Parameter:**

*server\_name*

Name des Servers

*makroname*

Der relative Pfad Ihrer Makrodatei im 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 35.

*html\_block*

Name des HTML-Blocks in der zu verarbeitenden Makrodatei

**Beispiel:**

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

---

## Optimieren der Leistung mit 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 31.

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. Dadurch 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=anzahl`
  - Bei ISC: `NumProcess=anzahl`

Dabei gibt *anzahl* die Anzahl der Prozesse an.

---

## Optimieren der Leistung durch Verbindungsverwaltung

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 Prozeß 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“ auf Seite 161
- „Vorteile der Direktverbindung“ auf Seite 162
- „Einsatzmöglichkeiten der Direktverbindung“ auf Seite 162
- „Starten von Connection Manager“ auf Seite 162
- „Verarbeitungsablauf für Net.Data und Direktverbindung“ auf Seite 163

## 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 einem Connection Manager und Cliettes. *Cliettes* sind vom 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 CLLETTE angeben. Jeder Cliette-Typ ist für die Bearbeitung einer spezifischen Sprachumgebungsfunktion konzipiert, z. B. die DB2-Cliette, die die Verbindung zur DB2-Datenbank herstellt und Operationen zur Ausführung von SQL-Aufrufen definiert, bevor Net.Data-Makros von Net.Data verarbeitet werden. Der Name der ausführbaren Datei wird in der Konfigurationsdatei für die Direktverbindung angegeben (dtwcm.cnf). Abb. 25 zeigt die Interaktion zwischen der Direktverbindung, der Makrodatei und den Sprachumgebungen.

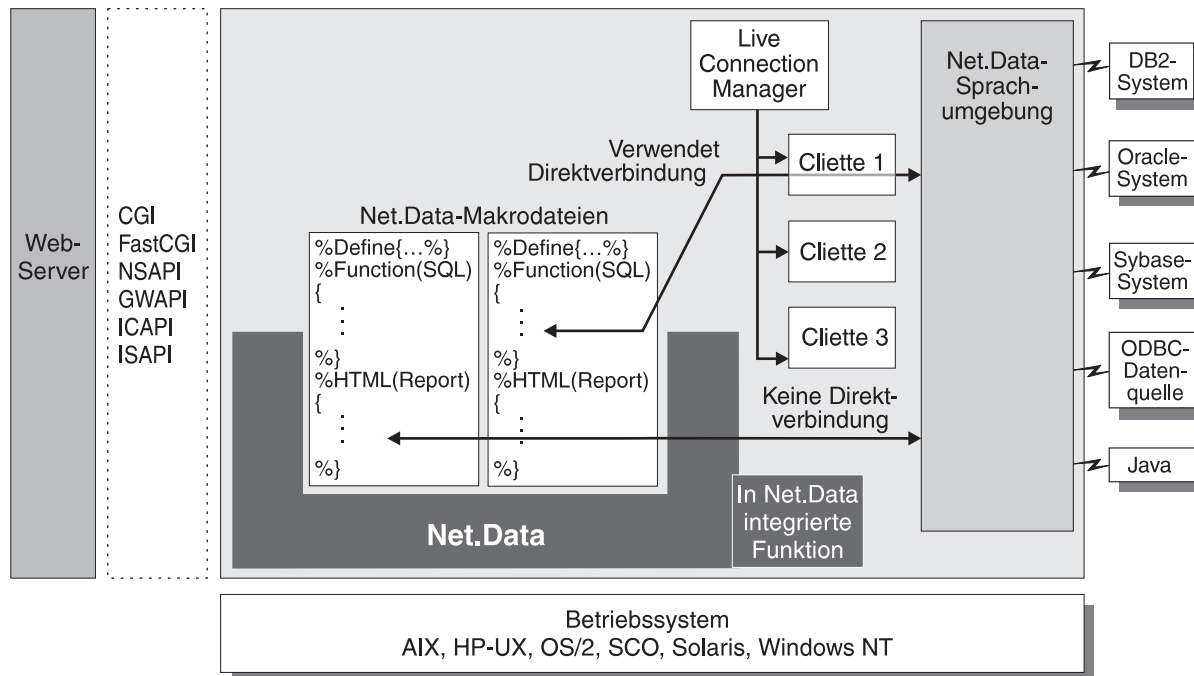


Abbildung 25. Direktverbindung mit Cliettes

In den folgenden Abschnitten wird die Direktverbindung näher beschrieben. Informationen zum Konfigurieren der Direktverbindung finden Sie in „Konfigurieren der Direktverbindung“ auf Seite 25.

## 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 CGI-, 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 25.

**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_verz>\connect\`.
2. Geben Sie `dtwcm` ein.

Dabei steht `<inst_verz>` 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 erneut zu unterdrücken.

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

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

**Hinweis:** Starten Sie Connection Manager von der Befehlszeile, bevor Sie angeben, daß Connection Manager automatisch gestartet werden soll, um sicherzustellen, daß die Konfigurationsdatei für Direktverbindung 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 Client aus der Initialisierungsdatei erforderlich ist. Bei DB2 ist der Client-Typ häufig ein auf dem DB2-Datenbanknamen basierender Name wie DTW\_SQL:CELDIAL.
4. Net.Data fordert von Connection Manager eine Client dieses Typs an.

5. Connection Manager sucht nach verfügbaren Clientes dieses Typs. Wenn keine verfügbar sind, stellt Connection Manager die Anforderung in eine Warteschlange und verarbeitet sie, wenn der richtige Client-Typ verfügbar ist.
6. Wenn eine Client zur Verfügung steht, teilt Connection Manager Net.Data mit, wie mit der Client kommuniziert werden muß.
7. Net.Data fordert die Client auf, die Funktion zu verarbeiten.
8. Dieser Prozeß wird ab Schritt 3 wiederholt, bis die Verarbeitung des Net.Data-Makros abgeschlossen ist.
9. Alle Clientes werden freigegeben.

Wenn eine Client 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.

---

## Optimieren der Leistung mit Net.Data-Caching

Sie können die Leistung Ihres Systems optimieren, indem Sie häufig angeforderte Net.Data-Web-Seiten in einem Cache zwischenspeichern. Durch das Abrufen einer Web-Seite aus dem Cache können Sie die benötigte Verarbeitungszeit reduzieren, weil das Net.Data-Makro nicht interpretiert wird und keine Net.Data-Sprachumgebungsaufrufe ausgeführt werden. Wenn die Net.Data-Web-Seiten von den in Datenbanken oder Dateien gespeicherten Informationen generiert werden, können die Datenträger-E/A-Zugriffe wahrscheinlich ebenfalls reduziert werden. Dadurch kann sowohl die Durchsatz- als auch die Antwortzeit verringert werden.

Net.Data-Caching bietet die Multi-Thread-Verwaltung mehrerer Caches in einem einzigen Prozeß. Caching eignet sich für eine Netzwerkumgebung, in der ein Cache von mehreren Prozessen auf mehreren Maschinen gemeinsam benutzt wird. Net.Data-Caching verwendet zur Verwaltung von Net.Data-Caches den Cache-Manager.

Je nach der Größe der definierten Caches und der Speicherbelegung auf Ihrem System ist eventuell zusätzlicher Hauptspeicher erforderlich, um zusätzliche Umlagerung zu vermeiden.

Informationen zum Definieren und Verwenden von Net.Data-Caching finden Sie in „Net.Data-Caching“ auf Seite 133.



---

## Net.Data-Fehlerprotokollierung: Überlegungen zur Leistung

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 mathematischer Funktionen

Sie können die Leistung Ihrer mathematischen Funktionen mit Hilfe der Konfigurationsvariable `DTW_OPTIMIZE_MATH` optimieren. Diese Variable bietet die Formatierung mathematischer Funktionen entsprechend der Programmiersprache C, wodurch Funktionen schneller verarbeitet werden können. Wenn diese Variable auf `YES` gesetzt ist, führt Net.Data die Formatierung entsprechend der Programmiersprache C durch. Der Standardwert ist `NO`. Informationen zum Konfigurieren dieser Variablen finden Sie in „`DTW_OPTIMIZE_MATH`: Variable für das Optimieren mathematischer Funktionen“ auf Seite 18.

### Syntax:

`DTW_OPTIMIZE_MATH=NO|YES`

**Hinweis:** Net.Data Version 1 formatierte mathematische Funktionen entsprechend der Programmiersprache REXX. Ihre Ausgabe sieht bei Verwendung der Konfigurationsvariablen `DTW_OPTIMIZE_MATH` eventuell anders aus. Abschließende Leerzeichen werden nicht angezeigt. Wenn es für Ihre Anwendung wichtig ist, daß die Ausgabe mit der von Makrodateien von Net.Data Version 1 übereinstimmt, wägen Sie zwischen Konsistenz und Leistung ab.



---

## 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.

Dieser Abschnitt enthält die folgenden Themen zum Protokollieren:

- „Planen der Protokollüberwachung“
- „Steuern der Protokollstufe in den Protokolldateien“ auf Seite 168
- „Arten nicht protokollierter Nachrichten“ auf Seite 168
- „Größe und Archivierung der Protokolldatei“ auf Seite 168
- „Protokolldateiformat“ auf Seite 169

---

### Planen der Protokollüberwachung

Beim Protokollieren von Fehlern müssen Sie die folgenden Punkte einplanen:

- Ermitteln des Plattenspeicherplatzes:

Wenn Fehler protokolliert werden sollen, müssen Sie zusätzlichen Plattenspeicherplatz für die Fehlerprotokolle freigeben.

- Konfigurieren von Net.Data:

Wenn Fehler für das gesamte Net.Data-System protokolliert werden sollen, setzen Sie eine Konfigurationsvariable in Ihrer Net.Data-Initialisierungsdatei auf `DTW_LOG_DIR`.

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 im Abschnitt „`DTW_LOG_DIR`: Variable für Speicherposition des Fehlerprotokolls“ auf Seite 17.

- Schreiben von Net.Data-Makros:

Legen Sie die Protokollstufe mit dem Schlüsselwort `DTW_LOG_LEVEL` in Ihrer Makrodatei fest.

- Ausführen von Net.Data:

Wenn Sie die Fehlerprotokollierung verwenden, können Sie die Fehlerprotokolle und Archivdateien in Ihrem Net.Data-System auf Fehler überprüfen.

- Optimierung:

Berücksichtigen Sie, daß sich die Protokollierung nachteilig auf die Leistung auswirken kann. Informationen zur Leistung finden Sie im Abschnitt „Net.Data-Fehlerprotokollierung: Überlegungen zur Leistung“ auf Seite 165.

---

## Steuern der Protokollstufe in den Protokolldateien

Sie können die Protokollstufe mit dem Schlüsselwort `DTW_LOG_LEVEL` angeben. Definieren Sie dieses Schlüsselwort in der `Net.Data`-Makrodatei. Für dieses Schlüsselwort 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 Nachrichten

`Net.Data` protokolliert die Fehler, die explizit von einem `MESSAGE`-Abschnitt in der Makrodatei bearbeitet werden, nicht.

---

## Größe und Archivierung der Protokolldatei

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.logMMMTTJJJJ_nn` archiviert.

Dabei gilt folgendes:

*MMM*        Der Monat (Jan-Dez)

*TT*         Das Datum

*JJJJ*       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.

---

## Protokolldateiformat

Protokolldateieinträge haben folgendes Format:

*[TT/MMM/JJJJ:HH:MM:SS] [MAKRO] [BLOCK] [PID-NR] [TID-NR]fehlernachricht*

### Parameter:

*TT* Das Datum

*MMM* Der Monat (Jan-Dez)

*JJJJ* Das Jahr

*HH* Die Stunde (00-23)

*MM* Die Minute (00-59)

*SS* Die Anzahl Sekunden (00-59)

*MAKRO* Die Makrodatei, die die Fehlernachricht generierte

*BLOCK* Der Name des HTML-Blocks, der die Fehlernachricht generierte

*PID-NR* 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-NR* Die Nummer der Thread-ID des Prozesses, der die Fehlernachricht generierte. Diese ID ist notwendig, weil mehrere Threads vom gleichen Net.Data-Prozeß in die Protokolldatei schreiben können.

*fehlernachricht* Der Text der Fehlernachricht



---

## Anhang A. 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 und 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 von 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_getNextRow_fp)(); /* dtw_getNextRow 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 Eingangspunkt 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_getNextRow  
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:

- In der Makrodatei 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.



---

## Anhang B. 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. Die generierte Makrodatei verwendet in Ihrer Datenbank gespeicherte Primär- und Fremdschlüsselinformationen, um Ihre Web-Berichte automatisch zu verbinden.

**Forum** Dieser SmartGuide erstellt ein Web-fähiges Diskussionsforum, in dem Sie Nachrichten versenden, anzeigen und beantworten können. Sie können die Forumkategorien und Diskussionsthemen nach Ihren Wünschen anpassen. Die generierte Forumanwendung enthält eine Funktion, mit der Sie die Forumnachrichten nach bestimmten Stichwörtern durchsuchen können. Wahlfrei können Sie in Ihrer Makrodatei Cookie-Funktionen aktivieren, damit sich das Forum an seine Benutzer „erinnert“.

**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 eine Net.Data-Makrodatei, mit der Sie Ihre gespeicherte Prozedur aufrufen können. Nun können Sie diese Makrodatei ändern oder sie 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.

---

## Einführung

Für die Ausführung der SmartGuides und das Generieren der Net.Data-Makrodateien, 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 zu Ihrer Umgebungsvariable CLASSPATH hinzu.

*[Pfad]*NetDataSmartGuides.jar

Dabei steht *[Pfad]* 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 Umgebungsvariable CLASSPATH den UDB-JDBC-Treiber hinzufügen.

Für die Betriebssysteme Windows NT und OS/2 muß der Umgebungsvariablen CLASSPATH die folgende Zeile hinzugefügt werden:

*[Pfad]*NetDataSmartGuides.jar;*[UDBInstallationPfad]*\java\db2java.zip

Für UNIX-Betriebssysteme muß der Umgebungsvariablen CLASSPATH die folgende Zeile hinzugefügt werden:

*[Pfad]*NetDataSmartGuides.jar:*[UDBInstallationPfad]*\java\db2java.zip

Dabei steht *[Pfad]* für den wahlfreien Pfad zur Datei NetDataSmartGuides.jar und *[UDBInstallationPfad]* 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.

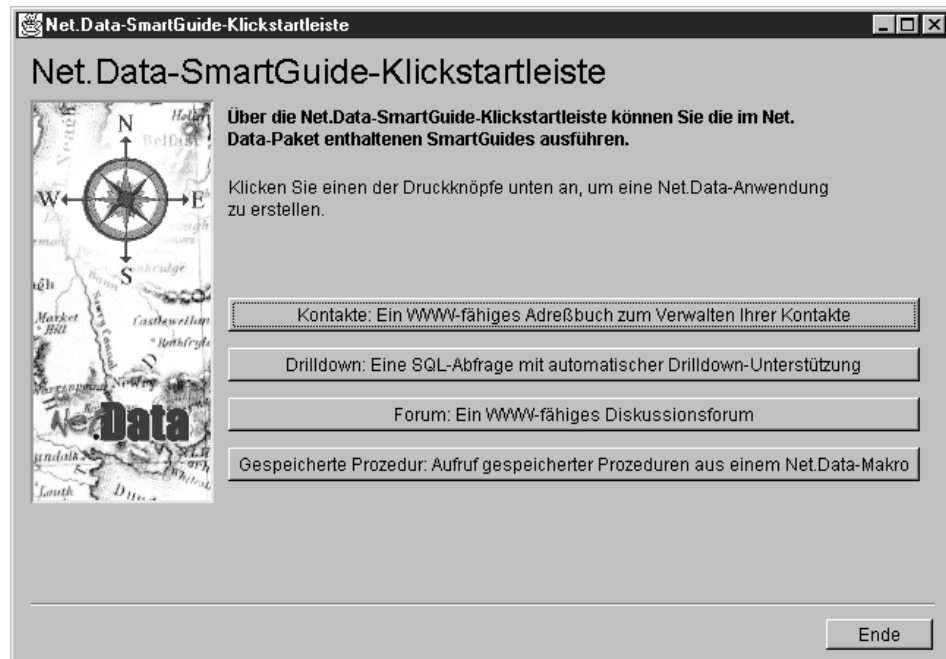


Abbildung 26. Die SmartGuide-Klickstartleiste

### 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. Geben Sie den folgenden Befehl ein, um die SmartGuides auszuführen:

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

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

Nun wird eine Klickstartleiste geöffnet, in der die verfügbaren SmartGuides wie in Abb. 26 auf Seite 175 dargestellt, aufgelistet werden.

2. Wenn Sie die Funktion zum Verbinden der Datenbank der SmartGuides **Drilldown** und **Gespeicherte Prozedur** verwenden wollen, müssen Sie den folgenden Befehl eingeben:

Für die Betriebssysteme Windows NT und OS/2:

```
jre -cp [Pfad]NetDataSmartGuides.jar:[UDBInstallationPfad]  
      \java\db2java.zip TaskGuide LaunchPad.class
```

Für UNIX-Betriebssysteme:

```
jre -cp [Pfad]NetDataSmartGuides.jar:[UDBInstallationPfad]  
      \java\db2java.zip TaskGuide LaunchPad.class
```

Dabei steht *[Pfad]* für den wahlfreien Pfad zur Datei NetDataSmartGuides.jar und *[UDBInstallationPfad]* für den Pfad zu Ihrer UDB-Installation, zum Beispiel C:\SQLLIB.

3. Klicken Sie den Namen des gewünschten SmartGuides an.

## Anhang C. 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 im Abschnitt „Net.Data-Servlets“ auf Seite 121 beschrieben.

Sie können NetObjects Fusion (NOF) zur Integration Ihrer Net.Data-Makrodateien verwenden. NOF bietet bessere Integration in Ihre Web-Site-Verwaltung und eine benutzerfreundliche grafische Benutzeroberfläche.

### Informationen zum Plug-In für NetObjects Fusion

Das Plug-In NetDataServlet.NFX verwendet die Net.Data-Servlets. Dieses NOF-Plug-In unterstützt den Aufruf einer vorhandenen Net.Data-Makrodatei 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 die Net.Data-Makrodatei bzw. -Funktion ausgeführt. Betten Sie mit dem Net.Data-Servlet-Plug-In ein beliebiges Makro- und Funktions-Servlet in eine über NetObjects Fusion verwaltete Web-Site ein. Dies wird in Abb. 27 beschrieben. Das Plug-In stellt viele der grundlegenden Makrodatei- bzw. Funktionsparameter bereit, die standardmäßig so eingestellt sind, daß das Erstellen einer Makrodatei automatisch erfolgt. Sie müssen NOF und die Plug-In-Dateien installieren und konfigurieren, um das Plug-In verwenden zu können.

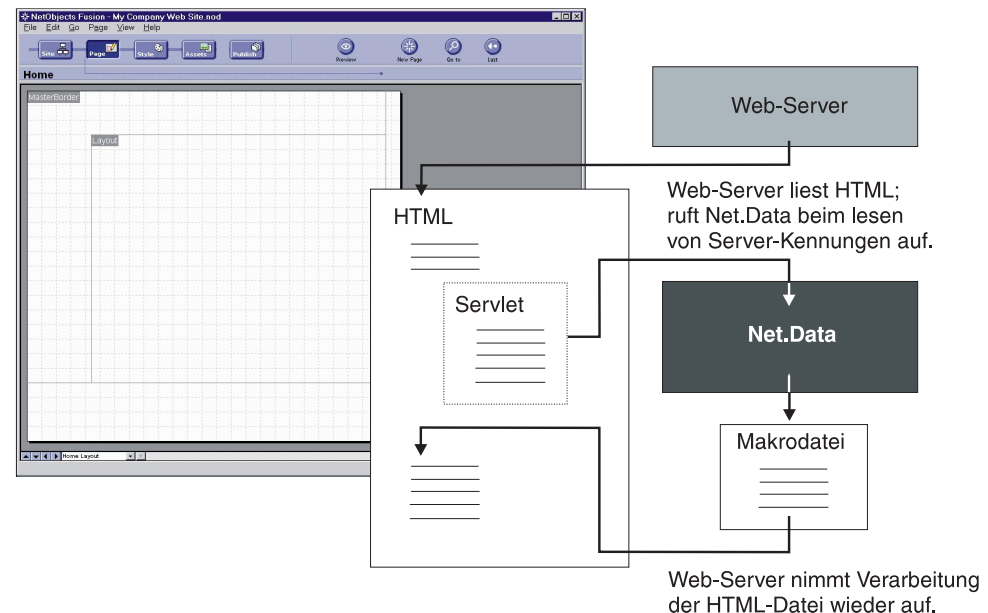


Abbildung 27. Net.Data-Servlets-Plug-ins

---

## Installieren des Plug-In für NetObjects Fusion

### Hardware- und Softwarevoraussetzungen:



Für die NOF-Plug-Ins ist NetObjects Fusion Version 2.0 oder höher erforderlich.

**Installation:** Kopieren Sie <inst\_verz>\fx\NetDataServlet.nfx und <inst\_verz>\fx\NetDataServlet.gif in das Verzeichnis <NetObjects Fusion>\components\.

---

## Konfigurieren des Net.Data-Plug-In für NetObjects Fusion

Sie können die Merkmale des Servlet, 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 Servlet 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 Servlet 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.

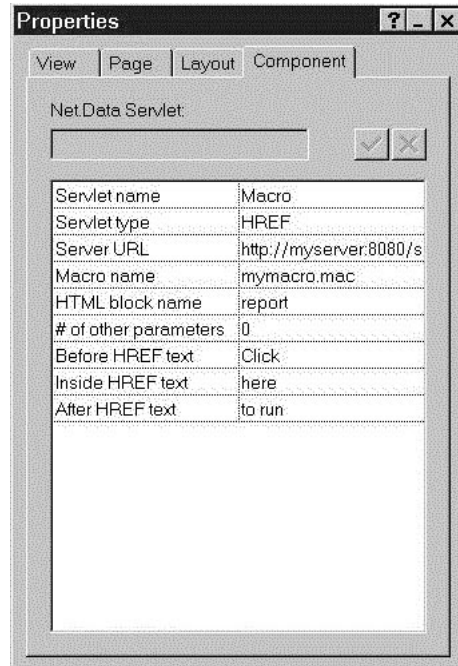


Abbildung 28. Die Net.Data-Servlet-Palette "Properties"

### Merkmale für die Net.Data-Servlets:

Sie können die folgenden Merkmale anpassen:

#### Servlet name (Servlet-Name)

Wählen Sie den Namen des gewünschten Servlet aus: Function oder Macro. Je nach ausgewähltem Servlet-Namen werden verschiedene Merkmale angezeigt.

#### Servlet typ (Servlet-Art)

Wählen Sie die gewünschte Servlet-Art aus: SSI, HREF oder FORM Submit Button. Je nach ausgewählter Servlet-Art werden verschiedene Merkmale angezeigt.

#### Submit label (Übergabebezeichnung)

Wenn Sie die Art FORM Submit Button auswählen, müssen Sie den Text für die Übergabebezeichnung angeben. Ansonsten wird dieses Merkmal nicht angezeigt.

**Server URL (Server-URL)**

Wenn Sie die Servlet-Art HREF auswählen, müssen Sie den Server-URL für den Servlet-fähigen Web-Server angeben. Wenn Sie SSI auswählen, wird dieses Merkmal nicht angezeigt.

**Macro name (Makroname)**

Wenn Sie den Makro-Servlet-Namen auswählen, müssen Sie den Namen einer vorhandenen auszuführenden Net.Data-Makrodatei angeben. Ansonsten wird dieses Merkmal nicht angezeigt.

**HTML block name (HTML-Blockname)**

Wenn Sie den Makro-Servlet-Namen auswählen, müssen Sie den Namen des HTML-Blocks in der auszuführenden Net.Data-Makrodatei angeben. Ansonsten wird dieses Merkmal nicht angezeigt.

**Function type (Funktionsart)**

Wenn Sie den Funktions-Servlet-Namen auswählen, müssen Sie die auszuführende Funktionsart auswählen: Function oder SQL. Ansonsten wird dieses Merkmal nicht angezeigt.

**Language Env (Sprachumgebung)**

Wenn Sie den Funktions-Servlet-Namen auswählen, müssen Sie die zu verwendende Net.Data-Sprachumgebung angeben. Ansonsten wird dieses Merkmal nicht angezeigt.

**Statement (Anweisung)**

Wenn Sie den Funktions-Servlet-Namen auswählen, müssen Sie die auszuführende Anweisung angeben. Ansonsten wird dieses Merkmal nicht angezeigt.

**Database (Datenbank)**

Wenn Sie den Funktions-Servlet-Namen auswählen, müssen Sie den Namen der zu verwendenden Datenbank angeben. Ansonsten wird dieses Merkmal nicht angezeigt.

**# of other parameters (Anzahl anderer Parameter)**

Geben Sie die Anzahl anderer, an Net.Data zu übermittelnder Parameter an (Maximum: 25). Geben Sie für jeden Parameter seinen Namen und Wert ein (wahlfrei).

**Before HREF text (Text vor HREF)**

Wenn Sie ein Servlet der Art HREF auswählen, geben Sie den Text an, der vor dem Text vor dem HTML-Befehl <A HREF> angezeigt werden soll. Ansonsten wird dieses Merkmal nicht angezeigt (wahlfrei).



#### **Inside HREF text (Text innerhalb HREF)**

Wenn Sie ein Servlet der Art HREF auswählen, geben Sie den Text an, der innerhalb des HTML-Befehls <A HREF> angezeigt werden soll. Ansonsten wird dieses Merkmal nicht angezeigt (wahlfrei).

#### **After HREF text (Text nach HREF)**

Wenn Sie ein Servlet der Art HREF auswählen, geben Sie den Text an, der nach dem Text nach dem HTML-Befehl <A HREF> angezeigt werden soll. Ansonsten wird dieses Merkmal nicht angezeigt (wahlfrei).

#### **SQL Reminder! (SQL-Hinweis!)**

Wenn Sie ein Servlet der Art HREF auswählen und eine Funktion der Art SQL angeben, wird eine Nachricht angezeigt, die Sie daran erinnert, daß die HREF-SQL-Anweisung für jedes Leerzeichen ( ) ein Pluszeichen (+) enthalten muß. Dieser Text kann nicht geändert werden, und er wird auch dem Veröffentlichen der Web-Seite nicht angezeigt. Ansonsten wird dieses Merkmal nicht angezeigt.

- b. Klicken Sie nach dem Definieren der Merkmale für Ihre Seite den Druckknopf **Publish** an, um die Web-Seiten mit dem NOF-Plug-In für Net.Data-Servlets zu erstellen und zu veröffentlichen.

**Anmerkung:** Wenn Sie die Servlet-Art SSI auswählen, muß die Dateierweiterung Ihrer Web-Seite .shtml sein. Sie können dies als den Standardwert für Ihre Web-Seite von der NOF-Palette **Properties** einstellen, indem Sie die Notizbuchindexzunge **Page** auswählen, den Druckknopf **Custom names** anklicken und im Feld **Extension Type** .shtml eingeben.

---

## Veröffentlichen von Servlets mit dem NOF-Plug-In

Klicken Sie nach dem Einstellen der Merkmale für Ihre Seite den Druckknopf **Publish** an, um die Web-Seiten mit dem Plug-In zu erstellen und zu veröffentlichen.

---

## Anhang D. Net.Data-Beispielmakro

| Diese Beispielmakroanwendung zeigt eine Liste von Mitarbeiternamen an, aus der  
| der Anwendungsbenutzer zusätzliche Informationen zu einem bestimmten Mitar-  
| beiter abrufen kann, indem er den Namen des Mitarbeiters in der Liste auswählt.  
| Das Makro verwendet die SQL-Sprachumgebung, um die Tabelle EMPLOYEE nach  
| den Mitarbeiternamen und Informationen zu einem bestimmten Mitarbeiter abzu-  
| fragen.

```

%{***** Sample Macro *****)
*   FileName = sqlsamp1.d2w
*   Description:
*       This Net.Data macro file queries...
*       - The EMPLOYEE table to create a selection list of
*         employees for display at a browser
*       - The EMPLOYEE table to obtain additional information
*         about an individual employee
*
*****%}
%{*****
*   Include for global DEFINES -
*****%}
%INCLUDE "sqlsamp1.hti"
%{*****
*   Function: queryDB           Language Environment: SQL
*   Description: Queries the table designated by the variable myTable *
*   and creates a selection list from the result. The value of the variable*
*   myTable is specified in the include file sqlsamp1.hti.
*****%}
%FUNCTION(DTW_SQL) queryDB() {
    SELECT * FROM $(myTable)
%MESSAGE {
    -204: {<p><b>ERROR -204: Table $(myTable) not found. </b>
    <p>Be sure the correct include file is being used.</b>
    %} : exit
    +default: "WARNING $(RETURN_CODE)" : continue
    -default: "Unexpected ERROR $(RETURN_CODE)" : exit
%}

%REPORT {
<select name=emp_name>
    %ROW{
<option>$(V2)
%}
</select>
%}
%}

%{*****
*   Function: fname           Language Environment: SQL
*   Description: Queries the table designated by the variable myTable*
*   additional information about the employee identified the *
*   by the variable emp_name.
*****%}
%FUNCTION(DTW_SQL) fname(){
SELECT EMPNME, PHONENO, JOB FROM $(myTable) WHERE EMPNME='$(emp_name)'
%MESSAGE {
    -204: "Error -204: Table not found "
    -104: "Error -104: Syntax error"
    100: "Warning 100: No records" : continue
    +default: "Warning $(RETURN_CODE)" : continue
    -default: "Unexpected SQL error" : exit
%}
%}
%}

```

```

%{*****
*   HTML block: INPUT                Title: Dynamic Query Selection      *
*                                                                           *
*   Description: Queries the EMPLOYEE table to create a selection list of*
*               the employees for display at the browser                *
*****%}
%HTML(INPUT){
<html>
<head>
<title>Generate Employee Selection List</title>
</head>
<body>
<h3>$(exampleTitle)</h3>
<p>This example queries a table and uses the result to create
a selection list using a <em>%REPORT</em> block.
<hr>
<form method="post" action="report">
@queryDB(<input type="submit" value="Select Employee">
</form>
<hr>
</body>
</html>
%}

```

```

%{*****
*   HTML block:   REPORT
*   Description:  Queries the EMPLOYEE table to obtain additional information
*               about an individual employee
*****%}
%HTML(REPORT){
<html>
<head>
<title>Obtain Employee Information</title>
</head>
<body>
<h3>You selected employee name = $(emp_name)</h3>
<p>Here is the information for that employee:
<PRE>
@fname()
</PRE>
<hr><a href="input">Return to previous page</a>
</body>
</html>
%}

%{      End of Net.Data macro 1 %}
=====
%{***** Include File *****
*   FileName = sqlsamp1.hti
*   Description:
*   This include file provides global DEFINES for the sqlsamp1.d2w
*   Net.Data macro.
*****%}
%define {
    emp_name    = ""
    reposition  = sign
    exampleTitle = "Sample Macro"
    myTable     = "MRZ.EMPLOYEE"
%}

%{      End of include file %}

```

---

## Anhang E. Bemerkungen

Die vorliegenden Informationen wurden für Produkte und Services entwickelt, die auf dem deutschen Markt angeboten werden. Möglicherweise bietet IBM die in dieser Dokumentation beschriebenen Produkte, Services oder Funktionen in anderen Ländern nicht an. Informationen über die gegenwärtig im jeweiligen Land verfügbaren Produkte und Services sind beim IBM Ansprechpartner erhältlich. Hinweise auf IBM Lizenzprogramme oder andere IBM Produkte bedeuten nicht, daß nur Programme, Produkte oder Dienstleistungen von IBM verwendet werden können. Anstelle der IBM Produkte, Programme oder Dienstleistungen können auch andere ihnen äquivalente Produkte, Programme oder Dienstleistungen verwendet werden, solange diese keine gewerblichen Schutzrechte der IBM verletzen. Die Verantwortung für den Betrieb von Fremdprodukten, Fremdprogrammen und Fremdservices liegt beim Kunden.

Für in diesem Handbuch beschriebene Erzeugnisse und Verfahren kann es IBM Patente oder Patentanmeldungen geben. Mit der Auslieferung dieses Handbuchs ist keine Lizenzierung dieser Patente verbunden. Lizenzanfragen sind schriftlich an IBM Europe, Director of Licensing, 92066 Paris La Defense Cedex, France, zu richten. Anfragen an obige Adresse müssen auf englisch formuliert werden.

Trotz sorgfältiger Bearbeitung können technische Ungenauigkeiten oder Druckfehler in dieser Veröffentlichung nicht ausgeschlossen werden. Die Angaben in diesem Handbuch werden in regelmäßigen Zeitabständen aktualisiert. Die Änderungen werden in Überarbeitungen oder in Technical News Letters (TNLs) bekanntgegeben. IBM kann jederzeit ohne weitere Mitteilung Verbesserungen und/oder Änderungen an den in dieser Veröffentlichung beschriebenen Produkten und/oder Programmen vornehmen.

Verweise in dieser Veröffentlichung auf Web-Sites anderer Anbieter dienen lediglich als Benutzerinformationen und stellen keinerlei Billigung des Inhalts dieser Web-Sites dar. Das über diese Web-Sites verfügbare Material ist nicht Bestandteil des Materials für dieses IBM Produkt. Die Verwendung dieser Web-Sites geschieht auf eigene Verantwortung.

Lizenznehmer des Programms, die Informationen zu diesem Produkt wünschen mit der Zielsetzung: (i) den Austausch von Informationen zwischen unabhängigen, erstellten Programmen und anderen Programmen (einschließlich des vorliegenden Programms) sowie (ii) die gemeinsame Nutzung der ausgetauschten Informationen zu ermöglichen, wenden sich an folgende Adresse:

IBM Corporation  
\_W92/H3\_  
\_555 Bailey Avenue\_  
\_P.O. Box 49023\_  
\_San Jose, CA 95161-9023\_  
\_U.S.A.

Die Bereitstellung dieser Informationen kann unter Umständen von bestimmten Bedingungen - in einigen Fällen auch von der Zahlung einer Gebühr - abhängig sein.

Die Lieferung des im Handbuch aufgeführten Lizenzprogramms sowie des zugehörigen Lizenzmaterials erfolgt im Rahmen der IBM Kundenvereinbarung oder einer äquivalenten Vereinbarung.

Informationen über Produkte anderer Hersteller als IBM wurden von den Herstellern dieser Produkte zur Verfügung gestellt, bzw. aus von ihnen veröffentlichten Ankündigungen oder anderen öffentlich zugänglichen Quellen entnommen. IBM hat diese Produkte nicht getestet und übernimmt im Hinblick auf Produkte anderer Hersteller keine Verantwortung für einwandfreie Funktion, Kompatibilität oder andere Ansprüche. Fragen hinsichtlich des Leistungsspektrums von Produkten anderer Hersteller als IBM sind an den jeweiligen Hersteller des Produkts zu richten.

Die oben genannten Erklärungen bezüglich der Produktstrategien und Absichtserklärungen von IBM stellen die gegenwärtige Absicht der IBM dar, unterliegen Änderungen oder können zurückgenommen werden, und repräsentieren nur die Ziele der IBM.

Diese Veröffentlichung enthält Beispiele für Daten und Berichte des alltäglichen Geschäftsablaufes. Sie sollen nur die Funktionen des Lizenzprogrammes illustrieren; sie können Namen von Personen, Firmen, Marken oder Produkten enthalten. Alle diese Namen sind frei erfunden; Ähnlichkeiten mit tatsächlichen Namen und Adressen sind rein zufällig.

---

## Marken

Folgende Namen sind in gewissen Ländern Marken der IBM Corporation:

AIX	IBM
IBM System AS/400	IMS
CBIDO	MVS/ESA
CBPDO	Net.Data
CICS	OpenEdition
CustomPac	OS/2
DB2	OS/390
DB2 Universal Database	OS/400
DataJoiner	RACF
Distributed Relational Database Architecture	SystemPac
DRDA	

Folgende Namen sind in gewissen Ländern Marken anderer Unternehmen:

Java und alle auf Java basierenden Marken und Logos sind in gewissen Ländern Marken von Sun Microsystems, Inc.

UNIX ist in gewissen Ländern eine eingetragene Marke, die ausschließlich von der X/Open Company Limited lizenziert wird.

Lotus und Domino Go Webserver sind in gewissen Ländern Marken der Lotus Development Corporation.

Microsoft, Windows, Windows NT und das Windows-Logo sind in gewissen Ländern Marken oder eingetragene Marken der Microsoft Corporation.



# Glossar

**Anschluß.** Eine 16-Bit-Zahl, die für die Kommunikation zwischen TCP/IP und einem Protokoll oder einer Anwendung höherer Ebene verwendet wird.

**Anwendungsprogrammierschnittstelle (API - Application Programming Interface).** Eine vom Betriebssystem oder einem separat erhältlichen Lizenzprogramm zur Verfügung gestellte Funktionsschnittstelle, über die ein in einer höheren Programmiersprache geschriebenes Anwendungsprogramm bestimmte Daten oder Funktionen des Betriebssystems oder des Lizenzprogramms verwenden kann. Net.Data unterstützt die folgenden eigenen Web-Server-APIs zur Verbesserung der Leistung über CGI-Prozesse: ICAPI, GWAPI, ISAPI und NSAPI.

**API.** Anwendungsprogrammierschnittstelle

**Applet.** Ein Java-Programm, das in einer HTML-Seite enthalten ist. Applets können mit einem Java-fähigen Browser, wie zum Beispiel Netscape, verwendet werden und werden zusammen mit der HTML-Seite geladen.

**BLOB.** Großes Binärobjekt (Binary Large Object)

**Cache.** Eine Teil des Speichers, 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.

**Cache-Manager.** Das Programm, das einen Cache für eine Maschine verwaltet. Es kann mehrere Caches verwalten.

**Caching.** Der Prozeß zum lokalen Speichern häufig verwendeter Ergebnisse einer Anforderung an den Web-Server bis zur Aktualisierung der Daten, so daß sie schnell abgefragt werden können

**CGI.** Common Gateway Interface

**Cliette.** Ein über längere Zeit laufender Prozeß der Net.Data-Direktverbindung, der Anforderungen vom Web-Server bearbeitet. Connection Manager terminiert Cliette-Prozesse zur Verarbeitung dieser Anforderungen.

**CLOB.** Großes Zeichenobjekt (Character Large Object)

**Common Gateway Interface.** Ein Standardverfahren zur Übergabe der Steuerung durch einen Web-Server an ein Anwendungsprogramm und zum anschließenden Datenempfang

**Connection Manager.** Eine ausführbare Datei (dtwcm) in Net.Data, die zur Unterstützung einer Direktverbindung erforderlich ist

**Cookie.** Ein Datenpaket, das durch einen HTTP-Server an einen Web-Browser gesendet und dann vom Browser bei jedem Zugriff auf diesen Server zurückgesendet wird. Cookies können vom Server willkürlich gewählte Daten enthalten und werden zur Aufrechterhaltung des Status zwischen ansonsten statuslosen HTTP-Transaktionen verwendet. (Nach *Free Online Dictionary of Computing*)

**Datenbank.** Eine Gruppe von Tabellen oder von Tabellen- und Indexbereichen

**Datenbankverwaltungssystem (DBMS - Database Management System).** Ein Softwaresystem, das die Erstellung, den Aufbau und die Änderung einer Datenbank sowie den Zugriff auf die hierin gespeicherten Daten steuert

**Datentyp.** Ein Attribut aus Spalten und Literalen

**DBMS.** Datenbankverwaltungssystem (Database Management System)

**Direktverbindung.** Eine Net.Data-Komponente, die aus einem Connection Manager und mehreren Cliettes besteht. Die Direktverbindung verwaltet die Wiederverwendung der Verbindungen von Datenbanken und virtuellen Java-Maschinen.

**Firewall (Prozeßabgrenzung).** Ein Computer mit Software, die ein internes Netzwerk vor unbefugtem externen Zugriff schützt

**Flat File Interface (FFI).** Eine Gruppe von integrierten Net.Data-Funktionen, über die Sie Daten in Dateien mit unverschlüsseltem Text lesen/schreiben können

**HTML.** Hypertext Markup Language

**HTTP.** Hypertext Transfer Potocol

**Hypertext Markup Language.** Eine Befehlssprache zum Schreiben von Web-Dokumenten

**Hypertext Transfer Protocol.** Ein Übertragungsprotokoll, das zwischen Web-Server und Browser eingesetzt wird

**ICAPI.** Internet Connection API

**ICS.** Internet Connection Server

**ICSS.** Internet Connection Secure Server

**Internet.** Ein internationales, öffentliches TCP/IP-Computernetzwerk

**Internet Connection Secure Server.** Gesicherter Web-Server von IBM

**Internet Connection Server.** Nicht gesicherter Web-Server von IBM

**Intranet.** Ein TCP/IP-Netzwerk innerhalb eines firmen-internen Firewall

**ISAPI.** Internet Server API von Microsoft

**Java.** Eine vom Betriebssystem unabhängige, objektorientierte Programmiersprache, die sich besonders für Internet-Anwendungen eignet

**LOB.** Großes Objekt (Large Object)

**Middleware.** Software, die zwischen einem Anwendungsprogramm und einem Netzwerk vermittelt. Sie verwaltet die Interaktion zwischen einem Client-Anwendungsprogramm und einem Server über das Netzwerk.

**NSAPI.** Netscape API

**Null.** Ein Sonderwert, der angibt, daß keine Informationen vorhanden sind.

**Perl.** Eine interpretierte Programmiersprache

**Pfad.** Eine Suchroute zum Lokalisieren von Dateien

**Sprachumgebung.** Ein Modul, das Zugriff von einem Net.Data-Makro auf eine externe Datenquelle wie DB2 oder eine Programmiersprache wie Perl bietet. Einige Sprachumgebungen werden mit Net.Data ausgeliefert, wie zum Beispiel REXX, Perl und Oracle. Sie können ferner Ihre eigenen Sprachumgebungen erstellen.

**TCP/IP.** Transmission Control Protocol / Internet Protocol

**Transmission Control Protocol / Internet Protocol.** Eine Gruppe von Übertragungsprotokollen, die Peer-zu-Peer-Konnektivitätsfunktionen sowohl für lokale als auch für Weitverkehrsnetzwerke unterstützen

**Uniform Resource Locator.** Eine Adresse, die einen HTTP-Server und wahlweise ein Verzeichnis und einen Dateinamen angibt, wie zum Beispiel:  
<http://www.software.ibm.com/data/net.data/index.html>.

**URL.** Uniform Resource Locator

**Web-Server.** (1) Ein Computer, auf dem HTTP-Server-Middleware ausgeführt wird (2) Server-Software wie zum Beispiel Internet Connection Server

---

# Index

## A

- Ablaufverfolgungsmarkierungen, für
  - Cache-Manager 140
- Abschnitte einer Makrodatei
  - Deklaration 69
  - HTML 69
- Administration Tool
  - ENVIRONMENT, Anweisungen 47
  - installieren, Java-Laufzeitbibliotheken 38
  - Konfigurieren von Net.Data
    - Cliettes 42
    - Direktverbindungsanschlüsse 41
    - Konfigurationsvariablenanweisungen 51
    - Pfadanweisungen 39
    - Übersicht 38
    - Vorbereitung 38
  - verschlüsseln, Datenbank-Cliette-Kennwörter 45
- AIX, Net.Data für (Anhang) 171
- Anmeldename und Kennwort, Cliettes konfigurieren 28
- Anschlüsse
  - Cache-Manager 15, 139
  - Direktverbindung
    - Konfigurationsdatei 26, 28
    - konfigurieren mit Administration Tool 41
- Apache Web Server installieren 31
- Arten von Variablen 80
- Aufrufen gespeicherter Prozeduren 96, 98
- Aufrufen von Funktionen 94
- Aufrufen von Net.Data
  - CGI 59
  - Direktanforderung 59
  - FastCGI 34
  - Formulare 61, 67
  - GWAPI 158
  - HTML-Blöcke 107
  - ICAPI 158
  - ISAPI 158
  - Makroanforderung 59
  - mit einer Makrodatei 61
  - NSAPI 159
  - ohne Makrodatei 63
  - Programmverbindungen (Links) 61, 67
  - Syntax 60
  - Übersicht 59
  - URL-Adressen 61, 68
- Ausführbare Variablen 81
- Ausgabe, inaktivieren für Standardberichte 109
- Authentifizierung, Sicherheit 56

## B

- Bedingt
  - Logik, IF-Blöcke 111
  - Variablen 80
- Beispielmakro A 182
- Bemerkungen 187
- Benutzerdefinierte Funktionen 91
- Benutzerverzeichnis
  - konfigurieren in der Initialisierungsdatei 17, 38
  - konfigurieren mit Administration Tool 51
- Berechtigung
  - Sicherheit 56
  - Zugriffsrechte für Net.Data-Dateien angeben 52
- Berichtsformate anpassen 110
- Berichtsvariablen 89
- BLOBs 115
- Blöcke, Makrodatei 72

## C

- Cache
  - aktivieren, aktueller 141
  - Definition 134
  - Hauptspeicher angeben 142
  - Kennung 135, 138
  - Pfad 141
  - Seitenalter angeben 142
  - Speicherbereich für Seiten angeben 142
  - Zeilengruppe konfigurieren 141
- Cache-ID
  - Definition 135
  - planen 138
- Cache-Manager
  - Anschluß 139
  - definieren 139
  - Definieren eines Cache 141
  - Definition 135
  - Konfigurationsdatei 9, 135, 139
  - Konfigurationsvariablen 15
  - Leistungsoptimierung 164
  - Protokolldatei
    - Ablaufverfolgungsmarkierungen 140
    - aktivieren 140
    - benennen 139
    - für jeden Cache 144
  - starten 146
  - stoppen 147
  - Überschreitung des Verbindungszeitlimits 140
  - Zeilengruppe konfigurieren 139
- Cache-Transaktionsprotokolldatei 144

- CACHE\_MACHINE 15
- CACHE\_PORT 15
- cacheadm
  - stoppen, Cache-Manager 147
  - Syntax 151
- Caching
  - Abfragen eines bestimmten Cache 151
  - Beispielanwendungen 133
  - cacheadm, Befehl 151
  - Einführung 134
  - Einschränkungen 136
  - Konfiguration ermitteln 135
  - leeren 151
  - Leistungsoptimierung 164
  - Markierungen 151
  - planen 138
  - Protokollierung 140, 151
  - Schnittstellen 136
  - Seite 148
  - Statistikdaten sammeln 151
  - stoppen 151
  - Terminologie 134
- Cliettes
  - Beschreibung 161
  - konfigurieren mit Administration Tool
    - ändern 44
    - Datenbankinformationen 45
    - hinzufügen 42
    - löschen 45
    - testen, DB2-Datenbankanmeldung 45
    - verschlüsseln, Datenbankkennwörter 45
  - Namen ausführbarer Dateien 27
- CLOBs 115
- Common Gateway Interface. Siehe CGI
- Connection Manager
  - Beschreibung 161
  - starten
    - AIX 162
    - mit der Nachrichtenoption 163
    - OS/2 und Windows NT 162

## D

- Dateien, Zugriffsrechte für Net.Data angeben 52
- Datenbank
  - Cliettes, konfigurieren 42
- Datentypen, gültig für gespeicherte Prozeduren 97
- DB2INSTANCE 16
- DBCS-Unterstützung für Funktionen 17
- DBLOBs 115
- DEFINE-Block
  - Beschreibung 72
  - Definieren von Variablen 78
- Definieren von Variablen
  - DEFINE-Anweisung bzw. -Block 78
  - HTML-Formularbefehle SELECT und INPUT 78

- Definieren von Variablen (*Forts.*)
  - URL-Daten 79
- Deklarationsabschnitt, Makrodateistruktur 69
- Direktanforderung
  - Beispiele 67
  - Beschreibung 59
  - Caching-Einschränkungen 136
  - Syntax 64
- Direktverbindung
  - Anschlüsse
    - konfigurieren in der Initialisierungsdatei 26, 28
    - konfigurieren mit Administration Tool 41
  - Cliettes
    - Konfigurationsdateien 10
    - konfigurieren mit Administration Tool 42
  - Einsatzmöglichkeiten 162
  - Konfigurationsdatei
    - aktualisieren 25
    - Anmeldename und Kennwort 28
    - Anzahl der Prozesse 26, 28
    - Beispiel 12
    - Beschreibung 8
    - Datenbank-Cliettes 26
    - Datenbankname 28
    - Format 25
    - Java-Cliettes 28
    - Name 26
    - Prozeßart 27
  - Leistungsoptimierung 160
  - starten, Connection Manager 162
  - Verarbeitungsablauf 163
  - Vorteile 162
- Domino Go Webserver installieren 31
- DTW\_CACHE\_PAGE 148
- DTW\_CM\_PORT 16
- DTW\_INST\_DIR 17, 51
- DTW\_LOG\_DIR 17
- DTW\_LOG\_LEVEL 51, 165, 168
- DTW\_MBMODE 17
- DTW\_OPTIMIZE\_MATH 18
- DTW\_SMTP\_SERVER 18
- dtwcm, Befehl 162

## E

- ENVIRONMENT, Anweisungen
  - Beispiel 24
  - Beschreibung 22, 47
  - Cliette-Name 23
  - DLL- oder Bibliotheksname 23
  - konfigurieren in der Initialisierungsdatei 22, 23
  - Parameterliste 23
  - Sprachumgebungsart 23
  - Syntax 23
- Ergebnismengen
  - mehrere 101

Ergebnismengen (*Forts.*)  
  mehrere, Richtlinien und Einschränkungen 104  
  nur eine 99  
  verarbeiten, gespeicherte Prozeduren 99  
EXEC\_PATH  
  konfigurieren in der Initialisierungsdatei 20  
  konfigurieren mit Administration Tool 39

## F

FastCGI  
  konfigurieren für Net.Data  
    installieren, Apache Web Server 31  
    installieren, Domino Go Webserver 31  
  Konfigurieren von Net.Data 31  
  Leistungsüberlegungen 159  
  simultane Prozesse ermitteln 160  
  unterstützte Sprachumgebungen 31, 159  
Fehlerprotokollierung  
  Beschreibung 167  
  DTW\_LOG\_DIR 17, 167  
  DTW\_LOG\_LEVEL 51, 168  
  planen 167  
  Protokolldatei  
    aktivieren 167  
    Format 169  
    Größe 167  
    Speicherposition angeben 17  
    Speicherpositionsvariable 17  
  Protokollstufe  
    angeben 51, 168  
    Auswirkung auf Leistung 165  
    Variable 51, 168  
  Überlegungen zur Leistung 165  
FFI\_PATH  
  konfigurieren in der Initialisierungsdatei 21  
  konfigurieren mit Administration Tool 39  
Firewalls 53  
Formatieren der Datenausgabe 109  
Formulare  
  Aufrufen von Net.Data 61, 67  
  in Web-Seiten zum Aufrufen von Net.Data 62  
FUNCTION-Block  
  Aufrufen von Funktionen 94  
  Beschreibung 73  
  Formatieren der Ausgabe 109  
  Geltungsbereich für Kennungen 76  
  Verarbeiten von Funktionsaufrufen 94  
  Verarbeiten von Variablenverweisen 94  
FunctionServlet  
  NOF-Plug-In 177  
Funktionen  
  aufrufen 94  
  Aufrufen gespeicherter Prozeduren 96  
  benutzerdefinierte 91  
  Beschreibung 90

Funktionen (*Forts.*)  
  definieren 91  
  FUNCTION-Blocksyntax 91  
  integrierte 116  
  MACRO\_FUNCTION-Blocksyntax 91  
Funktions-Servlet  
  ausführen 125  
  Beschreibung 122  
Funktionsaufrufe  
  Syntax 94  
  Verarbeitungsreihenfolge 94  
Fußzeilen, REPORT-Block 109

## G

Geltungsbereich  
  REPORT-Block 77  
  Variablen 76  
Geltungsbereich, Kennung  
  FUNCTION-Block 76  
  global 76  
  Makrodatei 76  
  ROW-Block 77  
Gemeinsam benutzte Bibliotheken  
  Laden für Sprachumgebungen unter AIX 171  
Gespeicherte Prozeduren  
  aufrufen in Makrodatei 96  
  Einschränkungen 104  
  gültige Datentypen 97  
  mehrere Ergebnismengen 101  
  mit einer Ergebnismenge 99  
  Net.Data-Tabellen 100, 102  
  REPORT-Blöcke 100, 103  
  Richtlinien 104  
  Schritte 98  
  Standardberichte 99, 101  
  Übergeben von Parametern 99  
  Verarbeiten von Ergebnismengen 99  
Globaler Geltungsbereich für eine Kennung 76  
Glossar 189  
Große Objekte  
  Beschreibung 115  
  gültige Formate 115  
  Hardware- und Softwarevoraussetzungen 116  
GWAPI  
  Aufrufen von Net.Data 158  
  konfigurieren für Net.Data 35

## H

HTML 108  
  Abschnitt, Makrodateistruktur 69  
  Befehle für Tabellen 109  
  Blöcke  
    Aufrufen von Net.Data 107  
    Beispiel 107  
    Beschreibung 74

## HTML (Forts.)

### Blöcke (Forts.)

Verarbeitung 108

FORM, Knopf zum Übergeben (Submit) 108

### Formulare

Aufrufen von Net.Data 61, 67

Informationen zu 62

SELECT- und INPUT-Befehle, Definieren von Variablen 78

generieren in einer Makrodatei 107

### Programmverbindungen (Links)

Aufrufen von Net.Data 61, 67

Informationen zu 61

URL-Adressen, Net.Data aufrufen 68

## HTML\_PATH

konfigurieren in der Initialisierungsdatei 22

konfigurieren mit Administration Tool 39

## I

### ICAPI

Aufrufen von Net.Data 158

konfigurieren für Net.Data 35

und Domino Go Webserver (GWAPI) 35

### IF-Blöcke 111

### IN-Parameter 95

### INCLUDE\_PATH

konfigurieren in der Initialisierungsdatei 20

konfigurieren mit Administration Tool 39

Informationen zu diesem Handbuch ix

### Initialisierungsdatei

aktualisieren 13

Beispiel 12

Beschreibung 7

ENVIRONMENT, Anweisungen 22

Format 13

Konfigurationsvariablenanweisungen 14

Pfadanweisungen 18

### INOUT-Parameter 95

### inst\_verz 38

### Installieren

Net.Data 5

### Integrierte Funktionen 116

### ISAPI

Aufrufen von Net.Data 158

konfigurieren für Net.Data 36

## J

Java-Cliettes konfigurieren 28

## K

Kennwort und Anmeldename, Cliettes konfigurieren 28

### Konfigurationsvariablenanweisungen

Benutzerverzeichnis 17

## Konfigurationsvariablenanweisungen (Forts.)

Beschreibung 14

CACHE\_MACHINE 15

CACHE\_PORT 15

DB2INSTANCE 16

DTW\_CM\_PORT 16

DTW\_INST\_DIR 17, 51

DTW\_LOG\_DIR 17

DTW\_LOG\_LEVEL 51

DTW\_MBMODE 17

DTW\_OPTIMIZE\_MATH 18

DTW\_SMTP\_SERVER 18

### konfigurieren

mit Administration Tool 51

konfigurieren in der Initialisierungsdatei 14

Optimieren mathematischer Funktionen 18

Konfigurieren des Cache-Managers 139, 141

Konfigurieren für DataJoiner

Konfigurieren von Net.Data

### Administration Tool

Anschlüsse 41

Cliettes 42

ENVIRONMENT, Anweisungen 47

installieren, Java-Laufzeitbibliotheken 38

konfigurieren, Variablenanweisungen 51

Pfadanweisungen 39

Übersicht 38

Vorbereitung 38

### FastCGI 31

### für CGI

Gegenüberstellung der Methoden 6

### Initialisierungsdatei

aktualisieren 13

Anweisungen ENVIRONMENT 22

Beschreibung 7

Konfigurationsvariablenanweisungen 14

Pfadanweisungen 18

### Konfigurationsdatei für den Cache-Manager

Anschlüsse 15

Beschreibung 9

Zeilengruppen 139, 141

### Konfigurationsdatei für Direktverbindungen 26

aktualisieren 25

Anschlüsse 26, 28

Beschreibung 8

manuell im Vergleich zu Administration Tool 5

Steuerdateivergleich 10

Übersicht 5

Zugriffsrechte für Net.Data-Dateien 52

zur Verwendung mit Web-Server-APIs 35

Kopfzeilen, REPORT-Block 109

## L

Leistung 152, 155

Caching 164

Leistung (*Forts.*)  
Direktverbindung 160  
FastCGI 159  
Fehlerprotokollierung 165  
REXX-Umgebung 171  
SQLCODE-Nachrichten  
Web-Server-APIs 157  
Listenvariablen 85  
LOBs 115

## M

MACRO\_FUNCTION-Block  
Aufrufen von Funktionen 94  
Syntax 91  
Verarbeiten von Funktionsaufrufen 94  
Verarbeiten von Variablenverweisen 94  
MACRO\_PATH  
konfigurieren in der Initialisierungsdatei 19  
konfigurieren mit Administration Tool 39  
MacroServlet  
NOF-Plug-In 177  
Makro-Servlet  
ausführen 123  
Beschreibung 122  
Makroanforderung  
Beispiele 61  
Beschreibung 59  
Syntax 61  
Makrodateien  
Aufbau 70  
bedingte Logik 111  
Beispiel 11, 70  
Beschreibung 1  
Blöcke 72  
DEFINE-Block 72  
Deklarationsabschnitt 69  
entwickeln 69  
FUNCTION-Block 73  
Funktionen 90  
Geltungsbereich für Kennungen 76  
Generieren von HTML 107  
HTML  
Abschnitt 69  
Block 74  
IF-Blöcke 111  
Navigation innerhalb und zwischen 75  
NOF-Plug-Ins 177  
Schleifen 114  
Variablen 76  
WHILE-Blöcke 114  
Mathematische Funktionen, Leistungsoptimierung 18  
MAX\_PROCESS 26, 28, 44  
MESSAGE-Block  
Beispiel 106  
Beschreibung 104

MESSAGE-Block (*Forts.*)  
Geltungsbereich 105  
Syntax 105  
Verarbeitung 104  
MIN\_PROCESS 26, 28, 44

## N

Navigation, innerhalb und zwischen Makros 75  
Net.Data  
aufrufen 59  
Dateien, Zugriffsrechte 52  
installieren 5  
konfigurieren 5  
Makrodateien, entwickeln 69  
Sicherheitsmechanismen 57  
Übersicht 1  
zusätzliche Informationen ix  
Net.Data-Makro. Siehe Makrodateien. 1  
Net.Data-Servlets  
Funktions-Servlet 122  
Makro-Servlet 122  
NOF-Plug-Ins  
ändern, Merkmale 181  
Beschreibung 177  
definieren 178  
veröffentlichen, Servlets 182  
Net.Data-Tabellen, gespeicherte Prozeduren 100, 102  
NetObjects Fusion (NOF), Plug-Ins  
Ändern von Servlet-Merkmalen 181  
Beschreibung 177  
definieren 178  
für Makro- und Funktions-Servlets 177  
Hardware- und Softwarevoraussetzungen 178  
installieren 178  
veröffentlichen 182  
NOF (NetObjects Fusion), Plug-Ins 177  
NSAPI  
Aufrufen von Net.Data 159  
konfigurieren für Net.Data 36

## O

Optimieren der Leistung 155  
OUT-Parameter 95

## P

Pfadanweisungen  
Aktualisierungsrichtlinien 19  
EXEC\_PATH 20  
FFI\_PATH 21  
HTML\_PATH 22  
INCLUDE\_PATH 20  
konfigurieren in der Initialisierungsdatei 18  
konfigurieren mit Administration Tool  
ändern 40

- Pfadanweisungen (*Forts.*)
  - konfigurieren mit Administration Tool (*Forts.*)
    - hinzufügen 40
    - löschen 40
  - MACRO\_PATH 19
  - schützen, Datenbestände 57
- Plug-Ins, NetObjects Fusion 177
- Programmverbindungen (Links)
  - Aufrufen von Net.Data 61, 67
  - in Web-Seiten zum Aufrufen von Net.Data 61
- Protokolldatei
  - aktivieren 17, 167
  - Archivierung 168
  - Cache-Manager 139, 140
  - Format 169
  - für jeden Cache 144
  - maximale Größe 167, 168
  - Steuerungsebene 167

## R

- REPORT-Block
  - Beschreibung 109
  - Formatieren der Datenausgabe 109
  - Geltungsbereich 77
  - gespeicherte Prozeduren 100
  - Kopf- und Fußzeilen 109
- REPORT-Blöcke
  - gespeicherte Prozeduren 103
- RETURN\_CODE, Variable 95, 104
- RETURNS, Parameter 96
- REXX, Leistungsverbesserung 171
- ROW-Block, Geltungsbereich für Kennungen 77

## S

- Schleifen, WHILE-Blöcke 114
- Schützen, Datenbestände 53
- Servlets
  - API-Dokumentation 122
  - ausführen 122
  - Beschreibung 121
  - definieren 122
  - Net.Data
    - ändern, Merkmale mit Plug-In 181
    - definieren, Plug-In 178
    - Funktion 122
    - Makro 122
  - NetObjects Fusion, Plug-Ins 177
  - NOF-Plug-Ins 177
  - veröffentlichen mit NOF-Plug-Ins 182
- Sicherheit
  - Authentifizierung 56
  - Berechtigung 56
  - Caching 136
  - Firewall 53

- Sicherheit (*Forts.*)
  - Net.Data-Mechanismen 57
  - Netzwerkverschlüsselung 55
  - Übersicht 53
  - verschlüsseln, Datenbank-Cliette-Kennwörter 45
  - Zugriffsrechte angeben 52
- Sprachumgebungen 118
  - Beispiele 22
  - konfigurieren in der Initialisierungsdatei 22
  - konfigurieren mit Administration Tool
    - ändern 48
    - hinzufügen 47
    - löschen 50
  - konfigurieren von Anweisungen
    - ENVIRONMENT 22, 47
  - Laden nutzter Bibliotheken unter AIX 171
  - Variablen 89
- SQLCODE-Nachrichten, inaktivieren
- Standardberichte
  - angeben für gespeicherte Prozeduren 99, 101
  - ausgeben 109
- Starten von Net.Data 59

## T

- Tabellenvariablen 86
- Tabellenverarbeitungsvariablen 88
- Token-Größen 76

## U

- Übergeben von Parametern, gespeicherte
  - Prozeduren 99
- Überschreitung des Verbindungszeitlimits, Cache-Manager 140
- Umgebungsvariablen 81
- Unterstützung der Landessprache für Funktionen 17
- URL-Adressen
  - Aufrufen von Net.Data 61, 68
  - Definieren von Variablen 79

## V

- Variable für das Optimieren mathematischer Funktionen, konfigurieren in der Initialisierungsdatei 18
- Variable für Installationsverzeichnisconfiguration
  - konfigurieren in der Initialisierungsdatei 17
  - konfigurieren mit Administration Tool 51
- Variablen
  - Arten von 76, 80
  - ausführbar 81
  - bedingt 80
  - Bericht 89
  - Beschreibung 76
  - definieren 78
  - für Listen 85



## Variablen (*Forts.*)

- für Tabellen 86
- für Umgebung 81
- Geltungsbereich 76
- Konfiguration, Anweisungen
  - Administration Tool 51
  - Benutzerverzeichnis 17, 18, 51
  - Beschreibung 14
  - Cache-Einheitennamen (CACHE\_MACHINE) 15
  - Cache-Manager-Anschluß (CACHE\_PORT) 15
  - DB2-Exemplar (DB2INSTANCE) 16
  - E-Mail-SMTP-Server (DTW\_SMTP\_SERVER) 18
  - Editiermasken (DTW\_CM\_PORT) 16
  - Fehlerprotokollstufe (DTW\_LOG\_LEVEL) 51
  - Initialisierungsdatei 14
  - Installationsverzeichnis (DTW\_INST\_DIR) 17, 51
  - Optimieren mathematischer Funktionen (DTW\_OPTIMIZE\_MATH) 18
  - SMTP-Server (DTW\_SMTP\_SERVER) 18
  - Speicherposition des Fehlerprotokolls (DTW\_LOG\_DIR) 17
  - Unterstützung der Landessprache (DTW\_MBMODE) 17
- Sprachumgebung 89
- Tabellenverarbeitung 88
- Token-Größen 76
- verdeckt 84
- verschiedene 87
- Verweisen auf 79
- Variablenverweise, Verarbeitungsreihenfolge 94
- Verarbeiten von Ergebnismengen, gespeicherte Prozeduren 99
- Verbindungsverwaltung
  - konfigurieren 25
  - Leistung 160
- Verdeckte Variablen
  - schützen, Datenbestände 57
  - Variablenamen verdecken 84
- Veröffentlichen, Servlets mit NOF-Plug-Ins 182
- Verschiedene Variablen 87
- Verschlüsselung
  - Datenbank-Cliette-Kennwörter 45
  - Netzwerk 55
- Verweisen auf Variablen 79

## W

- Web-Registrierdatenbank
- Web-Seiten, zwischenspeichern 148
- Web-Server
  - konfigurieren für FastCGI 31
  - konfigurieren für Web-Server-APIs 35
- Web-Server-APIs 157
  - Aufrufen von Net.Data
    - GWAPI 158
    - ICAPI 158
    - ISAPI 158

## Web-Server-APIs (*Forts.*)

- Aufrufen von Net.Data (*Forts.*)
  - NSAPI 159
- Beschreibungen 157
- konfigurieren für Net.Data
  - Beschreibung 35
  - GWAPI 35
  - ICAPI 35
  - ISAPI 36
  - NSAPI 36
- Leistungsoptimierung 157
- WHILE-Blöcke 114

## Z

- Zeilengruppe
  - Cache-Manager, konfigurieren 139
  - Cache, konfigurieren 141
- Zugriffsrechte, angeben für Net.Data-Dateien 52

