Net.Data 管理与程序设计指南，OS/2、Windows NT 以及 UNIX 版

Net.Data 管理与程序设计指南，OS/2、Windows NT 以及 UNIX 版

注意

在使用本信息及其所支持的产品之前，务必请阅读第147页的『附录E. 注意事项』中的信息。

第四版 **(1998 年 6 月)**

目录

前言

感谢您选择 Net.Data 版本 2 - IBM 的开发工具来创建动态 Web 页面! 使用 Net.Data 之后，您就可以迅速地开发具有动态内容的 Web 页面，这只要通过结合来自广泛种类数据源的数据并使用您已知的程序设计语言的功能即可实现。

Net.Data 版本 2 提供了显著改进的性能和一些新的功能，这些新功能将赋予您构建与采纳自己的 Internet 商业方案的能力。

关于 Net.Data

采用 IBM 的 Net.Data 产品之后，您就可以使用来自关系型或非关系型数据库管理系统(DBMS，包括 DB2、IMS 和 允许使用 ODBC 的数据库)的数据来创建动态的 Web 页面；还可以使用各种编程语言(例如 Java、JavaScript、Perl、C、C++ 和 REXX) 所编写的应用程序。

Net.Data 是一个在 Web 服务器上作为中件执行的宏处理器。您可以编写 Net.Data 应用程序(称之为宏)，Net.Data 将对它进行解释以便使用根据用户输入、数据库当前状态、现有商业逻辑以及您在宏中所设计的其它因素而定制的内容来创建动态的 Web 页面。

一个 URL (统一资源定位器)形式的请求，从浏览器(例如 Netscape Navigator 或 Internet Explorer) 流动到将请求转发给 Net.Data 进行执行的 Web 服务器。Net.Data 找出这个宏加以执行，并构建一个根据您所编写的函数定制的 Web 页面。这些函数能够:

- 在 Perl 脚本、C 与 C++ 应用程序或 REXX 程序中封装商业逻辑
- 访问诸如 DB2 等数据库

Net.Data 将这个 Web 页面传递到 Web 服务器，随后 Web 服务器通过网络转发这个页面，最后显示在浏览器上。

Net.Data 支持诸如超文本传输协议 (HTTP) 和公共网关接口 (CGI) 等工业标准接口。HTTP 用在浏览器和 Web 服务器之间，而 CGI 用在 Web 服务器和 Net.Data 之间。这种支持可以使您选择喜爱的浏览器或 Web 服务器与 Net.Data 一起使用。Net.Data 系列产品在 OS/400、OS/390、Windows NT、 AIX、OS/2、HP-UX、Sun Solaris 和 SCO 操作系统上提供了类型的功能。Net.Data 还支持多个操作系统上的 FastCGI 和主要的 Web 服务器应用程序接口 (API)。

另外，Net.Data 版本 2 中还包含了其它性能的改进，以满足应用程序的要求，包括:

- HTML 页面的高速缓存
- 不使用宏文件对 Net.Data 进行调用(直接请求)
- 在生成的 Web 页面中将无关重要的空白减到最少
- 优化的数学函数

Net.Data 版本 2 还包括了许多功能上的改进:

- 语言环境的增强包括从存储过程通过 SQL 或 ODBC 语言环境检索一个或多个结果集合的能力。
- 宏语言增强包括:
  - 将注释放在任意地方的能力
  - 嵌套的 IF 块

- WHILE 块
- 从一个存储过程接收单个或多个 结果集合的能力
- 支持 DBCS 的字符串和字处理函数
- 支持存储过程参数表中的 SQL 十进制数据类型
- 对 cookies 的支持
- 对动态生成的电子邮件消息的支持

一个图形管理工具可以帮助您管理 AIX、Windows NT 和 OS/2 操作系统上的 Net.Data 配置设置。管理工具还可以辅助您为使用“现场连接”的数据库连接指定安全性。

为了帮助您方便地从数据库访问数据，Net.Data 提供了各种各样的工具，包括 NetObjects Fusion 插件和基于 Java 开发的向导。这些工具在 Java 环境中与 Net.Data Java 小应用程序一起使用，允许您创建可以在操作系统间移植的应用程序。NetObjects Fusion 插件允许您使用 NetObjects Fusion Web 开发工具来构建使用来自关系型数据源的动态数据的复杂应用程序。Net.Data 向导提供了一个图形工具，它将指导您创建基本的 Net.Data 宏。

关于本书

本书讨论 Net.Data 的管理和编程概念，以及如何配置 Net.Data 和它的各个组件、如何计划安全性、如何改进性能。

根据您对于编程语言和数据库的知识，您需要学习如何使用 Net.Data 宏语言或 Java 小服务程序来开发宏。您要学习如何使用 Net.Data 提供的语言环境(这些语言环境使用 IMS Web 访问 DB2 数据库和 IMS 事务)，还要学习使用 Java、REXX、Perl 和其它编程语言来访问您的数据。

本书可能引用已经发布、但现在还未进入实用的某些产品或功能。

包括示例 Net.Data 宏、演示程序以及本书最新副本在内的更多信息，可以从以下 World Wide Web 站点获得:

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

谁应当阅读本书

本书的读者是规划和编写 Net.Data 应用程序的人员。操作系统的区别、Net.Data 消息以及其它信息都在 *Net.Data 参考*一书中有所描述。

要了解本书中讨论的概念，您应当熟悉 Web 服务器如何工作，了解简单的 SQL 语句，并知道 HTML 标记(包括 HTML 表格标记)。另外，您还应当熟悉 *Net.Data 参考*和 *Net.Data 语言环境参考*中的信息

关于本书中的例子

本书中出现的例子相对较为简单，目的是演示特定的概念，而不说明 Net.Data 构造元的所有用法。某些例子只是一些片段，需要其它代码才能运行。

第1章 Net.Data 是什么?

如果只使用 HTML，您可以创建静态的 Web 页面；换句话说，除非您对这些页面进行编辑，否则它们不会更改。要想将“现场”数据和应用程序放到 Web 上(例如当前的销售统计)，Web 站点的开发者通常会编写一些在 Web 服务器上作为中件执行的程序，从而动态地构建 Web 页面。编写这些类型的程序并不容易。

Net.Data 通过宏简化了交互式 Web 应用程序的编写。采用 Net.Data 宏之后，您就可以使用逻辑、变量、函数调用以及报表生成工具。宏是一个文本文件，包含 Net.Data 宏语言结构、HTML 标记以及语言环境语句(例如 SQL 和 Perl)。Net.Data 处理宏文件来产生 HTML 输出。宏组合了 HTML 的简单性以及 Web 服务器程序的动态功能，从而使得向静态 Web 页面中添加现场数据变得简单。现场数据可以从本地或远程的数据库以及平面文件中抽取，也可以由应用程序和系统服务生成。

第1页的图 1列出了 Net.Data 和 Web 服务器以及支持的数据和编程语言环境之间的关系。

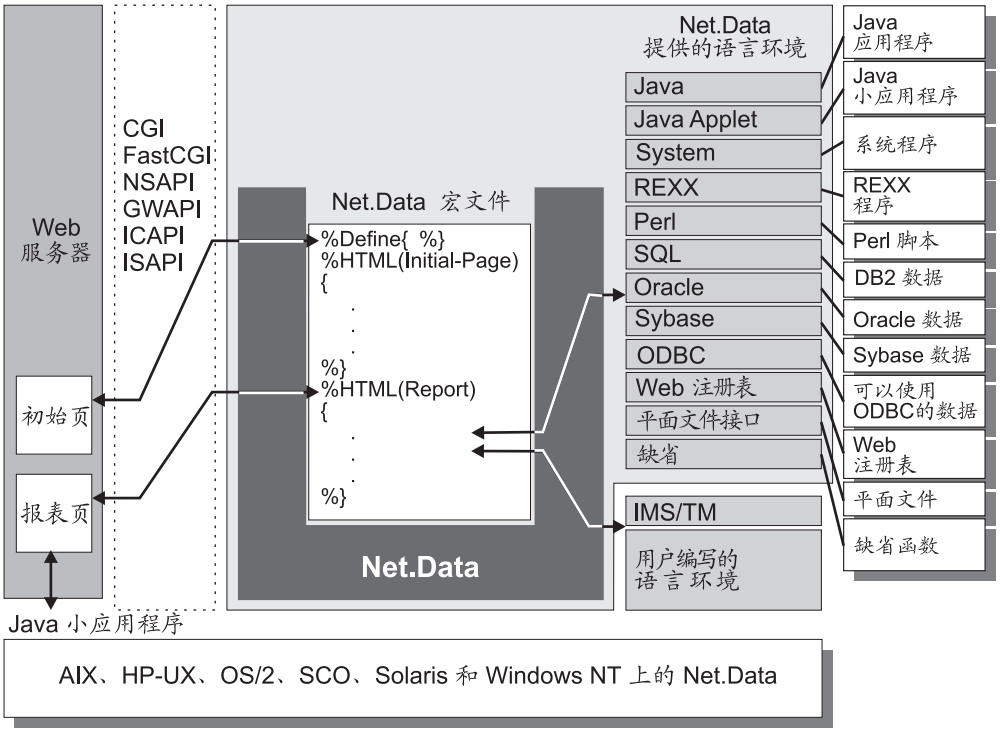


图 1. Net.Data、Web 服务器和支持的数据以及程序源之间的关系

Web 服务器将其作为 CGI 或 FastCGI 进程来调用 Net.Data，或通过在接收到请求 Net.Data 服务的 URL 时将 Net.Data 作为 DLL 或共享程序库来使用 Web 服务器应用程序的程序设计接口 (API) 线程，从而调用 Net.Data。这个 URL 中包含特定于 Net.Data 的信息，包括要处理的宏文件或者要直接调用的 SQL 语句或程序。当 Net.Data 完成对请求的处理时，它将把结果的 Web 页面发送给 Web 服务器。服务器将它传递给 Web 客户，在那里它将通过浏览器显示。

Net.Data 是创建动态 Web 页面的一个很好的选择，因为使用宏语言要比编写您自己的 Web 服务器应用程序简单，并且 Net.Data 允许您使用已知的语言，例如 HTML、SQL、Perl、REXX 和 JavaScript。Net.Data 还提供了语言环境，它们访问 DB2 数据库、使用 IMS Web 执行 IMS 事务、使用 REXX、Perl 和其它用于您的应用程序的语言。

另一个重要的优点是 Net.Data 支持对多个不同数据库源的访问，允许 Web 开发者使用来自各种数据库的数据，包括 DB2、IMS、Oracle、Sybase 和任何支持 ODBC 功能的数据源。请参阅 *Net.Data 语言环境参考*以获取有关 Net.Data 提供的语言环境的更多信息。

Net.Data Web 应用环境提供了以下功能:

解释宏语言

Net.Data 宏语言是一种解释语言。当调用 Net.Data 来处理宏时，Net.Data 将以一种顺序的方法直接解释每个语言语句，从文件的顶部开始。使用这种方法以后，如果您更改了一个宏，那么在下一次指定执行该宏的 URL 时将可以立即看到您所作的任何更改。不需要重新编译。

直接请求

需要执行简单的 SQL 语句、DB2 存储过程、REXX 程序、C 或 C++ 程序、Perl 脚本的简单请求，不需要创建宏。这些请求可以在从浏览器流向 Web 服务器的 URL 内直接指定。

自由格式

Net.Data 宏语言只有一些关于编程格式的规则。这种简单性为程序员提供了自由和灵活性。单条指令可以跨越多行，或者多条指令可以在一行中输入。指令可以从任何一列开始。空格或整个的空行都可以跳过。注释可以使用在任何地方。

无类型的变量

Net.Data 将所有的数据都看作字符串。Net.Data 使用内部函数来对代表有效数值的字符串执行算术运算，包括那些指数格式的字符串。宏语言变量在第60页的『Net.Data 宏变量』中详细讨论。

内部函数

Net.Data 提供了对文本和数值执行各种不同的处理、搜索以及比较操作的内部函数。其它内部函数提供了格式化的功能和算术计算的能力。

错误处理

当 Net.Data 检测到一个错误时，带有说明的错误信息将会返回给客户。您可以在错误信息返回到用户之前在浏览器中定制它们。请参阅 *Net.Data 参考*以获取更多信息。

第2章 配置 Net.Data

您可以通过使用产品附带的自述文件中的指导来为操作系统安装 Net.Data。大部分的配置步骤在安装过程中完成；这是根据操作系统而有所不同的。

在为您的操作系统安装 Net.Data 之后，必须配置 Net.Data 并修改对 Web 服务器的配置。配置任务包括:

- 定制 Net.Data 初始化 (INI) 文件
- 配置 Net.Data，使它能够与 FastCGI 或一个受到支持的 Web 服务器 API (可选)一起使用
- 定制 Web 服务器配置和环境变量文件
- 配置高速缓存管理器(可选)
- 配置现场连接(可选)
- 指定访问权

使用以下工具来配置 Net.Data:

- 一个文本编辑器

  在所有的操作系统上，使用一个文本编辑器来编辑 INI 文件和“现场连接”配置文件。您还要使用文本编辑器来更新所有的 Web 服务器配置文件。在更改这些文件之前进行备份是个不错的主意。

- Net.Data 管理工具

  管理工具提供了一个图形界面，用于定制 INI 文件和“现场连接”配置文件。您可以使用管理工具在 OS/2、Windows NT 和 AIX 操作系统上配置 Net.Data。

您所使用的方法取决于需要配置的组件以及运行 Net.Data 的操作系统，如第3页的表 1 中的描述。如果您使用一个特定的方法来启动一个配置任务，那么您应继续使用这种方法，以获取最佳结果。

*表 1. 配置方法(包括任务和操作系统)的比较。**A** - 可以使用管理工具进行配置，或手工配置。**M** - 只可以手工配置。*

任务	平台:			
	AIX	**NT**	**OS/2**	**HP|SUN|SCO**
配置 Net.Data INI 文件	A	A	A	M
定义 cliette 端口	A	A	A	M
定义 cliette	A	A	A	M
启用 cliette 口令加密	A	N/A	N/A	N/A
启用错误记录	A	A	A	M
为 FastCGI、CGI 和 API 配置 Web 服务器*	M	M	M	M
定义高速缓存管理器端口	M	M	N/A	N/A
配置高速缓存管理器	M	M	N/A	N/A

***技巧**: 许多 Web 服务器都提供了管理工具，您可以使用这些工具来配置 Web 服务器。

本章将描述如何配置 Net.Data 以及如何修改 Web 服务器的配置以便与 Net.Data 一起使用。另外，还将描述如何配置可选的组件。

- 第9页的『定制 Net.Data 初始化文件』
- 第22页的『为 FastCGI 配置 Net.Data』
- 第26页的『配置 Net.Data 以便与 Web 服务器 API 一起使用』
- 第28页的『使用 Net.Data 管理工具来配置 Net.Data』
- 第39页的『指定对 Net.Data 文件的访问权』

关于 Net.Data 初始化文件

Net.Data 使用它的初始化文件来建立各种配置变量的设置，并配置语言环境和搜索路径。配置变量的设置控制 Net.Data 操作的各个方面，例如在 Web 页面内对字符数据进行编码、字符串函数或字处理函数是否支持 DBCS 以及用于访问数据库数据的 DB2 实例的名称。 这些设置还指定了 Net.Data 如何连接到语言环境、数据库、连接管理、高速缓存并与它们通信，并指定错误记录是否被激活。

语言环境语句定义了可用的 Net.Data 语言环境，并标识在语言环境之间来回流动的特殊的输入、输出参数值。路径语句指定了到 Net.Data 使用的文件(例如，宏、REXX 程序和 Perl 脚本)的目录路径。

Net.Data 初始化文件 db2www.ini 位于 Web 服务器的文档目录中。请参阅适用于您所使用的操作系统的自述文件，以获取更多信息。

权限提示: 确保此 Web 服务器具有对该文件的访问权。请参阅第39页的『指定对 Net.Data 文件的访问权』，以获取更多信息。

关于可选组件的 Net.Data 配置文件

以下章节将讨论 Net.Data 可选组件的配置文件。

第4页的『现场连接配置文件』

第5页的『高速缓存管理器配置文件』

第5页的『Net.Data 初始化、控制和宏文件的公用节』

现场连接配置文件

“现场连接”在 Windows NT、OS/2 和 UNIX 操作系统上提供了连接管理，从而通过消除启动时的系统开销来改进性能。Net.Data 现场连接配置文件中包含有关一个或多个已经命名的 cliette 的信息。一个 cliette 是一个维护数据库连接的长时间运行的进程，或者是一个在多用户调用 Net.Data 宏期间持续存在的 Java 应用程序。cliette 启动之后，它就将一直存在，直至 Net.Data “现场连接”终止。多个 cliette 可以连接到单个数据库。

作为配置文件中 cliette 信息的一部分，您需要指定 cliette 名称、唯一的端口、以及进程的最小个数和最大个数。对于数据库 cliette，您还可以为每个 cliette 条目指定数据库

名称、注册以及口令。在 AIX 上，您可以加密口令。

权限技巧: 请确保启动连接管理器的用户 ID 对这个文件具有访问权。请参阅第39页的『指定对 Net.Data 文件的访问权』，以获取更多信息。

高速缓存管理器配置文件

高速缓存管理器配置文件中包含对高速缓存管理器和每个高速缓存的定义。Net.Data 高速缓存在第103页的『第9章 Net.Data 高速缓存』中描述。配置高速缓存管理器在第107页的『配置高速缓存管理器和 Net.Data 高速缓存』中描述。文件的结构是由一系列的段，或称节组成的:

高速缓存管理器节

这一节定义高速缓存管理器本身的参数，包括网络信息、记录状态以及跟踪状态。它需要而且必须标记为 `cache-manager`。

高速缓存定义节

这些节定义了每个高速缓存的参数；对于高速缓存管理器所管理的每个高速缓，在配置文件中都存在一个高速缓存定义节；这一部分包含网络信息、内存以及空间需求、记录状态和统计状态。对于高速缓存管理器所管理的每个高速缓存，都需要这个高速缓存定义节。

高速缓存管理器配置文件不是由管理工具来管理的，并且可以使用任何文本编辑器来进行修改。请参阅第103页的『第9章 Net.Data 高速缓存』，以学习如何定义这个文件。

权限技巧: 请确保启动高速缓存管理器的用户 ID 对这个文件具有访问权。请参阅第39页的『指定对 Net.Data 文件的访问权』，以获取更多信息。

Net.Data 初始化、控制和宏文件的公用节

为了能够使 Net.Data 的所有组件能够作为一个整体来工作，Net.Data 初始化、配置和宏文件中的某些部分必须一致。下表概述了这些文件中必须匹配的区域。

表 2. 对于 Net.Data 配置文件和宏文件的一致性需求

文件	公用节	注解
Net.Data INI 文件	环境语句	使用现场连接的语言环境必须在环境语句中指定数据库 cliette 名称
	现场连接配置变量	使用 Net.Data “现场连接”时，需要指定“现场连接”端口 DTW_CM_PORT。这个变量值必须与现场连接配置文件中的 MAIN_PORT 值相匹配。
	高速缓存配置变量	使用 Net.Data 高速缓存时，可以选择包含端口和机器变量。这些值必须和高速缓存管理器配置文件中所使用的那些值相匹配(如果使用的话)。
现场连接配置文件	Cliette 定义	每个 cliette 定义都必须与 INI 文件中相应的定义匹配。另外， MAIN_PORT 值必须与 INI 文件中的 DTW_CM_PORT 变量值匹配。

表 2. 对于 Net.Data 配置文件和宏文件的一致性需求 (续)

文件	公用节	注解
高速缓存管理器配置文件	高速缓存管理器配置变量	使用 Net.Data 高速缓存时，您可以选择包括端口或机器变量。这些值必须和 INI 文件中所使用的那些值相匹配(如果使用的话)。

以下几段说明了宏文件、初始化文件和现场连接配置文件之间的关系。两个 cliette 是由宏文件使用的 (DTW_SQL:SAMPLE、DTW_SQL:CELDIAL)，它们访问两个名为 SAMPLE 和 CELDIAL 的数据库。现场连接配置文件中包含 cliette 的名称和定义。 Net.Data 初始化文件中的 ENVIRONMENT 语句指 cliette 的名称。LOGIN 和 PASSWORD 的值是在“现场连接”配置文件中指定的。

第6页的图 2显示了包含 @DTW_ASSIGN 语句的宏文件段，该语句定义在访问数据库时使用哪个 cliette。

```
<*********************************************>
<* 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)
```

图 2. Net.Data 宏文件段

请注意，DATABASE 配置变量将替换 INI 文件的 ENVIRONMENT 语句，从而生成 cliette 的名称。这允许您从同一个宏文件访问多个数据库。

第7页的图 3显示了包含 ENVIRONMENT 语句和相关 cliette 类型的 Net.Data 初始化文件段。对于初始化文件中的每个 cliette 类型，都有一个 ENVIRONMENT 语句。对于每个数据库 cliette 类型， ENVIRONMENT 语句指定一个 cliette 名称。这个名称是由 cliette 的类型和一个变量引用组成的，$(DATABASE)，它在运行时被分辨。每个使用“现场连接”的语言环境都必须在 ENVIRONMENT 语句中有一个 cliette 的定义。

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

图 3. Net.Data 初始化文件段

第8页的图 4显示了一段现场连接配置文件，其中包含对 DTW_SQL:SAMPLE 和 DTW_SQL:CELDIAL 的 cliette 定义。

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

      ##################################################################
      # This is a comment in a Live Connection configuration file.
      # Comments start with a pound (hash) character.
      # Comments terminate at the end of the line and do not continue to
      # the next line unless another pound (hash) character is specified.
      # You can include comments at the end of lines containing Live
      # Connection keywords except on password lines.
      # You cannot include comments anywhere on lines containing the
      # password keyword.
      # You cannot include spaces and pound (hash) characters within any
      # name, such as cliette name or in database cliette passwords.
     ##################################################################
      CLIETTE  DTW_SQL:SAMPLE{
      MIN_PROCESS=1
      MAX_PROCESS=3
      START_PRIVATE_PORT=7100
      START_PUBLIC_PORT=7300
      EXEC_NAME=dtwcdb2.exe
      DATABASE=SAMPLE
            LOGIN=USER1
      PASSWORD=HAMLET
      }

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

图 4. 现场连接配置文件段

定制 Net.Data 初始化文件

包含在初始化文件中的信息是使用三类配置语句指定的，如以下章节所述:

- 第9页的『配置变量语句』
- 第13页的『定制路径配置语句』
- 第16页的『环境配置语句』

第9页的图 5中所示的示例初始化文件包含这些语句的例子，且对于 OS/2 和 Windows NT 是有效的。

```
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 path]   [Parameter list]
11 ENVIRONMENT (DTW_SYB)          [DLL path]   [Parameter list]
12 ENVIRONMENT (DTW_ORA)          [DLL path]   [Parameter list]
13 ENVIRONMENT (DTW_ODBC)         [DLL path]   [Parameter list]
14 ENVIRONMENT (DTW_DEFAULT)      [DLL path]   [Parameter list]
15 ENVIRONMENT (DTW_APPLET)       [DLL path]   [Parameter list]
16 ENVIRONMENT (DTW_REXX)         [DLL path]   [Parameter list]
17 ENVIRONMENT (DTW_PERL)         [DLL path]   [Parameter list]
18 ENVIRONMENT (DTW_SYSTEM)       [DLL path]   [Parameter list]
19 ENVIRONMENT (DTW_FILE)         [DLL path]   [Parameter list]
20 ENVIRONMENT (DTW_WEBREG)       [DLL path]   [Parameter list]
21 ENVIRONMENT (DTW_JAVAPPS)      [Parameter list]
22 ENVIRONMENT (DTW_DLDPB)        [Parameter list]
23 ENVIRONMENT (USR_TEST)         [Parameter list]
```

- 1 - 4 行，定义配置变量
- 5 - 9 行，定义到处理宏所需文件的路径
- 10 - 23 行，定义可用的环境语句。

图 5. Net.Data 初始化文件

以下章节将描述如何定制 INI 文件中的配置语句。

配置变量语句

Net.Data 配置变量语句设置配置变量的值。配置变量用于各种不同的目的。有些变量是语言环境所必需的，以便使它们能够正确地工作，或者以可以替代的方式操作。其它变量控制要构造的 Web 页面的字符编码或内容。另外，您可以使用配置变量语句来定义特定于应用程序的变量。

您所使用的配置变量取决于您所使用的语言环境和数据库，以及其它特定于应用程序的因素。

要更新配置变量语句: 使用您的应用程序所需的配置变量来定制初始化文件。配置变量具有以下语法:

NAME[=]*value-string*

等号是可选的，由方括号指示。

以下细目描述了您可以在初始化文件中使用的配置变量语句:

高速缓存管理器配置变量

如果高速缓存管理器不是在 Net.Data 宏所运行的机器上运行，则将使用两个可选的配置变量:

- CACHE_PORT 指定 Net.Data 使用哪个端口号码连接到高速缓存管理器。
- CACHE_MACHINE 指定本地或远程机器的 TCP/IP 主机名。

第10页的表 3描述了为这些变量指定机器 ID 和端口号码的选项。

表 3. 高速缓存管理器配置变量: 配置选项

缺省的连接管理器值	如果指定了高速缓存机器 ...	如果没有指定高速缓存机器 ...
如果指定了高速缓存端口...	Net.Data 使用指定的端口在指定的机器上连接到高速缓存管理器。	Net.Data 使用指定的端口在本地机上连接到高速缓存管理器。
如果没有指定高速缓存端口 ...	Net.Data 使用缺省的端口 7175 在指定的机器上连接到高速缓存管理器。	Net.Data 使用缺省的端口 7175 在本地机上连接到高速缓存管理器。

如果高速缓存管理器在本地机上运行，那么 UNIX 域套接字或已命名管道将用于通信，并且不需要进行配置。

高速缓存管理器只在 AIX 和 Windows NT 机器上运行。请参阅第103页的『第9章 Net.Data 高速缓存』，以了解 Net.Data 高速缓存。

CACHE_PORT: 高速缓存管理器端口变量

指定高速缓存管理器正在监听的 TCP/IP 端口。此端口号码必须与高速缓存管理器配置文件中指定的端口号码相匹配，这样 Net.Data 就可以与高速缓存管理器通信。如果没有指定，高速缓存管理器将使用缺省的端口 7175。

语法:

```
CACHE_PORT port_number
```

参数:

port_number

为高速缓存管理器分配的唯一的端口号码，用于服务高速缓存请求。缺省值为 7175。

CACHE_MACHINE: 高速缓存管理器机器 ID 变量

指定高速缓存管理器所驻留的机器。如果没有指定，Net.Data 将假定它就是本地机。

语法:

```
CACHE_MACHINE host_name
```

参数:

host_name

运行高速缓存管理器的本地或远程机器的限定 TCP/IP 主机名。缺省值是本地机的主机名。

DB2INSTANCE: DB2 实例变量

指定 SQL 语言环境所使用的 DB2 实例。当 Net.Data 连接到在 Windows NT、OS/2 和 UNIX 操作系统上运行的 DB2 时需要这个变量值。

OS/2、Windows NT 和 UNIX 操作系统上的 DB2 需要将 DB2INSTANCE 定义为一个环境变量。如果 Net.Data 检测到 DB2INSTANCE 没有定义为环境变量，那么它将把 DB2INSTANCE 环境变量设置为试图连接到 DB2 之前在 INI 文件中找到的 DB2INSTANCE 的值。

语法:

```
DB2INSTANCE instance_name
```

DTW_CM_PORT: 现场连接端口号码变量

指定 Net.Data 用于现场连接的唯一的端口号码。

语法:

```
DTW_CM_PORT port_number
```

其中 *port_number* 指定了用于现场连接的唯一的端口号码。

DTW_INST_DIR: Net.Data 安装目录变量

Net.Data INI 文件中的 DTW_INST_DIR 变量用于在 Net.Data 执行期间定位某些文件。您可以在安装时设置这个变量来指定主目录 <*inst_dir*> (Net.Data 安装在这个目录中)。安装之后不要更改这个值。

DTW_LOG_DIR: 错误记录位置变量

指定错误记录和 DTW_LOG_DIR 配置变量存储在哪里。在使用宏文件中的 DTW_LOG_LEVEL 变量启用了记录之后，记录文件将存储在 DTW_LOG_DIR 变量的路径语句中指定的目录中。缺省为 *\inst_dir*\logs\netdata.logs。请参阅第129页的『第11章 记录 Net.Data 错误信息』，以了解 Net.Data 的错误信息和 DTW_LOG_LEVEL 变量。

要求: 必须为 Net.Data 定义 DTW_LOG_DIR 变量，以便记录文件。如果没有定义，那么即使在宏文件中将 DTW_LOG_LEVEL 设置为 ERROR 或 WARNING 也不会进行记录。

语法:

```
DTW_LOG_DIR \inst_dir\path
```

例: 初始化文件配置

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

DTW_MBMODE: 本机的语言支持变量

对字处理和字符串函数激活国家语言支持。当这个变量的值为 YES 时，所有的字符串函数和字处理函数都将通过把字符串作为混合数据(即，作为可能同时包含来自单字节字符集和双字节字符集的字符的字符串)来正确地处理 DBCS 字符。缺省值为 NO。您可以通过在 Net.Data 宏文件中设置 DTW_MBMODE 变量来覆盖初始化文件中值的设置。

语法:

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

例: 启动国家语言支持

```
DTW_MBMODE = YES
```

要在宏文件中覆盖初始化文件设置:

- 将 DTW_MBMODE 变量作为初始化文件中 DEFAULT ENVIRONMENT 语句的 IN 参数，如下例所示:

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

- 在宏文件中，将配置变量 DTW_MBMODE 设置为应用程序所需的值。

DTW_OPTIMIZE_MATH: 优化数学函数变量

优化数学函数的性能。当 DTW_OPTIMIZE_MATH 设置成 YES 时，Net.Data 使用 C 数学格式化，函数能更快地运行；但是，输出格式和没有这个变量的格式不同。

- 特别地，十进制小数点之后的尾零不显示。
- 精度通常意味着有效位数。

如果 DTW_OPTIMIZE_MATH 设置成 NO，Net.Data 使用 REXX 数学格式。函数运行得较慢，但是输出格式和由先前版本的 Net.Data 生成的输出格式一致。缺省值是 NO。

语法:

```
DTW_OPTIMIZE_MATH NO|YES
```

要在宏文件中覆盖初始化文件设置:

在宏文件中，使用 DTW_ASSIGN 将配置变量 DTW_OPTIMIZE_MATH 设置为应用程序所需的值。

DTW_SMTP_SERVER: 电子邮件 SMTP 服务器变量

指定 SMTP 服务器用于发送电子邮件消息。这个变量的值可以是一个主机名，或是一个 IP 地址。如果没有设置这个变量，则 Net.Data 把本地主机用作 SMTP 服务器。

语法:

```
DTW_SMTP_SERVER server_name
```

其中 *server_name* 是用于发送电子邮件消息的 SMTP 服务器的主机名或 IP 地址。

例:

```
DTW_SMTP_SERVER = "myserver"
```

定制路径配置语句

Net.Data 从路径配置语句的设置中确定 Net.Data 文件所使用的文件和可执行程序的位置。路径语句有:

- MACRO_PATH
- EXEC_PATH
- INCLUDE_PATH
- FFI_PATH
- HTML_PATH

这些路径语句标识了一个或多个 Net.Data 在试图找出宏文件、文本文件LOB 文件 和包含文件时搜索的目录。您所需的路径语句取决于宏所使用的 Net.Data 的功能。

更新准则:

有些一般准则适用于所有路径语句。

- 每个指定的目录都由一个分号 (;) 分隔。
- 斜杠 (/) 和反斜杠 (\) 是同等对待的。
- 每个路径语句都可以指定多路径，只有 HTML_PATH 是例外，它只能有一个路径语句。路径是根据指定的顺序从左向右搜索的。这个多路径的功能可以让您在多个目录中组织文件。例如，您可以把每个 Web 应用程序放在它们自己的目录中。
- 建议您使用绝对路径语句。

技巧: Net.Data 搜索所有指定的目录，但不包括子目录。例如，如果以下目录中有 Net.Data 宏，则必须在路径语句中指定各子目录。

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

您的 MACRO_PATH 语句看起来可能是这样的:

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

以下章节将描述每个路径语句的目的和语法，并提供有效路径语句的示例。这些例子可能和您的应用程序不同，这取决于操作系统和配置。

MACRO_PATH

MACRO_PATH 配置语句标识了 Net.Data 搜索 Net.Data 宏文件的目录。例如，指定以下 URL 将请求带有路径和文件名 macro/sqlm.d2w 的 Net.Data 宏:

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

语法:

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

等号 (=) 是可选的，由方括号指出。

Net.Data 在 MACRO_PATH 配置语句中将路径 macro/sqlm.d2w 附加到路径后面，从左至右，直至 Net.Data 找到宏文件或搜索完所有路径。请参阅第45页的『第4章 调用 Net.Data』以获取有关调用 Net.Data 宏的信息。

例: 以下例子显示了初始化文件中的 MACRO PATH 语句以及调用 Net.Data 的相关链。

Net.Data 初始化文件:

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

HTML 链:

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

如果在目录 /u/SYSADM/macros 中找到文件 *query.d2w*，那么全限定路径就是 /u/SYSADM/macros/query.d2w。

EXEC_PATH

EXEC_PATH 配置语句标识了一个或多个目录，Net.Data 在其中搜索 EXEC 语句调用的外部程序或可执行变量。目录在路径语句中的顺序确定了 Net.Data 搜索目录的顺序。如果找到程序，则将外部程序名附加到路径说明后，形成一个传送到语言环境执行的全限定文件名。

语法:

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

例: 以下例子显示了初始化文件中的 EXEC PATH 语句以及调用外部程序的宏文件中的 EXEC 语句。

Net.Data 初始化文件:

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

Net.Data 宏:

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

如果在 /usr/lpp/netdata/prgms 目录中找到文件 *myFunction.cmd*，则程序的限定名为 /usr/lpp/netdata/prgms/myFunction.cmd。

INCLUDE_PATH

INCLUDE_PATH 配置语句标识了一个或多个 Net.Data 搜索的目录，以它们指定的顺序进行搜索，从而找到一个 Net.Data 宏中的 INCLUDE 语句所指定的文件。在找到这个文件之后，Net.Data 将把包含文件的名称附加到路径说明后面，以便产生限定的包含文件名。

语法:

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

技巧: 如果从本地的 Web 服务器包含 HTML 文件，可以使用 INCLUDE_URL 结构，如 *Net.Data 参考*中用于 INCLUDE_URL 的本地 Web 服务器示例中所示。通过使用演示的语法，您不必更新 INCLUDE_PATH 来指定 Web 服务器已知的目录。

例 1: 以下例子显示了初始化文件中的 INCLUDE_PATH 语句和指定包含文件的 INCLUDE 语句。

Net.Data 初始化文件:

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

Net.Data 宏:

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

如果在目录 /u/SYSADM/includes 中找到文件 *myInclude.txt*，则包含文件的全限定名称是 /u/SYSADM/includes/myInclude.txt。

例 2: 以下例子显示了 INCLUDE_PATH 语句和由子目录全限定的 INCLUDE 文件。

Net.Data 初始化文件:

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

Net.Data 宏:

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

包含文件是从目录 /u/SYSADM/includes/OE 和 /usr/lpp/netdata/includes/OE 中搜索的。如果文件在 /usr/lpp/netdata/includes/OE 中找到，则包含文件的全限定名称就是 /usr/lpp/netdata/includes/OE/oeheader.inc。

FFI_PATH

FFI_PATH 配置语句标识了一个或多个 Net.Data 搜索的目录，以它们指定的顺序进行搜索，从中搜索一个平面文件接口 (FFI) 函数引用的平面文件。

语法:

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

例: 以下例子显示了初始化文件中的 FFI PATH 语句。

Net.Data 初始化文件:

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

当调用 FFI 语言环境时，Net.Data 将查看 FFI_PATH 语句中指定的路径。

HTML_PATH

HTML_PATH 配置语句指定了 Net.Data 将大型对象 (LOB) 写入哪个目录。这个值可以通过配置初始化文件来更改。此路径语句只接受一个目录路径。

语法:

```
HTML_PATH [=] path
```

例: 以下例子显示了初始化文件中的 HTML PATH 语句。

Net.Data 初始化文件:

```
HTML_PATH = /pub/htm
```

当查询返回一个 LOB 时，Net.Data 将把它保存在 HTML_PATH 配置语句指定的目录中。

性能技巧: 在使用 LOB 时请考虑系统限度，因为它们会很块地消耗资源。请参阅第86页的『使用大型对象』，以获取更多信息。

环境配置语句

ENVIRONMENT 语句配置一个语言环境。语言环境是 Net.Data 的一个组件，Net.Data 用它来访问诸如 DB2 数据库等数据源，或者执行诸如 REXX 等语言编写的程序。Net.Data 提供了一系列语言环境，还提供了允许您创建自己的语言环境的接口。这些语言环境和语言环境接口在 *Net.Data 语言环境参考*中所有描述。

Net.Data 要求在您调用语言环境之前，该语言环境的 Net.Data 初始化文件中应该存在 ENVIRONMENT 语句。

Net.Data 指定了几个变量，它们影响 Net.Data 语言环境对 FUNCTION 块中定义的函数调用进行解释的方式。这些变量的设置必须传递到一个语言环境才能生效。

例如，宏可以定义一个 DATABASE 变量来指定一个数据库的名称，DTW_SQL 函数中的 SQL 语句将在此执行。DATABASE 的值必须传递到 SQL 语言环境 (DTW_SQL)，这样，SQL 语言环境就可以连接到指定的数据库。要将变量传递到语言环境，您必须向 DTW_SQL 的语言环境的参数列表中添加 DATABASE 变量。

示例 Net.Data 初始化文件对定制 Net.Data 语言环境配置语句的设置做了几个假设。这些假设对于您的环境来说可能是不正确的。请针对您的环境适当地修改这些语句。

要添加或更新 ENVIRONMENT 语句:

ENVIRONMENT 语句具有以下语法:

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

参数:

- *type*

  Net.Data 将此语言环境和 Net.Data 宏中定义的 FUNCTION 块相关联的名称。您必须在 FUNCTION 块定义中指定语言环境的类型，从而标识 Net.Data 要执行函数所应使用的语言环境。

- *library_name*

  DLL 或共享程序库的名称中包含 Net.Data 调用的语言环境接口。在 OS/2 中，指定 DLL 名称时是没有扩展名 *.dll* 的。在 AIX 中，共享对象的名称是使用 *.o* 来指定的。

- *parameter_list*

  在每个函数调用中传递给语言环境的参数列表(除了在 FUNCTION 块定义中指定的参数)。

  这些参数在 *dtw_lei* 结构的 *parm_data_array* 字段中传递，跟在 FUNCTION 块定义中指定的参数之后。(请参阅 *Net.Data 语言环境参考*以获取有关这些结构的信息。)

  在执行将由语言环境处理的函数之前，您必须将这些参数定义为配置变量或宏文件中的变量。如果一个函数修改了它的输出参数，那么在函数完成之后这些参数仍将保留修改后的值。

- *cliette_name*

  cliette 的名称。 *cliette_ name* 可以指 Java 应用程序语言环境 cliette，也可以是数据库 cliette。 *cliette_name* 参数是和 CLIETTE 关键字一起使用的，它们都只和 “现场连接” 一起使用。CLIETTE 和 *cliette_name* 是可选的，并且只能对数据库、Java 应用程序语言环境和用户定义的语言环境(具有为它们编写的 cliette) 指定。

  Java 应用程序 cliette

  这个 cliette 名称指定 Java 应用程序语言环境。

  语法:

  ```
  CLIETTE "DTW_JAVAPPS"
  ```

  数据库 cliette

  这个 cliette 名称指定一个与数据库相关联的 cliette。

  语法:

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

  参数:

  type 与 cliette 关联的数据库语言环境。请参阅第37页的，以获取有效类型的列表。

  db_name

  数据库 cliette 名称。这个名称通常与 cliette 所关联的数据库相同，例如 MYDBASE，但也可以是另一个名称。在使用 Oracle 语言环境时，*db_name*是可选的。

当 Net.Data 处理 INI 文件时，它不会装入语言环境 DLL 或共享程序库。Net.Data 在它首次执行标识某个语言环境的函数时装入该语言环境的 DLL。然后，只要 Net.Data 是装入的，该 DLL 就将保持装入状态。

例: 用于 Net.Data 提供的语言环境的 ENVIRONMENT 语句

在为您的应用程序定制 ENVIRONMENT 语句时，请在 ENVIRONMENT 语句中添加需要从初始化文件传递到语言环境的变量或 Net.Data 宏编写者需要在他们的宏中设置或覆盖的变量。

```
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 )
```

每个 ENVIRONMENT 语句必须在单独一行上。

配置现场连接

“现场连接”管理数据库和 Java 应用程序连接，从而改进 Net.Data 在 Windows NT、OS/2 和 UNIX 操作系统上的性能。通过使用连接管理器和维护开放连接的 cliette、进程，“现场连接”消除了连接到数据库或启动 Java 虚拟机的启动开销。

“现场连接”使用配置文件 dtwcm.cnf 来确定需要启动哪几个 cliette。它包含管理信息和对“现场连接”所使用的每个 cliette 的定义。请参阅第124页的『使用“连接管理”改进性能』来更多地了解“现场连接”。

第18页的图 6中所示的示例配置文件包含了以下类型的信息:

- 连接管理器端口信息
- 用于 DB2 连接的 SQL cliette 信息
- Java 应用程序 cliette 信息

```
 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  }
```

- 1 - 5 行，是配置文件所必需的，定义与“现场连接”一起使用的唯一的端口号码。
- 7 - 16 行，定义所有的数据库 cliette，标识了 cliette 名称、要运行的进程个数、数据库名称、端口号码以及 cliette 可执行文件。您可以包含一些附加信息，例如连接到 DB2 数据库的用户 ID 和口令。这些附加值显示在 13 - 15 行。
- 19 - 25 行，定义所有用于 Java 应用程序的 cliette，标识了 cliette 名称、要运行的进程个数、唯一的端口号码以及 cliette 可执行文件。

图 6. 现场连接配置文件

开始之前: 在定制现场连接配置文件之前，请先阅读执行步骤后面的提示和技巧部分。

要配置现场连接端口:

1. 用一个编辑器打开配置文件 dtwcm.cnf。
2. 配置三个现场连接端口号码:
   - MAIN_PORT
   - ADMIN_PORT1
   - ADMIN_PORT2

   第18页的图 6显示了缺省的端口号码。如果这些号码不唯一，则必须把它们更改为唯一的端口号码。
3. **重要事项**: 确保 MAIN_PORT 的值应与 Net.Data 初始化文件中 DTW_CM_PORT 的值相匹配。

要配置数据库 cliette:

1. 键入 cliette 环境语句。

   ```
   CLIETTE type:db_name
   ```

   参数:

   type 使语言环境和 cliette 关联的名称。请参阅第37页的，以获取有效类型的列表。

   db_name

   数据库 cliette 的名称，通常与 cliette 关联的数据库同名，例如 MYDBASE; 当然，*db_name* 也可以是另一个名称。在使用 Oracle 语言环境时，*db_name*是可选的。

2. 确定 MIN_PROCESS 和 MAX_PROCESS 的值。MIN_PROCESS 指定了启动连接管理器时要启动的进程个数。随后，如果同时到达其它的请求，则连接管理器将启动更多的 cliette，根据需要添加 cliette，直至到达为 MAX_PROCESS 指定的值。您所使用的这些值可能影响性能，但是您可以在今后对它们进行更改。

   键入 MIN_PROCESS 和 MAX_PROCESS 语句:

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

   参数:

   min_num

   在启动连接管理器时要启动的 cliette 进程的个数。对于这个数量的 cliette，您必须有足够多可用的、唯一的端口号码。

   max_num

   可以同时运行的 cliette 的最多个数。对于这个数量的 cliette，您必须有足够多可用的、唯一的端口号码。
3. 确定在您的系统上对数据库 cliette 使用哪些端口号码。这些号码必须是唯一的，以避免与高速缓存管理器或其它应用程序所使用的端口号码发生冲突。每个 cliette 使用两个端口。在指定一系列端口时，必须指定要使用的端口号码范围。前两个值是 START_PUBLIC_PORT 和 START_PRIVATE_PORT。另一个是 MAX_PROCESS，表示 cliette 的最多个数。以下例子显示了要使用哪些端口号码。

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

此例使用以下端口:

1000	1010
1001	1011
1002	1012
1003	1013
1004	1014

一个普遍的错误是两个 cliette 的集合所使用的端口号码重叠，或者与高速缓存管理器的端口号码重叠。请与系统管理员一起进行检查，以确保您计划使用的端口号码是可用的。针对您的操作系统的自述文件中有关于这个操作系统中哪些端口号码可用的一般准则。

4. 指定 cliette 可执行文件的名称。此文件名的指定如下:

   ```
   EXEC_NAME=./dtwcxxx
   ```

   其中 *xxx* 是数据库类型标识符。请参阅第20页的表 4，以获取有效的可执行文件名:

表 4. Cliette 可执行文件名

Cliette 描述	Cliette 类型	名称		平台有效性					
		UNIX	Windows NT 或 OS/2	AIX	NT	OS/2	HP	SUN	SCO
DB2 进程 cliette	DTW_SQL	dtwcdb2	dtwcdb2.exe	是	是	是	是	是	否
ODBC 进程 cliette	DTW_ODBC	dtwcodbc	dtwcodbc.exe	是	是	否	否	否	否
Sybase 进程 cliette	DTW_SYB	dtwcsyb	dtwcsyb.exe	是	是	否	否	否	否
Oracle 进程 cliette	DTW_ORA	dtwcora	dtwcora.exe	是	是	否	否	否	否

5. 指定 cliette 所关联的数据库的名称:

   ```
   DATABASE=db_name
   ```

   其中 *db_name* 是 cliette 所关联的数据库的名称; 例如 MYDBASE。

6. 可选: 更改 LOGIN 和 PASSWORD 变量的缺省值，这样，Net.Data 就可以使用启动连接管理器时使用的用户 ID 连接到 DB2 数据库。通过指定这些缺省值，您可以避免将这一信息放在配置文件中。例如，用以下几行来代替第18页的图 6中示例配置文件中的 14 和 15 行:

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

   技巧: 如果您在配置文件中定义多个 cliette 条目，则您可以对某个特定的数据库指定不同的数据库注册和口令。

要配置 Java 应用程序 cliette:

1. 键入 cliette 环境语句:

```
CLIETTE DTW_JAVAPPS
```

2. 确定 MIN_PROCESS 和 MAX_PROCESS 的值。MIN_PROCESS 指定了启动连接管理器时要启动的进程个数。随后，如果有同时的进程到达，则连接管理器将启动更多的 cliette，根据需要添加 cliette，直至到达为 MAX_PROCESS 指定的值。您所使用的这些值可能影响性能，但是您可以在今后对它们进行更改。

   键入 MIN_PROCESS 和 MAX_PROCESS 语句。

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

   参数:

   min_num

   在启动连接管理器时启动的 cliette 进程的个数。对于这个数量的 cliette，您必须有足够多可用的、唯一的端口号码。

   max_num

   可以同时运行的附加 cliette 的最多个数。对于这个数量的 cliette，您必须有足够多可用的、唯一的端口号码。

3. 确定在您的系统上对数据库 cliette 使用哪些端口号码。这些号码必须是唯一的，以避免与高速缓存管理器或其它应用程序所使用的端口号码发生冲突。每个 cliette 使用两个端口。在指定一系列端口时，必须指定要使用的端口号码范围。前两个值是 START_PUBLIC_PORT 和 START_PRIVATE_PORT。另一个是 MAX_PROCESS，表示 cliette 的最多个数。以下例子显示了要使用哪些端口号码。

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

   此例使用以下端口:

1000	1010
1001	1011
1002	1012
1003	1013
1004	1014

   一个普遍的错误是两个 cliette 的集合所使用的端口号码重叠，或者与高速缓存管理器的端口号码重叠。请与系统管理员一起进行检查，以确保您计划使用的端口号码是可用的。针对您的操作系统的自述文件中有关于这个操作系统中哪些端口号码可用的一般准则。

配置“现场连接”的提示和技巧:

- 连接管理器使用 cliette 名称来唯一地标识一系列 cliette。
- 对于数据库 cliette，您必须确保对每个计划访问的数据库都有一个已命名的 cliette 集合。对于很少访问的数据库，您可以将 cliette 的 MIN 和 MAX 个数设置为 1。同样，您还可以将 MIN 设置为 0，这意味着在对 cliette 进行 Net.Data 请求之前不会启动进程。
- cliette 的 NAME 必须与初始化文件中用于 cliette 类型的 ENVIRONMENT 语句所引用的 cliette 名称一致。cliette 名称中可以包含变量，如果是 DB2 cliette，那么它应该包含变量引用 $(DATABASE)。ENVIRONMENT 语句中用于 cliette 名称的缺省值是 DTW_SQL:$(DATABASE)。您可以在 INI 文件中使用变量引用，但不能在现场连接配置文件中使用变量引用。

DATABASE 变量是在 Net.Data 宏文件中定义的。当宏文件中遇到 SQL 语句时，Net.Data 初始化文件中的 $(DATABASE) 变量引用将被 DATABASE 的当前值代替。

您可以使用这个方法来访问多个数据库。如果您想要在 Net.Data 宏中访问三个数据库(例如，D1、D2 和 D3)，并且初始化文件中有标准的 CLIETTE ″DTW_SQL:$(DATABASE)″ 行，那么您在配置文件中需要这样三个部分:

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

- 进程被启动，但不停止。如果您将最大进程个数设置为 M，并且任意时刻 M 个进程是同时使用的，在关闭连接管理器之前它们将保持活动状态，这样您就不希望 MAX_PROCESS 的值这么高，为了启动很少使用的进程而用尽了所有的系统资源。

  建议: 请尝试对 MIN_PROCESS 和 MAX_PROCESS 使用不同的值，看看哪一个能在您的系统上发挥最佳性能。如果连接管理器接收到的请求超过了指定的最大值，最后一个请求将被排队，直至某个 cliette 完成了它的处理。当有一个 cliette 可用时，排队的那个请求将被处理。这种将请求排队的过程对于应用程序用户来说是透明的。
- 您可以对不同名称的部分使用同一个 cliette。例如，配置文件中所有的 DB2 数据库部分都使用相同的 cliette 类型。但是两个部分不能有相同的名称。

如果您使用 CGI，并且只希望几个数据库使用 “现场连接”，则只要在配置文件中列出您所希望的数据库即可。如果 Net.Data 在处理 Net.Data 宏时遇到了 SQL 部分，它将向连接管理器询问某个特定的 cliette。如果连接管理器没有该类型的 cliette，它将以一个 NO_CLIETTE_AVAIL 消息作为应答。然后，Net.Data 用一个 DLL 版本来处理该请求。

要自动将连接管理器作为 Windows NT 的服务启动:

在 Windows NT 上，您可以指定将连接管理器作为 Windows NT 的服务启动，而不是从命令行启动。将连接管理器作为 Windows NT 的服务运行可以使连接管理器在每次启动机器时自动启动。

技巧: 在将连接管理器设置为自动启动之前，先从命令行启动，以确保现场连接配置文件是正确的。

- 从 Windows NT 任务栏，选择**开始->设置->控制面板->服务**。
- 选择 **Net.Data 连接管理器**，然后单击**启动**按钮。
- 选择**自动启动类型**，然后单击**确定**。

为 FastCGI 配置 Net.Data

FastCGI 允许 Net.Data 在 Apache Web Server 和 Domino Go Webserver (IBM Internet Connection Secure Server (ICSS) 的改进型产品)上以 FastCGI 方式运行。FastCGI 方式提供了与其它 Web API 程序类似的性能，还提供了 CGI-BIN 程序的可靠性(分离的内存空间)。

开始之前:

在使用 FastCGI 之前，请确保您已经安装了先决产品:

- **对于 Apache**: 下载并安装 Apache Web Server 1.2.0 或更高版本。

- **对于 Domino Go Webserver**: 从以下站点下载并安装 Domino Go Webserver for AIX:

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

要为 FastCGI 配置 Net.Data:

1. 为您的操作系统配置 Web 服务器和 FastCGI 配置文件:

   对 Apache Web server:

   更新 http.conf 文件。

   - 声明新的应用程序:

     ```
     AppClass inst_dir
     -processes proc_num
     -initial-env LIBPATH=libpath
     -initial-env ORACLE_HOME=oracle_path
     -initial-env ORACLE_SID=oracle_instance
     -initial-env SYBASE=sybase_path
     -initial-env DSQUERY=sybase_instance
     -initial-env DB2INSTANCE=db2_instance
     -initial-env RXQUEUE_OWNER_PID=REXX_perf_var
     -initial-env LANG=locale
     ```

   - 声明 FastCGI 模块:

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

   对于 AIX 上的 Domino Go Webserver:

   更新 httpd.conf 和 fcgi.conf 文件:

   - 在 httpd.conf 文件中，声明服务部分:

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

   - 在 fcgi.conf 文件中，声明应用程序:

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

   参数:

   inst_dir

   Net.Data 的可执行文件所使用的路径和目录名。

   对于 Apache:

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

对于 Domino Go Webserver:

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

Role Responder

Domino Go Webserver 所必需的关键字(只有 Domino Go Webserver 需要)。

URL Domino Go Webserver 所必需的关键字和 URL 地址(只有 Domino Go Webserver 需要)。这个 URL 指向为 EXEC_PATH 语句指定的路径。

BindPath

Domino Go Webserver 所需的关键字和路径语句(仅在 AIX 上)。 Net.Data 和 FastCGI 所使用的唯一的 UNIX 套接字的路径。

proc_num

可以同时处理的请求个数。缺省值为 1，但应该根据应用程序的要求增加这个数值以改进性能。请参阅第123页的『使用 FastCGI 来改进性能』以获取调节信息。

对于 Apache:

```
-processes 7
```

对于 ICS 或 Domino Go Webserver:

```
NumProcesses 7
```

libpath 在 Net.Data INI 文件的每个 ENVIRONMENT 语句中声明的 LIBPATH (共享程序库或 DLL) 语句。

对于 Apache:

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

对于 Domino Go Webserver:

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

oracle_path

在使用 Oracle 时所必需。Oracle 数据库可执行文件的路径和目录。

对于 Apache:

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

对于 Domino Go Webserver:

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

oracle_instance

在使用 Oracle 时所必需。Oracle 数据库的实例。您必须对 Oracle 使用现场连接。

对于 Apache:

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

对于 Domino Go Webserver:

```
Environ ORACLE_SID=mvpdb2
```

sybase_path

在使用 Sybase 时所必需。Sybase 数据库可执行文件的路径和目录。

对于 Apache:

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

对于 Domino Go Webserver:

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

sybase_instance
在使用 Sybase 时所必需。Sybase 数据库的实例。您必须对 Sybase 使用现场连接

对于 Apache:

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

对于 Domino Go Webserver:

```
Environ DSQUERY=SybaseAIX
```

db2_instance
在使用 DB2 时所必需。DB2 数据库的实例。

对于 Apache:

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

对于 Domino Go Webserver:

```
Environ DB2INSTANCE=wwwinst
```

REXX_perf_var
在 AIX 上使用 REXX 时所必需。这个性能变量是在 AIX 操作系统上和 FastCGI 以及 REXX 一起使用的。缺省值为 0。对于其它的产品和操作系统，则在 Net.Data 宏文件中声明这个变量。请参阅 *Net.Data 参考*中的附录『Net.Data for AIX』，以获取更多有关此变量的信息。

对于 Apache:

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

对于 Domino Go Webserver:

```
Environ RXQUEUE_OWNER_PID=0
```

locale UNIX 场所变量。对于美国英语使用 En_US。

对于 Apache:

```
-initial-env LANG=En_US
```

对于 Domino Go Webserver:

```
Environ LANG=En_US
```

2. **对于 Apache**: 在 srm.conf 文件中添加 fgi-bin 目录，作为新的脚本别名:
   ```
   ScripAlias /fcgi-bin/ /u/mydir/apache/fci-bin
   ```
3. 将所有静态或动态生成的 Web 页面从 CGI-BIN 移植到 FCGI-BIN。例如:
   ```
   <A HREF="http://server/fcgi-bin/db2www/filename.ext/block/
   [?name=val&...]">any text</A>
   ```
4. 用 FCGI-BIN 代替 CGI-BIN 来修改用于 Net.Data 的 URL 调用的最终用户文档。例如:
   ```
   http://server/fcgi-bin/db2www/filename.ext/block/[?name=val&...]
   ```

配置 Net.Data 以便与 Web 服务器 API 一起使用

使用 Web 服务器应用程序设计接口 (API) 而不是 CGI 可以很大地改进 Net.Data 的性能。Net.Data 支持以下服务器 API:

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

有关每个 API 的更多信息，请参阅第121页的『使用Web 服务器 API 改进性能』以及您所使用的 Net.Data 版本的自述文件。

要求: 要以 ICAPI、GWAPI、ISAPI 或 NSAPI 方式运行 Net.Data，您必须重新配置 Web 服务器，以便将 Net.Data DLL 或共享程序库用作它的服务命令。重新配置之后，您必须重新启动 Web 服务器以使 Net.Data 初始化文件的更改生效。缺省情况下，Net.Data 以 CGI 方式运行。

以下章节描述了如何配置 Net.Data 和 Web 服务器以运行 Web 服务器 API 方式。其中提供了一般步骤和例子，但它们可能和您的操作系统有所不同。请参阅针对您所使用的操作系统的 Net.Data 的自述文件，以获取特定的指导。

要配置 ICAPI 和 GWAPI:

Domino Go Webserver 是 IBM Internet Connection Secure Server 的改进型产品。如果要升级，您可能希望使用新的 Domino Go Webserver。请注意，GWAPI 和 ICAPI 是相同的产品，只是重新命名之后用来标识使用的是哪一个 Web 服务器。

1. 停止 Web 服务器。
2. 请确保 ICAPI 或 GWAPI DLL 或共享程序库位于服务器的 `CGI-BIN` 或 `ICAPI-LIB` 目录中。

   请参阅针对您所使用的操作系统的 Net.Data 的自述文件或程序目录，以获取特定的文件和目录名称。
3. 在 Web 服务器的配置文件 (`httpd.conf` 或 `httpd.cnf`) 中添加一个服务语句来调用 API。

   例如:

   ```
   Service /cgi-bin/db2www*   /usr/lpp/netdata/icapi-lib/db2www:dtw_icapi*
   ```

   请参阅针对您所使用的操作系统的 Net.Data 的自述文件，以获取特定的文件和目录名称。
4. 重新启动 Web 服务器。

ICAPI 和 GWAPI 具有完全的兼容性来支持现有的应用程序。使用与 CGI 相同的方法来调用 URL 和表，或者与 ICAPI 或 GWAPI 相链接。使用 CGI 成功执行的宏也可以使用 ICAPI 或 GWAPI 来成功执行。不需要对这些宏作任何修改。

要配置 ISAPI:

1. 停止 Web 服务器。
2. 将 Net.Data 所附带的、用于 ISAPI 的 DLL 复制到服务器的子目录中。例如:

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

其中的 *filetype* 对于 Windows NT 和 OS/2 来说是 .dll，对于 UNIX 操作系统来说则是 .o。

请参阅针对您所使用的操作系统的 Net.Data 的自述文件，以获取特定的文件和目录名称。

3. 因为 ISAPI 绕过了 CGI 处理，因此在表和链接中不需要 URL 的 cgi-bin/db2www/ 部分。相反，需要使用 dtwisapi.*filetype*。例如，如果以下 URL 将 Net.Data 作为 CGI 程序调用:

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

   那么您应使用以下 URL 将 Net.Data 作为 ISAPI 插件来调用:

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

4. 如果您将宏 test1.d2w 存储在子目录 /order/ 中(在 MACRO_PATH 中指定的某个目录或 Web 服务器的当前目录下)，则使用以下 URL 在 CGI 方式中调用 Net.Data:

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

   那么，在 ISAPI 方式中调用 Net.Data 的等价的 URL 是:

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

5. 重新启动 Web 服务器。

要配置 NSAPI:

1. 停止 Web 服务器。
2. 将 Net.Data 所附带的、用于 NSAPI 的 DLL 复制到服务器目录中。例如:

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

   其中的 *filetype* 对于 Windows NT 和 OS/2 来说是 .dll，对于 UNIX 操作系统来说则是 .o。

   请参阅针对您所使用的操作系统的 Net.Data 的自述文件，以获取特定的文件和目录名称。

3. 使用下面列出的更改来修改您的服务器配置文件。请参阅针对您所使用的操作系统的 Net.Data 的自述文件或程序目录，以了解操作系统之间的区别。

   obj.conf　添加到文件开头:

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

   obj.conf　添加到服务命令:

   ```
   Service fn="dtw_nsapi" method=(GET|HEAD|POST)
     type="magnus-internal/d2w"
   ```

   mime.types　添加此类型，其中 *d2w* 是宏文件的缺省扩展名。您可以指定任意的三个字符的组合。

   ```
   type=magnum-internal/d2w exts=d2w
   ```

4. 将 Net.Data 宏文件从 netdata/macro 目录移动到服务器的根文档目录:

   ```
   /netscape/server/docs/
   ```

5. 将服务器的根文档目录添加到初始化文件的 MACRO_PATH 语句中。这个更改告诉 Net.Data 在哪里查找宏文件。

6. 因为 NSAPI 绕过了 CGI 处理，因此在表和链接中不需要 URL 的 cgi-bin/db2www/ 部分。服务器知道具有 d2w 文件类型的文件是 Net.Data 宏，因为您在更改 Netscape 配置文件时对它进行了定义。例如，以下 URL 将 Net.Data 作为 CGI 程序调用:

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

   而以下 URL 将 Net.Data 作为 NSAPI 程序调用:

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

7. 重新启动 Web 服务器。

如果将 Net.Data 宏放在几个目录中，那么最后的三个步骤改为:

1. 将目录和其中包含的 Net.Data 宏移动到服务器的根文档目录。
2. 更新初始化文件中的 MACRO_PATH 变量来包括宏文件所在的所有目录和子目录。
3. 修改指向这些 Net.Data 宏的链接和表，保留它们的目录名。例如，在 CGI 方式中运行时，以下 URL 将调用一个存储在 /orders/ 目录中的 Net.Data 宏:

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

   经过修改后，在 NSAPI 方式中调用 Net.Data 的 URL 要短一些，但仍保留了目录名:

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

使用 Net.Data 管理工具来配置 Net.Data

Net.Data 管理工具帮助您在 Windows NT、AIX 和 OS/2 操作系统上配置和管理 Net.Data 初始化文件 (DB2WWW.INI) 以及用于 “现场连接” 的初始化文件 (DTWCM.CNF)。使用这一工具之后，您可以完成以下任务:

- 第29页的『启动管理工具』
- 第29页的『配置路径语句』
- 第31页的『配置端口』
- 第31页的『配置 Cliette』
- 第35页的『配置语言环境』
- 第38页的『定义配置变量』

请参阅第28页的『开始之前』以学习如何设置管理工具以及如何确保您具有正确的软件先决条件。

开始之前

1. 计划对 Net.Data 语言环境、数据库、cliette、端口和配置变量的配置。
2. 从 CD-ROM 安装 Net.Data。
3. 安装 Java 运行时程序库(各操作系统的 JDK 1.1 及后继版本)。请查看针对您所使用的操作系统的 Net.Data 的自述文件，以获取更多信息。

   请确保安装 JDK 之后在您的 CLASSPATH 中有classes.zip。
4. 如果已经安装了与 DB2 Universal Database 封装在一起的 IBM JDBC 驱动程序，则将该驱动程序目录添加到 Java CLASSPATH 语句中，以便启用 DB2 注册测试。
5. 更改到存储 Net.Data 管理工具程序的目录:

对于 OS/2 和 Windows NT:

inst_dir\connect*admin_directory*，其中 *inst_dir* 是安装期间为 Net.Data 指定的目录，而 *admin_directory* 是管理工具文件所在的目录。

对于 AIX:

/usr/lpp/internet/db2www/db2.v2/*admin_directory*，其中 *admin_directory* 是管理工具文件所在的目录

启动管理工具

您所使用的操作系统决定了您应如何启动管理工具。

对于 OS/2 和 Windows NT:

从 IBM Net.Data 版本 2 的文件夹中选择 **Net.Data 管理工具**图符。

对于 AIX:

更改到 Net.Data 的安装目录 (inst_dir)。从命令行输入 ndadmin 来启动该工具。

管理工具被启动，并显示 Net.Data 管理笔记本。

配置路径语句

使用**路径**页面来添加、修改或删除用于定位处理 Net.Data 宏所需文件的路径语句。这些语句在第13页的『定制路径配置语句』中有所描述。第30页的图 7显示了**路径**页面。

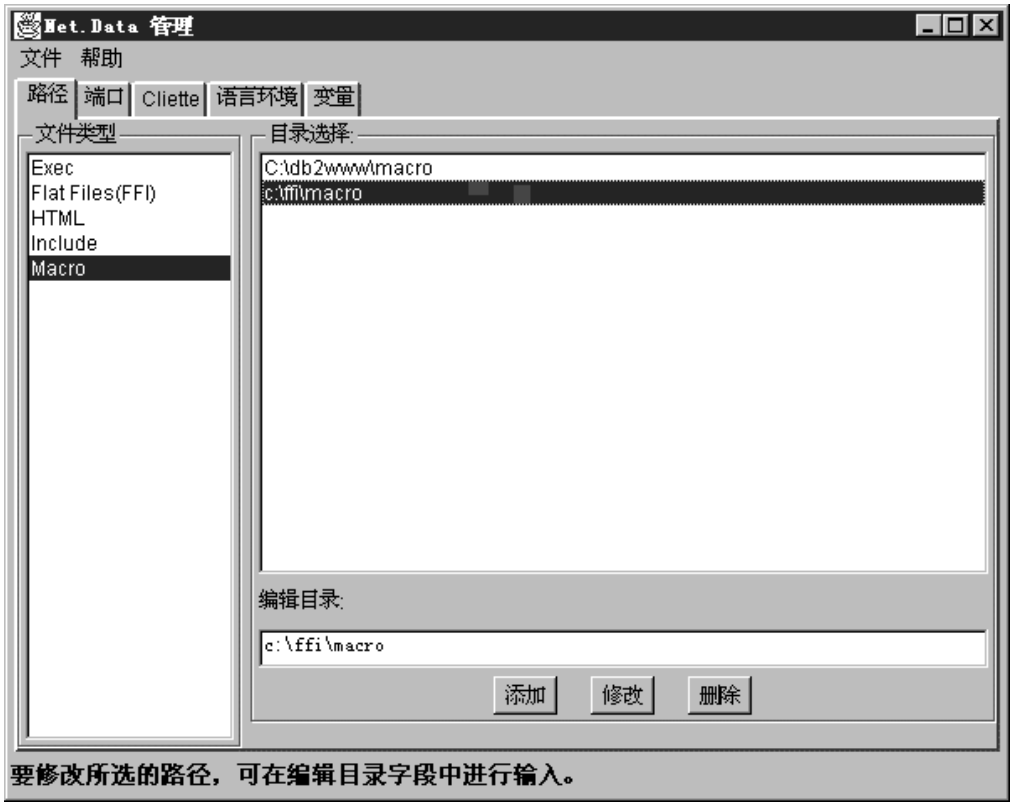


图 7. Net.Data 管理工具的路径页面. 使用这个页面来添加、修改或删除路径语句。

配置技巧: HTML 文件类型只有一个路径。

要添加一个路径语句:

1. 启动管理工具。
2. 在**路径**页面中，从**文件类型**中选择一个文件类型，例如，选择 Exec。
3. 在**编辑目录**字段中，键入新的路径并单击**添加**按钮。

   如果指定的路径不存在，则将显示一个警告窗口。如果没有选择目录，则新的目录将作为最后一项添加至列表中。
4. 关闭管理工具，或单击另一个标签来完成另外的配置任务。

要修改一个路径语句:

1. 启动管理工具。
2. 在**路径**页面中，从**文件类型**列表中选择您想要更改的文件类型。
3. 在**目录选择**列表中选择您想要修改的路径。选定的路径将在**编辑目录**字段中打开。
4. 修改**编辑目录**字段中的路径并单击**修改**按钮。如果输入的路径不存在，则将显示一个警告窗口。
5. 关闭管理工具，或单击另一个标签来完成另外的配置任务。

要删除一个路径语句:

1. 启动管理工具。
2. 在**路径**页面中，从**文件类型**列表中选择您想要删除的文件类型。

3. 在**目录选择**字段中，选择您想要删除的路径。选定的路径将在**编辑目录**字段中打开。
4. 单击**删除**按钮。
5. 关闭管理工具，或单击另一个标签来完成另外的配置任务。

配置端口

使用**端口**页面来指定 Net.Data 所使用的 TCP/IP 端口号码。第31页的图 8显示了**端口**页面。

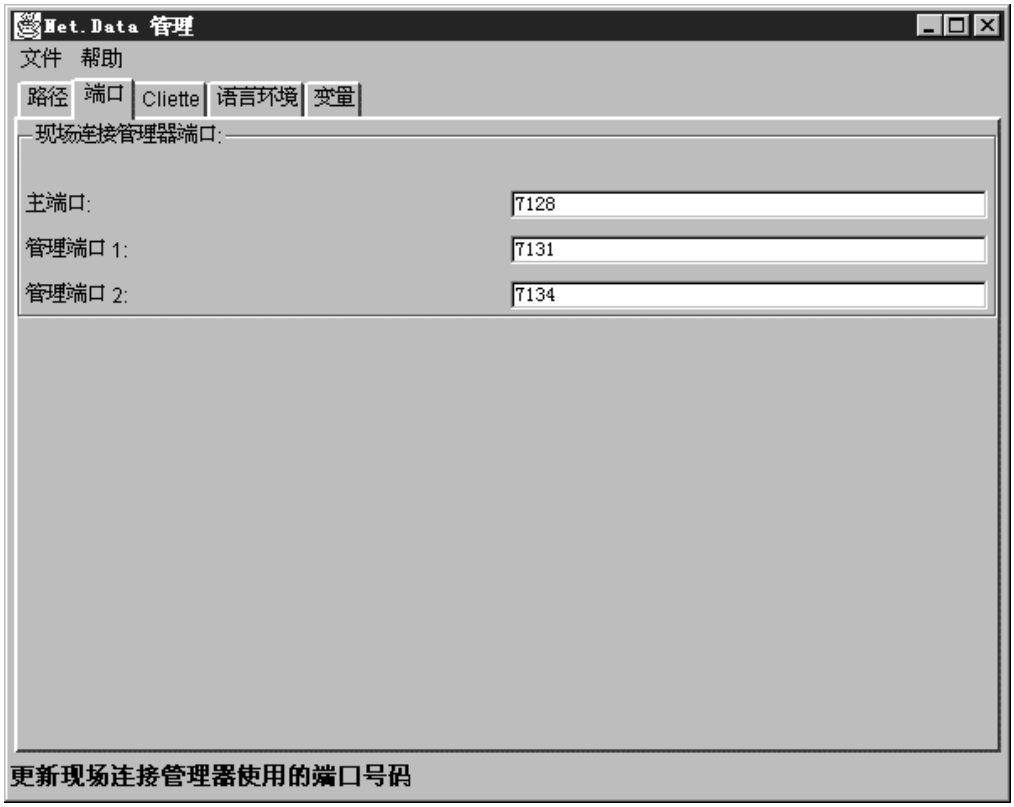


图 8. Net.Data 管理工具的端口页面. 使用这个页面来指定端口。

要指定 TCP/IP 端口号码:

1. 启动管理工具。
2. 在**端口**页面中，在每个端口字段中键入唯一的端口号码。在您将光标移动到下一个字段时，管理工具将验证您在每个字段中键入的端口号码。
3. 关闭管理工具，或单击另一个标签来完成另外的配置任务。

配置 Cliette

使用 **Cliette** 页面来添加、修改或删除“现场连接”数据库 cliette，您还可以管理数据库和用于 cliette 的管理员用户 ID 和口令。第124页的『使用“连接管理”改进性能』中提供了有关 cliette 的更多信息。第32页的图 9显示了 **Cliette** 页面。

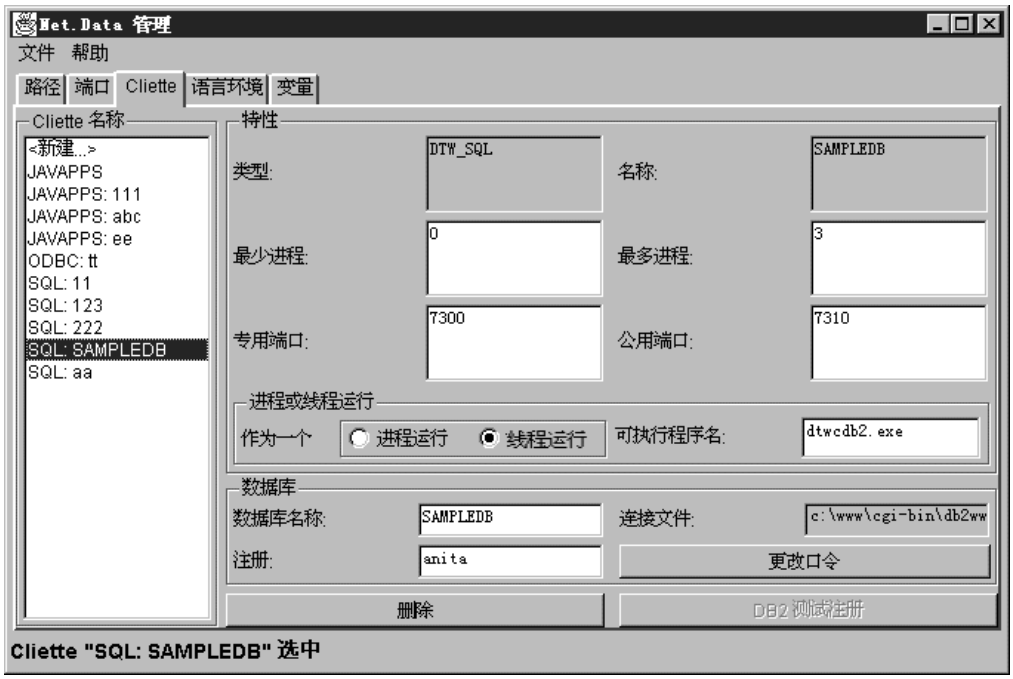


图 9. Net.Data 管理工具的 Cliette 页面. 使用这个页面来添加、修改和删除 cliette。

要添加一个 cliette:

1. 启动管理工具。
2. 在 **Cliette** 页面中，从 **Cliette 名称**列表中选择**<新建...>**。将打开**添加一个 cliette** 窗口。

图 10. Net.Data 管理工具的添加一个 Cliette 页面. 使用这个页面来添加 cliette。

如果启用了加密，那么在您第一次创建或修改 cliette 时将提示您输入加密口令。这个口令将被保存，以后就不需要再输入了。

3. 从**类型**列表中选择一个 cliette 类型。
4. 在**名称**字段中为新的 cliette 键入一个名称。这个名称可以是数据库的名称，也可以是另一个唯一的 cliette 名称。例如: MYCLIETTE。
5. 如果启用了**加密口令**字段，则键入加密口令。您不需要再次输入这个口令，因为管理工具将为您保存该口令。
6. 单击**添加**按钮。

   新的 cliette 将被创建并添加到 cliette 列表底部。另外，新的名称是突出显示的，cliette 的缺省特性将显示在**特性**组框中。您可以更改这些值来与您的配置相匹配。
7. 关闭管理工具，或单击另一个标签来完成另外的配置任务。

要修改一个 cliette:

1. 启动管理工具。
2. 在 **Cliette** 页面中，从 **Cliette 名称**列表中选择您想要更改的 cliette 名称。cliette 的特性显示在**特性**组框中。
3. 根据需要在**特性**组框中修改特性。
   a. **类型**字段显示了要定义的 cliette 的类型，且对应于一个语言环境类型名称。Net.Data 在您添加新的 cliette 时填充此字段，选项是在“添加一个 Cliette”窗口中的 **Cliette 类型**中定义的。
   b. **名称**字段显示了 cliette 名称，这通常是数据库的名称。Net.Data 在您添加新的 cliette 时填充此字段。
   c. 在**最少进程**字段中键入启动连接管理器时可以启动的 cliette 进程的个数。对于每个进程，您都需要一个唯一的端口地址。请参阅第18页的『配置现场连接』，以获取有关“最少进程”值的更多信息。
   d. 在**最多进程**字段键入可以同时运行的 cliette 进程的个数(除了启动连接管理器时启动的进程以外)。对于每个进程，您都需要一个唯一的端口地址。请参阅第18页的『配置现场连接』，以获取有关“最多进程”值的更多信息。
   e. 在**专用端口**字段中键入唯一的端口号码，以指定与连接管理器同时启动的 cliette 进程一起使用时的启动端口号码。对于**最少进程**值所指定的每个进程，都将使用一个附加端口号码。例如如果您对**专用端口**指定端口号码 7012，对**最少进程**指定值 5，则将使用端口号码 7012-7016，并且绝对不能与系统中其它的端口分配发生冲突。
   f. 在**公用端口**字段中键入唯一的端口号码，以指定与附加进程同时启动的 cliette 进程一起使用时的启动端口号码，最多为**最多进程**字段中指定的个数。对于每个进程都使用了一个附加端口号码。例如，如果您对**公用端口**指定端口号码 7020，对**最多进程**指定值 5，则将使用端口号码 7020-7024，并且绝对不能与系统中其它的端口分配发生冲突。
   g. **可执行程序名**字段显示了 cliette 可执行文件的名称。
4. 如果 cliette 要与数据库一起使用，则根据需要修改**数据库**组框的值:
   a. 指定 cliette 所关联的数据库的名称(在**数据库名称**字段中)，例如，MYDBASE。
   b. **连接文件**字段包含了用于您所使用的 cliette 类型的连接文件的名称和路径。
   c. **注册**字段指定了用于连接到数据库的注册用户 ID。

d. **更改口令**按钮将打开“更改数据库口令”窗口。键入加密口令和新的口令(两次)。您可以通过使用**安全性**下拉菜单中指定的加密功能来加密数据库口令。

5. 选择**文件**，然后选择**保存**来保存您所作的更改。
6. 关闭管理工具，或单击另一个标签来完成另外的配置任务。

要测试 DB2 数据库注册和连接:

1. 在管理工具的 **Cliette** 页面上单击 **DB2 测试注册**按钮。测试完成之后将打开一个确认窗口，显示连接测试的状态。
2. 关闭此窗口继续配置，或关闭管理工具。

要删除一个 cliette:

1. 启动管理工具。
2. 在 **Cliette** 页面中，从 **Cliette 名称**列表中选择您想要删除的 cliette 名称。
3. 单击**删除**按钮。
4. 关闭管理工具，或单击另一个标签来完成另外的配置任务。

要启用对 cliette 用户 ID 和口令的加密:

加密为使用 cliette 的数据库连接提供了安全性。启用加密的时候，现场连接配置文件中的所有数据库口令均被加密，并且在访问和解密的时候需要一个加密口令。

要求: 必须使用 Net.Data 版本 2 现场连接配置文件才能使用加密。

1. **重要事项**: 对现场连接配置文件 <path>dtwcm.cnf进行备份。如果您丢失了加密口令，或者想要解密数据库口令并恢复口令，就需要这个文件。
2. 在管理工具的 **Cliette** 页面上，选择**安全性 -> 启用加密**下拉菜单选项。将打开“启用加密”确认窗口。
3. 单击**是**继续。将打开“加密口令”窗口。
4. 对于使用具有加密口令的 cliette 的权限输入口令(两次)。
5. 单击**确定**定义新的口令并对 cliette 加密所有的数据库口令。

要关闭对 cliette 用户 ID 和口令的加密:

1. 在管理工具的 **Cliette** 页面上，选择**安全性 -> 关闭加密**下拉菜单选项。将打开“关闭加密”确认窗口。
2. 单击**是**继续。出于安全性的原因，所有的口令都设置为 *USE_DEFAULT。您可以从“现场连接”文件的备份 <path>dtwcm.cnf 中恢复口令。

要更改加密口令:

1. 在管理工具的 **Cliette** 页面上，选择**安全性 -> 更改加密口令**下拉菜单选项。将打开“更改加密口令”确认窗口。
2. 单击**是**继续。将打开“更改加密口令”窗口。
3. 键入原来的口令一次，新的口令两次。
4. 单击**确定**更改加密口令。

要更改数据库口令:

1. 在管理工具的 **Cliette** 页面中，单击**更改口令**按钮。将打开“更改数据库口令”窗口。
2. 键入加密口令一次，新的数据库口令两次。

3. 单击**确定**更改口令并关闭窗口。如果您启用了加密，则更改后的数据库口令将被加密。

配置语言环境

使用**语言环境**页面来添加、修改或删除 Net.Data 语言环境。语言环境在第16页的『环境配置语句』中讨论。第35页的图 11显示了**语言环境**页面。

图 11. Net.Data 管理工具的语言环境页面. 使用这个页面来指定语言环境。

要添加一个语言环境:

1. 启动管理工具。
2. 在**语言环境**页面中，从**语言环境**列表中选择**<新建...>**。将打开**添加一个新的语言环境**窗口。
3. 在该字段中键入语言环境的名称并单击**添加**按钮。将打开“添加一个语言环境”窗口。

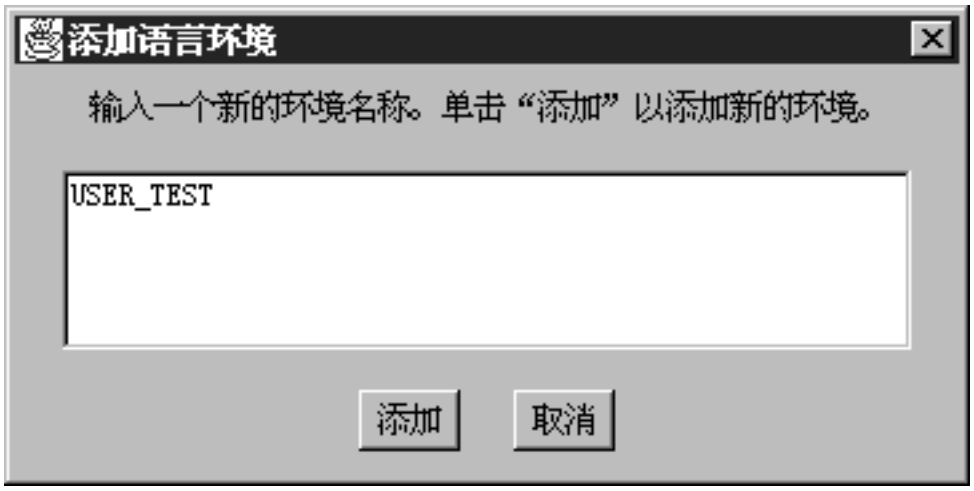


图 12. Net.Data 管理工具的“添加一个语言环境”窗口. 使用此页面来指定一个新的语言环境。

新的语言环境将被创建，并且这个名称将被添加到语言环境列表的底部。另外，新的名称是突出显示的，语言环境的缺省特性将显示在**特性**组框中。您可以更改这些值来与您的配置相匹配。

4. 关闭管理工具，或单击另一个标签来完成另外的配置任务。

要修改一个语言环境:

1. 启动管理工具。
2. 在**语言环境**页面中，从**语言环境**列表中选择您想要更改的语言环境的名称。cliette 的特性显示在**特性**组框中。
3. 根据需要在**特性**组框中修改特性，如第36页的图 12中所示:
   a. 在**名称**字段中指定语言环境的名称；这个名称对应于定义 cliette 时所使用的语言环境类型。要更改此值，可在**语言环境**列表中双击另一个名称。请参阅第16页的『环境配置语句』以获取有关语言环境类型的更多信息。
   b. 在**共享程序库或 dll 名称**字段中指定共享程序库或 DLL 程序的名称以及语言环境的路径。
   c. 选择 **DB2 信息**按钮来显示“DB2 信息”窗口，如第37页的图 13中所示。

图 13. Net.Data 管理工具的 "DB2 信息" 窗口. 使用此页面来指定专用于 DB2 数据库的信息。

指定 DB2 环境变量的值:

1) 在**连接文件**字段中键入连接文件的路径的文件名。
2) 在 **DB2 实例**字段中为使用 SQL 语言环境时所关联的数据库指定 DB2INSTANCE 值。
3) 在 **DB2 路径**字段中指定 DB2 产品可执行文件的路径目录名，通常是 \SQLLIB。
4) 单击**确定**保存更改并关闭窗口。

d. 在**参数**组框中指定输入和输出参数，它们在每次调用语言环境时传递给语言环境或从语言环境传回。

技巧: 除非您正在定义自己的语言环境，否则不要更新这些字段。

e. 在**现场连接 cliette** 组框中指定是否使用 cliette 以及哪些 cliette 应和语言环境关联。

1) 通过选择**使用现场连接 cliette** 校验框来指定用于语言环境的 cliette 是否活动。如果希望在调用语言环境时使用 **Cliette** 字段中指定的 cliette，则选中这个校验框。
2) 在 **Cliette** 字段中指定和正在定义的语言环境一起运行的 cliette 的名称。该名称的语法取决于您是在配置数据库还是 Java 应用程序语言环境。缺省为 `DTW_SQL:$(DATABASE)`。

用于数据库的语法:

type`:`*name*

其中:

type cliette 的语言环境类型。它可以是以下的一个值:

对于 **Windows NT**:

DTW_ODBC、DTW_ORA、DTW_SYB、DTW_SQL、DTW_JAVAPPS

对于 **OS/2**:

DTW_SQL、DTW_JAVAPPS

对于 **AIX**:

DTW_ODBC、DTW_ORA、DTW_SYB、DTW_SQL、DTW_JAVAPPS

name 在 **Cliette** 页面中定义的 cliette 名称。缺省为 $(DATABASE)。

Java 应用程序的语法:

```
DTW_JAVAPPS
```

4. 选择**文件**，然后选择**保存**来保存您所作的更改
5. 关闭管理工具，或单击另一个标签来完成另外的配置任务。

要删除一个语言环境:

限制: 您只能删除用户创建的语言环境，但不能删除 Net.Data 原来所附带的缺省语言环境。

1. 启动管理工具。
2. 在**语言环境**页面中，从**语言环境**列表中选择您想要删除的语言环境的名称。
3. 单击**删除**按钮。
4. 关闭管理工具，或单击另一个标签来完成另外的配置任务。

定义配置变量

使用**变量**页面来指定 Net.Data 的主目录并选择错误信息记录的等级。第39页的图 14显示了**变量**页面。

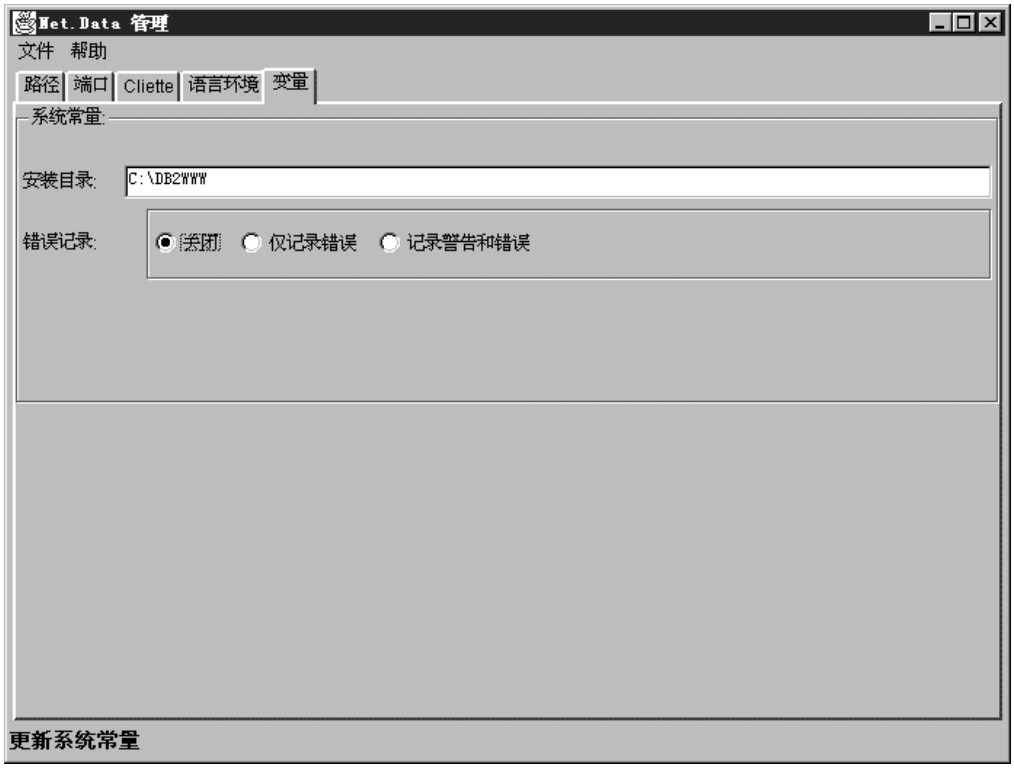


图 14. Net.Data 管理工具的变量页面. 使用这个页面来指定初始化变量。

要指定 Net.Data 的主目录:

这个变量也就是我们所知的安装目录变量。

1. 启动管理工具。
2. 在**变量**页面中，在**安装目录**字段键入要存储记录文件的目录路径。缺省为 *inst_dir*\logs\。例如: e:\db2www。
3. 关闭管理工具，或单击另一个标签来完成另外的配置任务。

要指定 Net.Data 的错误信息记录等级:

1. 启动管理工具。
2. 在**变量**页面中，从**错误记录**组框中选择一个错误记录的等级。
   - **关闭**
   - **仅记录错误**
   - **记录警告和错误**
3. 关闭管理工具，或单击另一个标签来完成另外的配置任务。

指定对 Net.Data 文件的访问权

使用 Net.Data 之前，需要确保执行 Net.Data 的用户 ID 对 Net.Data 所使用的文件具有适当的访问权。这就意味着这些文件必须位于 Web 服务器可以连接的库或目录中，或者这些用户 ID 对于它们的目录具有显式访问权。

尤为特别的，请确保执行 Net.Data 的用户 ID 具有以下权限:

- 要读取 Net.Data 初始化文件 db2www.ini
- 要执行 Net.Data 可执行文件和 DLL，并要在可执行文件和 DLL 的路径中搜索目录
- 要读取适当的 Net.Data 宏文件并搜索 MACRO_PATH 路径配置语句中标识的适当的目录
- 要执行适当的文件并搜索 EXEC_PATH 路径配置语句中标识的适当的目录
- 要读取适当的文件并搜索 INCLUDE_PATH 路径配置语句中标识的适当的目录
- 要读取和写入适当的文件并搜索 FFI_PATH 路径配置语句中标识的适当的目录
- 要读取现场连接配置文件 DTWCN.CNF
- 要读取高速缓存管理器配置文件 CACHEMGR.CNF
- 要读取语言环境引用的外部 Perl 和 REXX 可执行文件

对这些文件授予访问权的方法取决于运行 Net.Data 的操作系统。

第3章 保障您资产的安全性

您必须确定对于您的资产哪一级安全性是适当的。本章将描述可用于保障您资产安全性的方法，并提供对用于计划 Web 站点安全性的附加资源的参考。

以下章节包含了保护资产的准则。描述的安全性机制包括:

- 第41页的『使用防火墙』
- 第43页的『在网络上加密数据』
- 第43页的『使用权限审批』
- 第43页的『使用权限』
- 第44页的『使用 Net.Data 机制』

另外，Net.Data 提供了数据库 cliette 口令加密；请参阅第31页的『配置 Cliette』以获取更多信息。

使用防火墙

*防火墙*是一些硬件、软件和策略的集合，是为在网络环境内限制对资源的访问而设计的。

防火墙

- 保护内部网络不受侵入或窃密
- 保护内部网络不受内部用户带入的数据和程序的侵害
- 限制内部用户对外部数据的访问
- 在防火墙遭到破坏时限制可能造成的损坏

Net.Data 可以和在您的环境中执行的防火墙产品一起使用。

以下可能的配置为管理 Net.Data 应用程序的安全性提供了建议。这些配置提供了一些高级信息，并假定您已经配置了一个将您的安全 intranet 与公用 Internet 隔开的防火墙。请仔细考虑这些配置以及您所在组织的安全策略:

- **高安全性配置**:

  此配置创建了一个将 Net.Data 和 Web 服务器与安全 intranet 以及公用 Internet 隔开的子网。防火墙软件用于在 Web 服务器和公用 Internet 之间创建一个防火墙，并在 Web 服务器和安全 intranet (其中包含 DB2 服务器)之间创建另一个防火墙。这一配置由第42页的图 15显示。

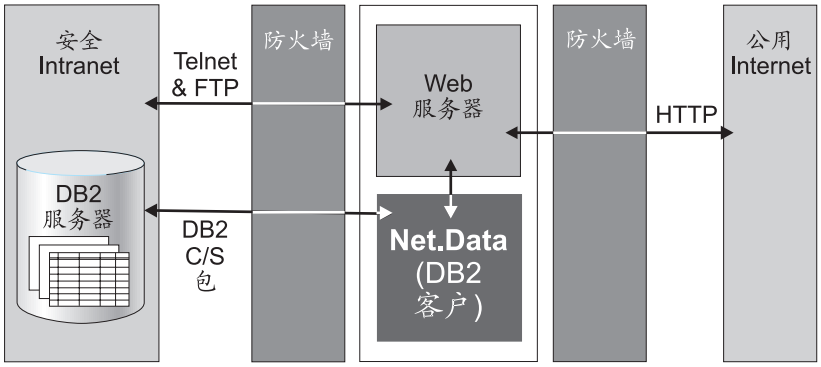


图 15. 高安全性配置

要设置此配置:

– 在 Web 服务器设备上安装 Net.Data，并确保 Net.Data 可以通过以下途径在 intranet 内部访问 DB2 服务器:

  - 在 Web 服务器设备上安装 Client Application Enabler (CAE)。
  - 配置防火墙，以允许 DB2 通信量通过防火墙。一个方法是添加信息包过滤规则，从而允许来自 Net.Data 的DB2 客户请求和从 DB2 服务器到 Net.Data 的应答信息包。

– 允许在 Web 服务器和安全 intranet 之间的 FTP 和 Telnet 访问。一个方法是在 Web 服务器设备上安装一个 socks 服务器。

– 在防火墙软件的信息包过滤配置文件中，指定从标准 HTTP 端口进入的 TCP 信息包可以访问 Web 服务器。同样，指定出去的 TCP 应答信息包可以从 Web 服务器进入公用 Internet 上的任何主机。

- **中等安全性配置**:

  在此配置中，防火墙软件将安全 intranet 和 DB2 服务器与公用 Internet 隔开。Net.Data 和 Web 服务器位于防火墙外部的工作站平台上。这种配置要比第一种简单一些，但仍提供了数据库保护机制。第42页的图 16显示了这一配置。

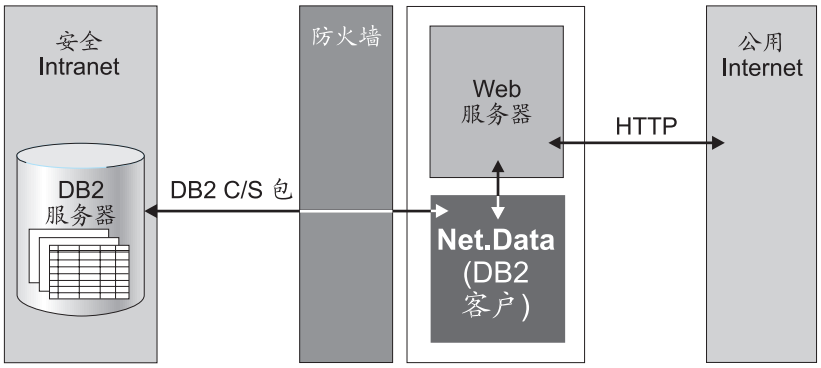


图 16. 中等安全性配置:

您必须在 Web 服务器上安装 CAE 以使 Net.Data 能够与 DB2 服务器通信。防火墙的配置必须能让 DB2 客户请求从 Net.Data 流到 DB2 ，并能让应答信息包从 DB2 流到 Net.Data。

- **低安全性配置**:

在此配置中，DB2 服务器和 Net.Data 安装在防火墙和安全 intranet 的外部。在遇到外部攻击时，它们是不受保护的。对于这种配置，防火墙不需要任何信息包过滤规则。第43页的图 17显示了这一配置。

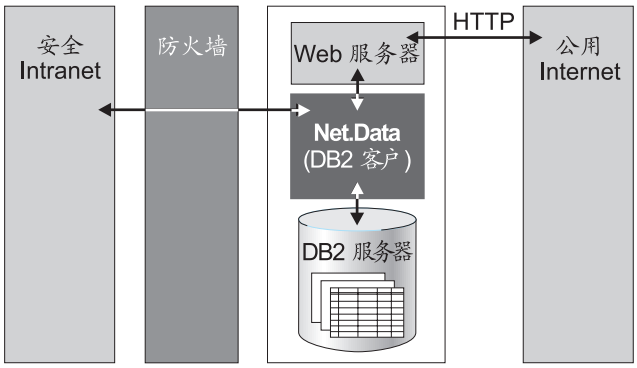


图 17. 低安全性配置:

在网络上加密数据

使用支持安全套接字层 (SSL) 的 Web 服务器时，您可以加密所有在客户系统和 Web 服务器之间发送的数据。这一安全度量支持对注册 ID、口令、通过 HTML 表从客户系统传输到 Web 服务器的所有数据以及从 Web 服务器发送到客户系统的所有数据进行加密。大部分 Web 服务器支持 SSL，例如 Internet Connection Secure Server，版本 2 发行版本 2 或更高版本以及 Lotus Domino Go Webserver，4.6.1 或更高版本。

使用权限审批

Web 服务器将用户 ID 和它所处理的每个 Net.Data 请求关联起来。然后，处理请求的进程或线程就可以访问该用户 ID 已授权的资源。

Net.Data 支持两种类型的权限审批: 一种是保护您服务器上的某些目录，一种是保护您的数据库。

- 大部分 Web 服务器允许您指定服务器上要保护的目录。您还可以使系统要求那些访问您指定目录的人给出用户 ID 和口令。请参阅 Web 服务器的*管理员指南*，以确定系统的能力。
- DB2 对于能够限制某些用户访问表和列的数据库访问有一个权限审批系统。您可以使用 Net.Data 的特殊变量(例如 LOGIN 和 PASSWORD) 来链接到 DB2 权限审批例行程序。

使用权限

诸如 DB2 等数据源提供自己的权限机制来保护它们所管理的信息。这些机制假定与正在执行 Net.Data 请求的进程相关联的用户 ID 已经正确授权，如第43页的『使用权限审批』中所说明的那样。然后，根据已授权用户 ID 所具有的权限，现有的访问控制机制将对这些数据源作出允许或拒绝访问的决定。

使用 Net.Data 机制

除了上述方法以外，您还可以使用 Net.Data 提供的其它机制，例如路径语句和隐藏变量，以及使用 HTML 表或 SQL 语句的方法。

Net.Data 评估路径配置语句的设置来确定 Net.Data 宏文件所使用的文件和可执行程序的位置。这些路径语句标识了 Net.Data 在试图定位宏文件、可执行文件和包含文件时搜索的一个或多个目录。通过在这些路径语句中有选择地包括目录，您可以显式地控制用户可以从浏览器访问的文件。请参阅第3页的『第2章 配置 Net.Data』以获取有关路径语句的附加细节。

您可以使用隐藏变量，对用他们的 Web 浏览器察看您的 HTML 源码的用户隐藏 Net.Data 宏的各种特性。例如，您可以隐藏数据库的内部结构。请参阅第65页的『隐藏变量』以获取更多信息。

您还可以使用以下方法来设置保护方案:

- 使用 Net.Data 创建自己的保护方案。例如，你可以通过 HTML 表请求来自用户的验证信息并使用数据库中的数据进行验证，或者通过 Net.Data 宏所调用的外部程序来请求。
- 通过允许别人发送给数据库的 SQL 语句来保护您的资产，例如，将 SELECT 语句限制在两个表中。

有关保护资产的更多信息，请参阅以下 Web 站点中有关“经常询问的问题”(FAQ) 的 Internet 安全性列表:

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

第4章 调用 Net.Data

您可以配置 Net.Data，使它能够与公共网关接口 (CGI) 或 FastCGI 一起使用，或与 Web 服务器 API 一起使用，例如 Lotus Domino Go Webserver (GWAPI)、Internet Connection Server (ICAPI)、Netscape Server (NSAPI) 和 Microsoft Internet Server (ISAPI)。用于调用 Net.Data 的语法取决于 Net.Data 的配置。请参阅第3页的『第2章 配置 Net.Data』，以了解如何配置 Net.Data。

本章将讨论如何使用 CGI 来调用 Net.Data。请参阅第121页的『使用Web 服务器 API 改进性能』，以学习如何在 API 方式中调用 Net.Data。

您还可以指定是希望 Net.Data 执行一个宏文件还是单个 SQL 语句、存储过程或函数。这些调用类型被分别称为宏请求和直接请求。

宏请求 通过指定一个使用 Net.Data 宏语言编写的宏文件来调用 Net.Data。

直接请求

通过指定以下内容来调用 Net.Data:

- 语言环境的名称
- 一个 SQL 语句或程序的名称，并带有函数调用所需的任何参数值
- 调用 SQL 语句或函数所需的表数据

现在，想要编写单个 SQL 查询或调用单个函数(例如 DB2 存储过程、REXX 程序或 Perl 函数)的 Web 开发者可以向数据库发出直接请求了。直接请求中没有任何需要 Net.Data 宏文件的复杂的 Net.Data 应用逻辑，因此可以绕过 Net.Data 宏处理器。为了改进性能，直接请求参数被传递到适当的语言环境进行处理。

第46页的图 18说明了宏请求和直接请求之间的区别。宏请求总是在请求的 URL 中指定一个宏文件，还可以使用表数据。直接请求则不在 URL 中指定宏文件，但仍然可以使用表数据。

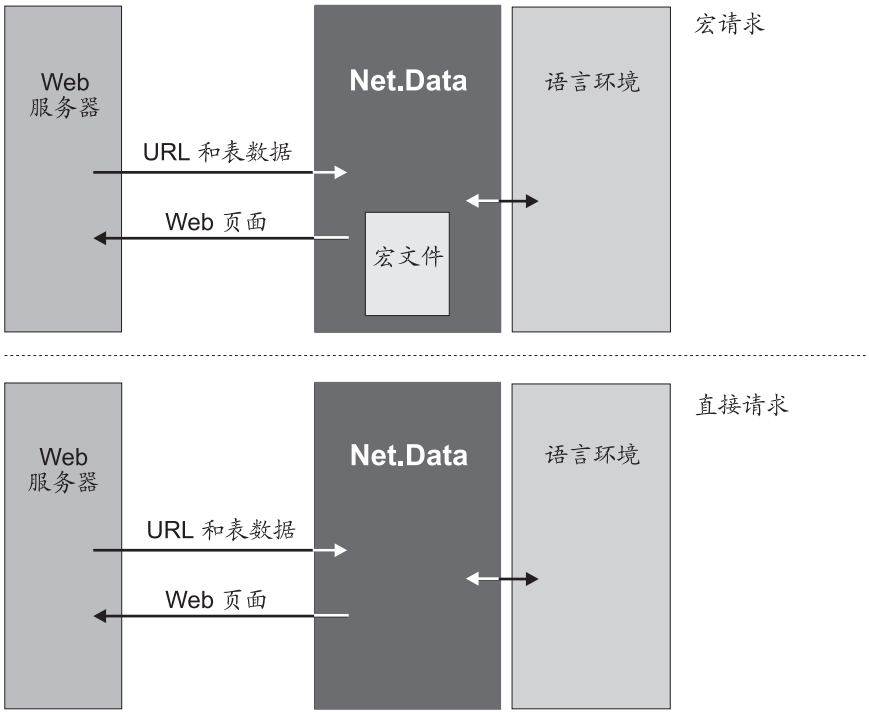


图 18. 宏请求与直接请求

调用 Net.Data 的语法取决于 Net.Data 的配置以及请求的类型。对于宏请求和直接请求这两者来说，可以从 Web 浏览器通过 HTML 链、HTML 表或 URL 来调用 Net.Data。Web 服务器使用 CGI、 FastCGI 或一个 Web 服务器 API 来调用 Net.Data。

对于宏请求来说，Net.Data 宏文件的名称和要在 Net.Data 宏内部执行的 HTML 块的名称都是在链、表或 URL 中指定的。

对于直接请求，Net.Data 语言环境的名称、SQL 语句或程序的名称、以及其它必需的参数值都是使用 Net.Data 定义的语法在 URL 内定义的。

本章将描述这两种调用方法:

- 第46页的『使用宏文件(宏请求)调用 Net.Data』
- 第48页的『不使用宏文件对 Net.Data 进行调用(直接请求)』

使用宏文件(宏请求)调用 Net.Data

本节将告诉您如何通过指定一个宏文件来调用 Net.Data。对于宏请求风格的调用，您可以使用一个 URL、HTML 表或 HTML 链来调用 Net.Data。

以下例子显示了调用 Net.Data 的不同方式。

- HTML 链:

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

- HTML 表:

```
<FORM METHOD=method ACTION="http://server/cgi-bin/db2www/
filename/block/[?name=val&...]">any text</FORM>
```

- URL:

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

参数:

server 指定了 Web 服务器的名称。如果是本地服务器，则可以忽略服务器名称而使用相关的 URL。

filename 指定 Net.Data 宏文件的名称和扩展名，其中文件名是 MACRO_PATH 指定目录下的相关路径。

block 在引用的 Net.Data 宏文件中指定 HTML 块的名称。

method 指定与表一起使用的 HTML 方法。建议采用 METHOD=POST。

?name=val&...

指定一个或多个传递给 Net.Data 的可选参数。

HTML 链

您可以通过在宏文件中使用 HTML <a> 标记来创建 Web 页面中的链，而这个链将会执行一个 HTML 块。您可以使用 HREF 属性来指定宏和 HTML 块，并在链标记中包括一些文本甚或图象。当 Web 页面显示在浏览器上时，这个方法可以将文本或图象标识为『热点』。当浏览器前的用户单击该文本或图象时，Net.Data 将执行宏中的 HTML 块。

以下例子显示了这样一个链: 当用户在 Web 页面上选择文本 ″List all monitors″ 时，这个链将使得一个 SQL 查询开始执行。

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

该链调用了这个宏:

```
%DEFINE DATABASE="MNS97"

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

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

查询将返回一个包含型号、成本和 EQPTABLE 表格中对每个监视器所描述信息的表格。此例通过生成一个缺省报表显示了查询的结果。请参阅第81页的『报表块』以获取有关如何使用 REPORT 块定制报表的信息。

通常，Net.Data 宏的每个块都是以 %block_name{ 开头，以 %} 结束的。请参阅 *Net.Data 参考*以获取有关 Net.Data 宏语言语法的附加细节。

HTML 表

您可以使用 HTML 表来动态地定制 Net.Data 宏的执行。这些表允许用户提供输入值，而这些值将影响宏的执行和 Net.Data 构建的 Web 页面的内容。

以下例子构建在第47页的『HTML 链』中监视器列表的例子上，它使得浏览器前的用户可以使用一个简单的 HTML 表来选择要求显示信息的产品类型。

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

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

当浏览器前的用户作出了他们的选择并单击“提交”按钮之后，Web 服务器将处理调用 Net.Data 的 FORM 标记的 ACTION 参数。然后，Net.Data 将执行 equiplst.d2w 宏中的 Report 块:

```
%DEFINE DATABASE="MNS97"

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

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

在上述例子中，SQL 语句中 TYPE=$(hdware) 的值是从 HTML 表的输入中获得的。

请参阅 *Net.Data 参考*以获取对于 ROW 块中所使用变量的详细描述。

不使用宏文件对 Net.Data 进行调用(直接请求)

本节将告诉您如何在不指定宏文件的情况下调用 Net.Data。对于直接请求风格的调用，您可以使用 Net.Data 定义的语法在 URL 中指定 Net.Data 语言环境的名称、要执行的 SQL 语句或程序以及其它必需的参数值。

SQL 语句或程序以及其它指定的参数都被直接传递到指定的语言环境进行处理。直接请求可以改进性能，因为 Net.Data 不需要读取和处理宏文件。SQL、ODBC、Oracle、Sybase、Java、System、Perl 和 REXX 这些 Net.Data 提供的语言环境支持直接请求调用，您可以使用 URL、HTML 表或链来调用 Net.Data。

直接请求通过传递 URL 或表数据的查询字符串中的 (NAME=VALUE) 对中的参数来调用 Net.Data。以下例子说明了可以指定直接请求的上下文。

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

其中 *direct_request* 代表直接请求的语法。例如，以下 HTML 链中包含直接请求:

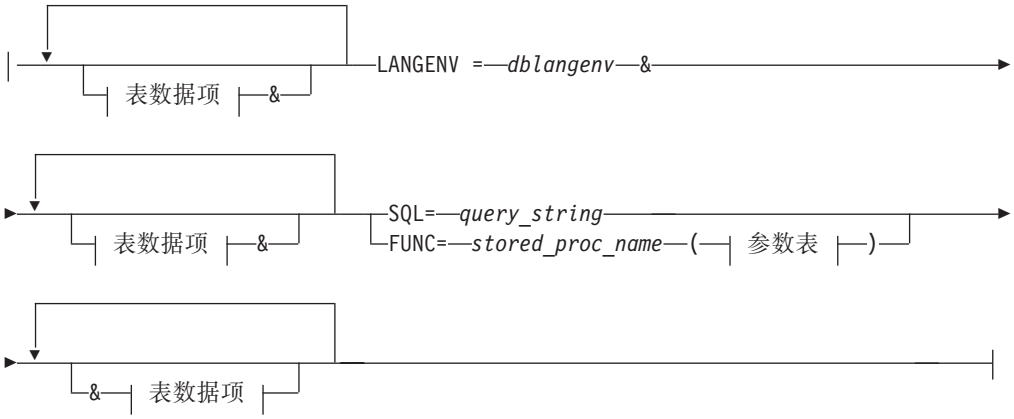
```
<A HREF="http://server/cgi-bin/db2www?LANGENV=DTW_PERL&FUNC=my_perl(hi)">
  any text</A>
```

直接请求的语法

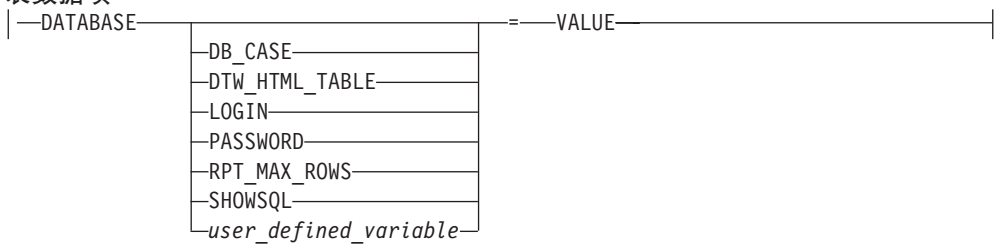
使用直接请求调用 Net.Data 的语法中可以包含一个对数据库语言环境或非数据库语言环境的调用。

语法

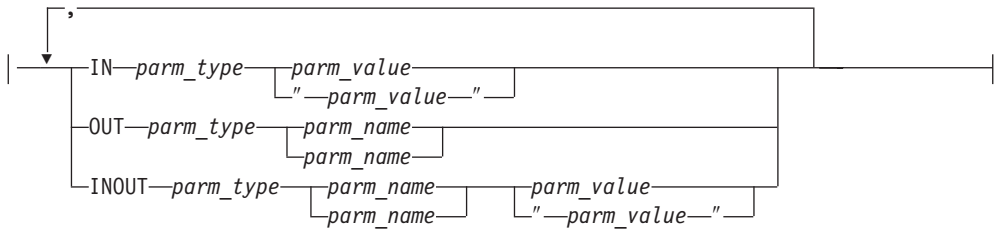
数据库语言环境调用



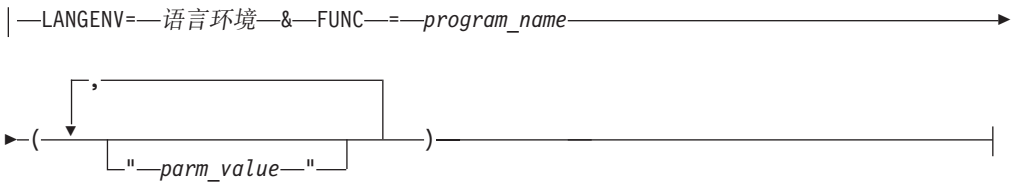
表数据项



参数表



非数据库语言环境调用



参数

数据库语言环境调用

向 Net.Data 指定一个调用数据库语言环境的直接请求。

表数据项

允许您指定 SQL 变量设置或请求简单的 HTML 格式的参数。请参阅 *Net.Data 参考*以了解这些变量的更多信息。

DATABASE
指定 Net.Data 应将 SQL 请求传递到哪个数据库。此参数是必需的。

DB_CASE
指定 SQL 语句的大小写情况(大写或小写)。

DTW_HTML_TABLE
指定 Net.Data 是否应返回一个 HTML 表格。

LOGIN
指定数据库用户 ID。

PASSWORD
指定数据库口令。

RPT_MAX_ROWS
指定函数在报表中返回的表格的最大行数。

SHOWSQL
指定 Net.Data 是应当隐藏还是显示要执行的 SQL 语句。

user_defined_variable
传递给 Net.Data 的变量，并提供必需的信息或实现 Net.Data 的行为。用户定义的变量是您为应用程序定义的。

VALUE
指定 Net.Data 变量的值。

LANGENV
为 SQL 语句或存储过程调用指定目标语言环境。如果该语言环境是数据库语言环境，则必须指定数据库名称。

dblangenv
数据库语言环境的名称:

- DTW_SQL
- DTW_ODBC
- DTW_ORA
- DTW_SYB

SQL
表示直接请求指定了在线 SQL 语句的执行。

query string
指定一个字符串，其中包含任何可以使用动态 SQL 来执行的有效的 SQL 语句。

FUNC
表示直接请求指定了一个存储过程的执行。

stored_proc_name
指定任何有效的 DB2 存储过程名。

parm_type
为 DB2 存储过程指定任何有效的参数类型。

parm_name
指定任何有效的参数名。

parm_value
为 DB2 存储过程指定任何有效的参数值。

IN 指定 Net.Data 应当使用该参数将输入数据传递到存储过程。

INOUT
指定 Net.Data 应当使用该参数将输入数据传递到存储过程并返回来自语言环境的输出数据。

OUT
指定语言环境应当使用该参数返回来自存储过程的输出数据。

非数据库语言环境调用
向 Net.Data 指定一个调用非数据库语言环境的直接请求。

LANGENV
为函数的执行指定目标语言环境。

lang_env
指定非数据库语言环境的名称:

- DTW_PERL
- DTW_REXX
- DTW_SYSTEM

FUNC
表示直接请求指定了一个程序的执行。

program_name
指定程序，其中包含要执行的函数。

parm_value
为函数指定任何有效的参数值。

直接请求示例

以下例子显示了可以在使用直接请求方法时调用 Net.Data 的不同方式。

HTML 链

例 1: 一个调用 Perl 语言环境并调用 Net.Data 初始化文件中 EXEC 路径语句内的 Perl 脚本的链。

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

例 2: 一个调用 Perl 语言环境的链，同前例，但它传递的字符串中具有双引号和空格字符的 URL 编码值。

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

技巧: 您必须在 URL 内对某些字符编码，例如空格和双引号。在此例中，参数值中的双引号字符和空格必须分别编码成 %22 和 + 字符。请参阅 *Net.Data 参考*中对 DTW_URLESCSEQ 函数的描述，以获取所有在 URL 内必须编码的字符列表。

HTML 表

例 1: 一个将执行 SQL 查询(使用 SQL 语言环境)的 HTML 表，它还连接到 CELDIAL 数据库并查询一个表格

```
<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>
```

技巧: 通过使用一个 Net.Data 宏所创建的 HTML 表，您可以在直接请求调用中使用变量替换。

URL

例 1: 一个将执行 SQL 查询(使用 SQL 语言环境)的 URL

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

例 2: 一个调用 Perl 语言环境并调用不在 Net.Data 初始化文件的 EXEC 路径语句内的可执行文件的 URL。

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

技巧: 要使用不在 EXEC 路径配置语句中指定的路径来引用一个文件名，需要提供全限定的路径和文件名作为 *function_name* 的值。

例 3: 一个调用 System 语言环境并调用外部 Perl 脚本的 URL

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

例 4: 一个调用 REXX 语言环境、调用 REXX 程序并将参数传递到程序的 URL

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

例 5: 一个调用存储过程并将参数传递到 SQL 语言环境的 URL

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

第5章 开发 Net.Data 宏

Net.Data 宏是一个文本文件，它由一系列 Net.Data 宏语言结构组成:

- 指定 Web 页的版面设置
- 定义变量和函数
- 调用 Net.Data 的内部函数或宏文件中定义的函数
- 格式化 HTML 中的处理输出，并把它返回给 Web 浏览器

正如第55页的图 19所示，Net.Data 宏包括两部分: 声明部分和 HTML 部分。

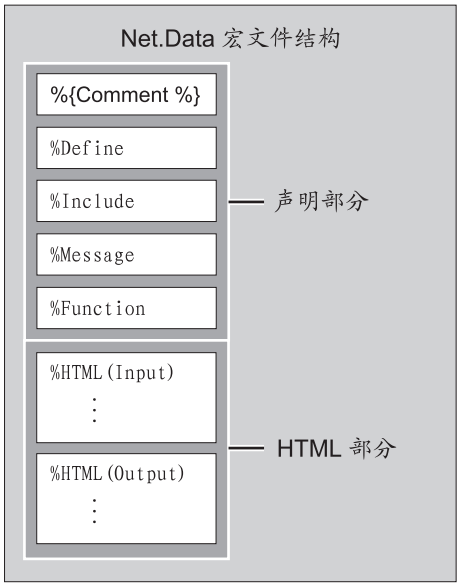


图 19. 宏文件结构

- *声明部分*包含宏文件中变量和函数的定义。
- *HTML 部分*包含 HTML 块，其中含指定 Web 页的版面设置的 HTML 语句。

您可以以任何顺序多次使用这些部分。 请参阅*Net.Data 参考*以获取关于宏文件部分的语法和结构。

权限提示: 确保此 Web 服务器具有对该文件的访问权。 请参阅第39页的『指定对 Net.Data 文件的访问权』，以获取更多信息。

本章检查组成 Net.Data 宏文件的不同块以及用于写宏文件的方法。

- 第56页的『Net.Data 宏文件的剖析』
- 第60页的『Net.Data 宏变量』
- 第69页的『Net.Data 函数』
- 第80页的『在宏中生成 HTML』
- 第83页的『宏文件中的条件逻辑和循环』
- 第86页的『使用大型对象』

Net.Data 宏文件的剖析

宏文件由两部分组成:

- 声明部分，包含了要在 HTML 部分中使用的定义。声明部分使用两个主要的可选块:
  - DEFINE 块
  - FUNCTION 块

  声明部分还可以包含其它语言结构和语句，例如 EXEC 语句、IF 块、INCLUDE 语句和 MESSAGE 块。关于语言结构的更多信息，请参阅*Net.Data 参考*中关于语言结构的章节。

  权限提示: 确保此 Web 服务器具有对 EXEC 和 INCLUDE 语句引用的文件的访问权。 请参阅第39页的『指定对 Net.Data 文件的访问权』，以获取更多信息。

- HTML 部分定义 Web 页面的布局、引用变量，并使用 HTML 块作为宏的入口和出口点来调用函数。在调用 Net.Data 时，指定一个 HTML 块名作为处理宏文件的入口点。 第58页的『HTML 块』中描述了 HTML 块。

在本小节中，用一个 Net.Data 宏说明了宏语言的元素。此例子宏表达了一种格式，该格式提示要向 REXX 程序传送的信息。宏将此信息传送至称为 OMPSAMP.CMD 的外部 REXX 程序，该程序回送用户输入的数据。然后在第二个 HTML 页面上显示结果。

首先看整个宏，然后详细看每块:

```
%{ **********************         DEFINE 块         ************************%}
%DEFINE{
   page_title="Net.Data macro Template"
%}

%{ **********************      FUNCTION 定义块      ************************%}
%FUNCTION(DTW_REXX) rexx1 (IN input) returns(result)
    { %EXEC{ompsamp.cmd %}
%}

%FUNCTION(DTW_REXX) today () RETURNS(result)
  {
      result = date()
%}

%{ **********************    HTML 块: 输入      ************************%}
%HTML(INPUT) {
  <html>
 <head>
<title>$(page_title)</title>
</head><body>
<h1>Input Form</h1>
Today is @today()

<FORM  METHOD="post" ACTION="output">
输入一些数据传送给一个 REXX 程序:
<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 块: 输出      ************************%}
```

```
%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>
%}
```

示例宏包含四个主要块: DEFINE、FUNCTION 和两个 HTML 块。在一个 Net.Data 宏中可以有多个 DEFINE、FUNCTION 和 HTML 块。

两个 HTML 块包含了人们熟悉的 HTML 标记，这使 Web 宏更容易书写。如果您熟悉 HTML，就知道构建一个宏只涉及添加要在服务器上动态处理的宏语句和要发送到数据库的 SQL 语句。

虽然宏看起来类似于 HTML 文档，但是 Web 服务器是通过 Web 服务器 API 或使用 CGI 的 Net.Data 来访问它的。Net.Data 需要两个参数: 要处理的宏的名称和该宏中要显示的 HTML 块。

调用了宏文件之后，Net.Data 从头开始处理它。以下各章节着眼于当 Net.Data 处理文件时所发生的事情。

DEFINE 块

DEFINE 块包含了 DEFINE 语言结构以及以后要在 HTML 块中使用的变量定义。下例显示具有一个变量定义的 DEFINE 块:

```
%{ **********************        DEFINE 块          ************************%}
%DEFINE{
   page_title="Net.Data macro Template"
%}
```

第一行是一个注解。注解是 %{ 和 %}中的任何文本。注解可处于宏文件中的任何地方。下一个语句起始于一个 DEFINE 块。可以在一个定义块中定义多个变量。在此例中，只定义了一个变量 page_title。定义了该变量之后，就可以使用语法 $(page_title) 在宏中的任何地方引用它。使用变量便于以后对宏作全局变更。该块的最后一行 %} 标识了 DEFINE 块的结束。

FUNCTION 块

FUNCTION 块包含了对 HTML 块所调用的函数的说明。函数由语言环境处理，可以执行程序、SQL 查询或存储过程。

下例显示两个 FUNCTION 块，它们分别定义一个对外部 REXX 程序的函数调用和一个对宏文件中所包含的函数的函数调用。

```
%{ **********************   FUNCTION 块  ***********************************%}
%FUNCTION(DTW_REXX) rexx1 (IN input) returns(result) { <-- 该函数接受
                                                         一个参数并返回一个
                                                         结果，该结果被关联的
```

```
                                                        函数调用所代替
     %EXEC{ompsamp.cmd %}  <-- 函数执行一个称为 "ompsamp.cmd" 的外部 REXX 程序
%}

%FUNCTION(DTW_REXX) today () RETURNS(result) {
      result = date()  <-- 该函数的单独的源语句包含在内部。
%}
```

第一函数块 rexx1 是一个 REXX 函数说明，它依次运行一个称为 ompsamp.cmd 的外部 REXX 程序。输入变量 input 被该函数接受并自动传送至外部 REXX 命令。REXX 命令还返回变量 result。REXX 命令中的变量 result 的内容代替 Output 块中包含的调用的 @rexx1() 函数调用。REXX 程序直接可访问 input 和 result 变量，正如 ompsamp.cmd 源码中所显示的那样:

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

该函数中的代码回送传送给它的数据。您可以通过用一般 HTML 风格的标记(象 <b> 或)括起请求的 @rexx1() 函数来按您的需要格式化结果文本。与 result 变量相比，REXX 程序宁愿使用 REXX SAY 语句来将 HTML 标记写入标准输出。

第二块 today 也属于 REXX 程序。但是，在此情况下，整个 REXX 程序(整行)都包含在本身的函数说明中。不需要外部程序。REXX 和 Perl 函数都允许内部程序，因为它们是解释语言，可以动态语法分析和执行。内部程序的优点是简明，不需要一个程序文件来管理。第一个 REXX 函数也可以内部处理。

HTML 块

HTML 块定义 Web 页面的布局、引用变量，并使用 HTML 块作为宏的入口和出口点来调用函数。 HTML 总是在 Net.Data 调用请求中指定的，并且每个宏必须至少有一个 HTML 块。

例子宏文件中的第一个 HTML 块称为 INPUT。INPUT 块包含具有一个输入字段的简单表格。

```
%{ **********************    HTML 块: 输入      ************************%}
%HTML (INPUT) {               <--- 标识该 HTML 块的名称。
  <html>
 <head>
<title>$(page_title)</title>  <--- 加注变量替换。
</head><body>
<h1>Input Form</h1>
Today is @today()             <--- 该行包含对一个函数的调用。

<FORM METHOD="post" ACTION="output">  <--- 该表格提交时,
                                           将调用 "output" HTML 块。
输入一些数据传送给一个 REXX 程序:
<INPUT NAME="input_data"      <--- 在表格被提交时 "input_data" 被定义,
TYPE="text" SIZE="30">             并且可以在该宏中的任何地方引用它。
                                   它已被初始化为输入字段中的某种用户类型。
<p>
<INPUT TYPE="submit" VALUE="Enter">

< hr>
<p>
[
<a href="/">Home page</a>]
 </body><html>
%}                            <--- 关闭 HTML 块。
```

整个块由 HTML 块标识符 %HTML (INPUT) {...%}括起。INPUT 标识了该块的名称。您可为它取任何名称。HTML <title>标记包含了变量替换的例子。变量 page_title 的值替换了表格的标题。

此块还具有一个函数调用。表达式 @today() 是对函数 today 的调用。该函数是在随后描述的 FUNCTION 块中定义的。Net.Data 将 today 函数的结果(即当前日期)插入 HTML 文本中与 @today() 表达式相同的位置。

FORM 语句的 ACTION 参数提供了一个在 HTML 块之间或在宏之间游历的例子。表格提交时，ACTION 参数中对另一块名称的引用将访问此块。HTML 表中的任何输入数据都作为隐式变量传送给块。该表上定义的单个输入字段，就是这样的。当表格提交时，该字段中输入的数据在变量 *input_data* 中传送给 HTML 输出块。

如果另一个宏文件在相同的 Web 服务器上，您就可以用交叉引用来访问该宏文件中的 HTML 块。例如，ACTION 参数 ACTION="../othermacro.d2w/main" 访问宏文件 othermacro.d2w 中 HTML 块调用的主程序。表格中的任何数据类型再次在 *input_data* 中传送给该宏。

在调用 Net.Data 时，您将变量作为 URL 的一部分传送。例如:

```
<a href="/cgi-bin/db2www/othermacro.d2w/main?input_data=value">Next macro</a>
```

不需要定义环境变量来接收输入数据，因为您具有大多数 CGI 程序。Net.Data 将为您处理环境变量。您只需要引用变量名。

例子中的下一个 HTML 块是 OUTPUT 块。它包含 HTML 标记和 Net.Data 宏语句，这些宏语句定义 INPUT 块请求所处理的输出。

```
%{ **********************    HTML 块: 输出       ************************%}
%HTML(OUTPUT) {
  <html>
 <head>
<title>$(page_title)</title>  <--- 更多替换。

</head><body>
<h1>Output Page</h1>
<p>@rexx1(input_data)  <--- 该行包含对函数 rexx1 的调用，并传送参数 "input_data"。
<p>
< hr>
<p>
[
<a href="/">Home page</a> |
<a href="input">Previous page</a>]
%}
```

与 INPUT 块相似，此块也是 标准 HTML，其中使用 Net.Data 宏语句来替换变量和函数调用。再次用 page_title 变量来替换标题语句。与上面相似，此块也包含一个函数调用。在此情况下，它调用函数 rexx1 并将变量 input_data 的内容传送给它，该变量接收自 Input 块中定义的表格。您可以将任何数目的变量传送给函数，或从函数中传送回任何数目的变量。函数定义确定了传送的变量的数目和类型。

Net.Data 宏变量

Net.Data 允许您在 Net.Data 宏中定义和引用变量。另外，也可以将这些变量从宏传送给语言环境，反之亦然。

可以根据变量类型和是否具有预定义值来定义 Net.Data 变量。可以根据变量的定义将这些变量分为以下类型:

- 在 DEFINE 块中用 DEFINE 语句显式定义的变量
- 预定义变量，这些变量由 Net.Data 提供并设为一个值。此值通常无法更改。
- 隐式定义的变量，有四类:
  - 未显式定义，但是在首次引用时例示的那些变量
  - 参数变量，它们是 FUNCTION 块定义的一部分，只可以在 FUNCTION 中引用。
  - 由 Net.Data 例示并对应于表格数据或 URL 数据名称-值偶对的那些变量。
  - 与一个 Net.Data 表相关联并只可以在 ROW 或 REPORT 块中引用的那些变量。

一个标识符(它是一个变量或函数调用)成为*可见*，意味着当它被说明或例示之后就可被引用。标识符可见的区域称为它的*作用域*。作用域有 5 种类型:

- 全局

  如果您可以在一个宏文件中的任何地方引用一个标识符，则该标识符就具有全局作用域。具有全局作用域的标识符有:

  - Net.Data 内部函数
  - 表格数据
  - URL 数据
  - 在一个 HTML 块中例示的变量
- 宏文件

  如果一个标识符的说明出现在任何块的外面，则它有此作用域。一个块以左括号 ({) 开始，以百分号加右括号 (%}) 结束。(注意，DEFINE 块是此定义的例外，应当将它作为独立的 DEFINE 语句处理)。具有宏文件作用域的标识符从说明点到宏文件结束都是可见的。
- FUNCTION 块或 MACRO_FUNCTION 块

  如果在函数定义的参数表中说明一个标识符，则该标识符具有函数块作用域。如果一个标识符在函数定义的外面已经存在相同名称，Net.Data 将使用函数块中的参数表中的标识符。Net.Data 不使用或修改函数块外面的标识符及其值。

  如果一个标识符在函数外已被说明或初始化并且没有在函数参数表中说明，则该标识符不具有函数块作用域。在函数块中使用此标识符时，它保持函数调用前指定的值。在函数块中更新之后，此标识符在函数调用之后保持新值。

  如果在参数表中没有定义并且在函数调用之前没有说明或初始化一个标识符，则该标识符具有函数块作用域。 不能在函数块之外引用它。
- REPORT 块

  如果一个标识符只可以在 REPORT 块中被引用(例如表列名 N1、N2、 ...、N*n*)，则它具有报表块作用域。只有 Net.Data 隐式定义为表处理的一部分的那些变量才可以具有报表块作用域。例示的任何其它变量都具有函数块作用域。
- ROW 块

如果只可以从 ROW 块中调用一个标识符(例如表值名 V1、V2、...、Vn)，则该标识符具有行块作用域。只有 Net.Data 隐式定义为表处理的一部分的那些变量才可以具有行块作用域。 例示的任何其它变量都具有函数块作用域。

引用一个标识符时，它被替换以标识符的值。如果对变量的引用没有值与它关联，或函数调用没有返回值，则此引用就被替换以空字符串。

以下各节描述如何定义和引用变量，并描述不同的变量类型以及使用方法。

定义变量

Net.Data 宏中有三种定义变量的方式:

- **DEFINE 语句或块**

  定义一个变量以在 Net.Data 宏中使用的最简单方式是使用 DEFINE 语句。此语法是 Net.Data 特有的:

  ```
  %DEFINE variable_name="variable value"

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

  variable_name 是给予变量的名称。变量名必须以一个字母或下划线开头，并包含任何字母数字字符或下划线。 所有变量名都是区别大小写的，但是 N_*columnName* 和 V_*columnName* 除外，它们是表变量。

  要在字符串中包含引号，必须使用两个连续的引号。单独的两个连续引号等于一个空串。例如:

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

  变量 HI 显示 say "hello"。

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

  变量 reply 显示 hello。

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

  变量 empty 为空。

  要在一个 DEFINE 语句中定义几个变量，可使用 DEFINE 块:

  ```
  %DEFINE{
          variable1="value1"
          variable2="value2"
          variable3="value3"
          variable4="value4"
  %}
  ```

- **HTML 表 SELECT 和 INPUT 标记**

  您可以为 HTML 表使用 SELECT 和 INPUT 标记。下例使用标准 HTML 表标记来定义一个变量:

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

  或

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

variable_name 是给予变量的名称，而变量值是根据表格中接收的输入来确定的。请参阅第47页的『HTML 表』以获取关于如何在 Net.Data 宏中使用此类变量定义的例子。

接收自 INPUT 或 SELECT 标记的变量值覆盖由 Net.Data 宏中的 DEFINE 语句设置的变量值。

- **URL 数据**

  您可以调用 Net.Data 宏作为 URL 请求或在 URL 中包含变量(例如一个用户标识符)以发送给 Net.Data。例如:

  ```
  http://www.ibm.com/cgi-bin/db2www/stdqry1.d2w/input?field=custno
  ```

  在上例中，变量名 `field` 和变量值 `custno` 指定 Net.Data 接收自输入语句的附加数据。Net.Data 根据表格接收和处理数据。

引用变量

您可以引用先前定义变量以返回它的值。

要在 Net.Data 宏中引用一个变量，可在 `$(` 和 `)` 中指定变量名。例如:

```
$(variableName)
$(homeURL)
```

当 Net.Data 发现一个变量引用时，它用值来替换变量引用。

要将变量作为 HTML 的一部分使用，可在 HTML 块中引用它们。例如，如果您定义了变量 `homeURL`:

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

您可以指向主页为 $(homeURL) 并创建一个链接:

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

可以在 Net.Data 宏的任何部分引用变量。如果变量在被引用时尚未被定义， Net.Data 将定义此变量并为它赋一个空的初始值。如果变量在被引用时尚未被定义， Net.Data 将返回一个空字符串。Net.Data 不定义变量。

限制: 不允许循环引用。例如，当引用了变量，并且已求出了最后的值时，下面的 DEFINE 语句结果出错:

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

变量类型

在宏文件中可以使用以下变量引用类型:

- 第63页的『条件变量』
- 第63页的『环境变量』
- 第63页的『可执行变量』
- 第65页的『隐藏变量』
- 第65页的『列表变量』

- 第66页的『表格变量』
- 第67页的『杂项变量』
- 第67页的『表格处理变量』
- 第68页的『报表变量』
- 第68页的『语言环境变量』

如果您将字符串赋给变量，而变量由 Net.Data 定义为某种方式，例如 ENVVAR、LIST、条件列表变量，则变量不再表现为定义的方式。换句话说，变量成为一个包含字符串的简单变量。

条件变量

条件变量让您通过使用类似于 IF、THEN 结构的方法来为一个变量定义一个条件值。在定义条件变量时，可以指定两个可能的变量值。如果引用的第一个变量存在，条件变量将获取第一个值；否则获取第二个值。条件变量的语法是:

```
varA = varB ? "value_1" : "value_2"
```

如果 varB 已定义，则 varA="*value_1*"，否则 varA="*value_2*"。这是等价于使用 IF 块，如下例所示:

```
%IF $(varB)
    varA = "value_1"
%ELSE
    varA = "value_2"
%ENDIF
```

请参阅第65页的『列表变量』以获取使用条件变量与列表变量的例子。

环境变量

您可以引用存在于运行 Net.Datav 的进程中的 Net.Data 环境变量。

定义环境变量的语法是:

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

其中 *var* 是定义的变量名。

例如，变量 SERVER_NAME 可被定义为环境变量:

```
%define SERVER_NAME=%ENVVAR
```

然后被引用:

```
The server is $(SERVER_NAME)
```

输出是这样的:

```
The server is www.software.ibm.com
```

请参阅*Net.Data 参考*以获取关于 ENVVAR 语句的更多信息。

可执行变量

您可以用可执行变量来从变量引用中调用其它函数。

使用 DEFINE 块中的 EXEC 语言结构来定义 Net.Data 宏中的可执行变量。关于 EXEC 语言元素的更多信息，请参阅*Net.Data 参考*中的语言结构章节。 在下例中，定义了变量 runit 来执行可执行程序 testProg:

```
%DEFINE runit=%exec "testProg"
```

runit 成为可执行变量。

Net.Data 在 Net.Data 宏中遇到一个有效变量时运行可执行程序。 例如，当 Net.Data 宏中有一个有效变量引用建立成变量 runit 时，即执行 testProg 程序。

一种简单的方法是从另一个变量定义中引用一个可执行变量。以下例子演示了这个方法。 变量 date 定义成一个可执行变量，dateRpt 定义成一个变量引用，它包含这个可执行变量。

```
%DEFINE date=%exec "date"
%DEFINE dateRpt="Today is $(date)"
```

不管 $(dateRpt) 出现在 Net.Data 宏中的何处，Net.Data 都搜索可执行程序 date，并在找出时返回:

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

当 Net.Data 在宏文件中遇到可执行变量时，它将使用下列方法寻找被引用的可执行程序:

1. 它在 Net.Data 初始化文件由 EXEC_PATH 指定的目录中搜索。 请参阅第14页的『EXEC_PATH』，以获取细节。
2. 如果 Net.Data 找不到此程序，系统将搜索系统 PATH 环境变量或库表所定义的目录。 如果找到了此可执行程序，则 Net.Data 运行它。

限制: 不要将可执行变量设置成它调用的可执行程序的输出值。在先前的例子中，变量 date 的值是空的。如果在 DTW_ASSIGN 函数调用中使用此变量来把它的值分配给另一个变量，则赋值后新变量的值也是空。 可执行变量的唯一目的是去调用它定义的程序。

也可以给要执行的程序，通过在变量定义上指定此程序名，将参数传送给它。在此例中，距离和时间的值传送给程序 *calcMPH*。

```
%DEFINE mph=%exec "calcMPH $(distance) $(time)"
```

下一个例子将系统日期作为 HTML 报表的一部分返回:

```
%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()
%}
```

每个报表都显示日期以便于跟踪。此例还将用户号和名称放在另一个 Net.Data 宏的链接中。单击报表中的任何用户将调用 exmp.d2w Net.Data 宏，即将用户号和名字传送给 Net.Data 宏。

隐藏变量

您可以使用隐藏变量，对用他们的 Web 浏览器察看您的 HTML 源码的用户隐藏应用程序的实际变量名。要定义隐藏变量:

1. 在 HTML 块中变量的最后一个引用之后，为每个需要隐藏的字符串定义一个变量。如下例所示，变量在 HTML 块中使用之后，总是用 DEFINE 语言结构来定义。$$(*variable*) 变量被引用然后被定义。
2. 在引用此变量的 HTML 块中，使用两个美元符号代替一个美元符号来引用变量。 例如，$$(X) 替换 $(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
%}

.
.
.
```

Web 浏览器显示 HTML 表时，$$(name) 和 $$(addr) 分别被替换以 $(name) 和 $(addr)，所以实际的表格和列名肯定不出现在 HTML 表上。应用程序用户无法区分实际变量名是隐藏的。当用户提交这个表时，调用 HTML(REPORT) 块。当 @mySelect() 调用 FUNCTION 块时，$(Field) 在 SQL 语句中用 SQL 查询的 customer.name 或 customer.addr 替换。

列表变量

使用列表变量来构建一个定界的值字符串。 当要构建一个具有多个项目的 SQL 查询时（象某些 WHERE 或 HAVING 语句一样），它们特别有用。 列表变量的语法是:

```
%LIST " value_separator " variable_name
```

建议: 空格是必须的。在大多数情况下，在值分隔符之前和之后都插一个空格。大部分查询都为值分隔符使用布尔或数学运算符（例如，AND、OR 或 >)。 下例说明条件、隐藏和列表变量的使用:

```
%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()
%}
```

在 HTML 表中，如果没有选择任何框，则条件为空，因此查询中的 whereClause 也为空。 否则，whereClause 中包含了选定的值，值之间用 OR 分隔。 例如，如果选择了所有这三个城市，则 SQL 查询为:

```
SELECT name, city FROM citylist
WHERE cond1='Sao Paolo' OR cond2='Seattle' OR cond3='Shanghai'
```

此例显示已选择 Seattle，这出现在该 SQL 的结果中。

```
SELECT name, city FROM citylist
WHERE cond1='Seattle'
```

表格变量

表格变量定义相关数据的集合。它包含一个相同记录或列的数组，和一个在每一行描述字段的列名的数组。 在 Net.Data 宏中如以下语句中所示地定义一个表格:

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

TABLE 后面的数目是对此表格可包含行数的限制。要指定不具有行数限制的表格，可如下例所示地使用缺省值或指定 ALL:

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

在定义表格时，它具有零行和零列，没有分配存储器。用值填充表格的唯一方式是将它作为 OUT 或 INOUT 参数来传送给一个函数。DTW_SQL 语言环境自动将 SELECT 语句的结果放到表格中。

对于其它所有语言环境，例如 DTW_REXX 或 DTW_PERL，语言环境都不自动设置表格值。但是表格的单个元素可作为输出参数用于本机语言解释器，并且必须由正在执行的脚本或程序进行设置。请参阅*语言环境参考*中的语言环境说明以获取详细信息。

您可以通过引用表格变量名来传送一个表格。还可以在一个函数的 REPORT 块中引用表格中的个别元素。 请参阅第67页的『表格处理变量』，以获取细节。在 SQL 函数中表格变量通常是填充了值的，然后，表格变量在 SQL 函数或另一个函数中在作为参数

传送给该函数之后，被用作对报表的输入。您可以将表格变量作为 IN、OUT 或 INOUT 参数传送给任何非 SQL 函数。表格只能作为 OUT 参数传递给 SQL 函数。

表格中的列名和字段值被编址为起始地址为 1 的数组元素，而不象标准 C 和 C++ 语言将起始数组约定为 0。

杂项变量

这些变量是 Net.Data 定义的变量，可用来:

- 影响 Net.Data 的处理
- 查找函数调用的状态
- 获取关于数据库查询结果集合的信息
- 确定关于文件位置和日期的信息

杂项变量即可以具有 Net.Data 确定的预定义值，又可具有您设置的值。例如， Net.Data 根据正在处理的当前文件来确定 DTW_CURRENT_FILENAME 变量值，您可以在该文件中指定 Net.Data 是否除去由制表机和新行字符产生的额外空白。

预定义变量在宏文件中用作变量引用，并提供关于一个函数调用的状态、日期或文件的正确状态。例如，要检索当前文件的名称，可使用:

```
<p>This file is <i>$(DTW_CURRENT_FILENAME)</i>.</P>
```

通常使用 DEFINE 语句或 @DTW_ASSIGN() 函数来设置可修改变量值，可修改变量值可让您影响 Net.Data 如何处理宏文件。例如，要指定是否除去空白，可以使用以下 DEFINE 语句:

```
%DEFINE DTW_REMOVE_WS="YES"
```

请参阅 *Net.Data 参考*中的变量章节，以获取带语法和例子的有效杂项变量列表。

表格处理变量

Net.Data 定义表格处理变量供 REPORT 和 ROW 块使用。 使用这些变量从 SQL 查询和函数调用引用值。

表格处理变量具有一个 Net.Data 确定的预定义值，并允许您引用由正在处理的行、列或字段所调用的函数或者 SQL 查询的结果集中的值。您还可以访问关于正在处理的行数的信息或所有列名列表。

例如，在 Net.Data 处理来自 SQL 查询的结果集时，它为每个当前列名指定变量值，即 N1 赋给第一列，N2 赋给第二列等等。您可以为 HTML 输出引用列名。

在宏文件中使用处理变量作为变量引用。例如，要检索正在处理的当前列名，可使用:

```
<p>Column 1 is <i>$(N1)</i>.</P>
```

表格处理变量还提供关于查询结果的信息。如下例所示，您可以在宏中引用变量 TOTAL_ROWS 来显示在 SQL 查询中返回多少行。

```
Names found: $(TOTAL_ROWS)
```

一些表格处理变量受其它变量或内部函数的影响。例如， TOTAL_ROWS 要求 DTW_SET_TOTAL_ROWS SQL 语言环境变量是激活的，因此在处理来自 SQL 查询或函数调用的结果时，Net.Data 指定 TOTAL_ROWS 的值，这如下例所示:

```
%DEFINE DTW_SET_TOTAL_ROWS="YES"
...

Names found: $(TOTAL_ROWS)
```

请参阅 *Net.Data 参考*中的变量章节，以获取带语法和例子的有效表格处理变量列表。

报表变量

Net.Data 以缺省报表格式显示宏文件生成的 HTML 输出。缺省报表格式使用 <PRE> </PRE> 标记以表格格式进行显示。通过用显示输出的说明来定义 REPORT 块，或通过使用报表变量之一来防止生成缺省报表，可以覆盖缺省报表。

报表变量有助于您定制如何显示 HTML 输出以及如何与缺省报表和 Net.Data 表格一起使用。必须先定义这些变量，然后才可以在 DEFINE 语句或 @DTW_ASSIGN() 函数中使用它们。

报表变量指定空格、覆盖缺省报表格式、指定相对于缺省报表输出的 HTML 表格以及其它功能。例如，您可以使用 ALIGN 变量来控制表格处理变量的前导和尾随空格。下例使用 ALIGN 变量用一个空格来分割列表中每个行名，这是由一个查询来返回的。

```
%DEFINE ALIGN="YES"
...
<p>Your query was on these columns: $(NLIST)
```

START_ROW_NUM 报表变量让您确定从哪一行开始显示查询的结果。例如，以下变量值指定了 Net.Data 将在第三行开始显示查询的结果。

```
%DEFINE START_ROW_NUM = "3"
```

您还可以确定 Net.Data 是否对缺省格式使用 HTML 标记。当 DTW_HTML_TABLE 设置为 YES 时，将生成一个 HTML 表格而不是文本格式的表格。

```
%DEFINE DTW_HTML_TABLE="YES"

%FUNCTION(DTW_SQL){
SELECT NAME, ADDRESS FROM $(qTable)
%}
```

请参阅 *Net.Data 参考*中的变量章节，以获取带语法和例子的有效报表变量列表。

语言环境变量

这些变量与语言环境一起使用并影响语言环境处理请求的方式。必须使用 DEFINE 语句或 @DTW_ASSIGN() 函数，在引用之前定义这些变量。在适当的 Net.Data 宏块中设置或引用语言环境变量。

使用这些变量，您可以执行诸如这样的任务: 建立与数据库的连接、为 Java 小程序提供替换文本和建立 NLS 支持。

例如，您可以使用 SQL_STATE 变量来访问或者显示从数据库返回的 SQL 状态值。

```
%FUNCTION (DTW_SQL) val1() {
  select * from customer
       %REPORT {
```

```
 ...
  %ROW {
 ...
%}
 SQLSTATE=$(SQL_STATE)
%}
```

下一个例子显示怎样定义要访问哪个数据库。

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

请参阅 *Net.Data 参考*中的变量章节，以获取带语法和例子的有效语言环境变量列表。

Net.Data 函数

Net.Data 提供许多功能，您可能发现在 Web 应用程序中这些函数很有用。书写自己的函数也很容易。 第69页的图 20显示了 Net.Data 函数和宏文件是如何相互作用的。

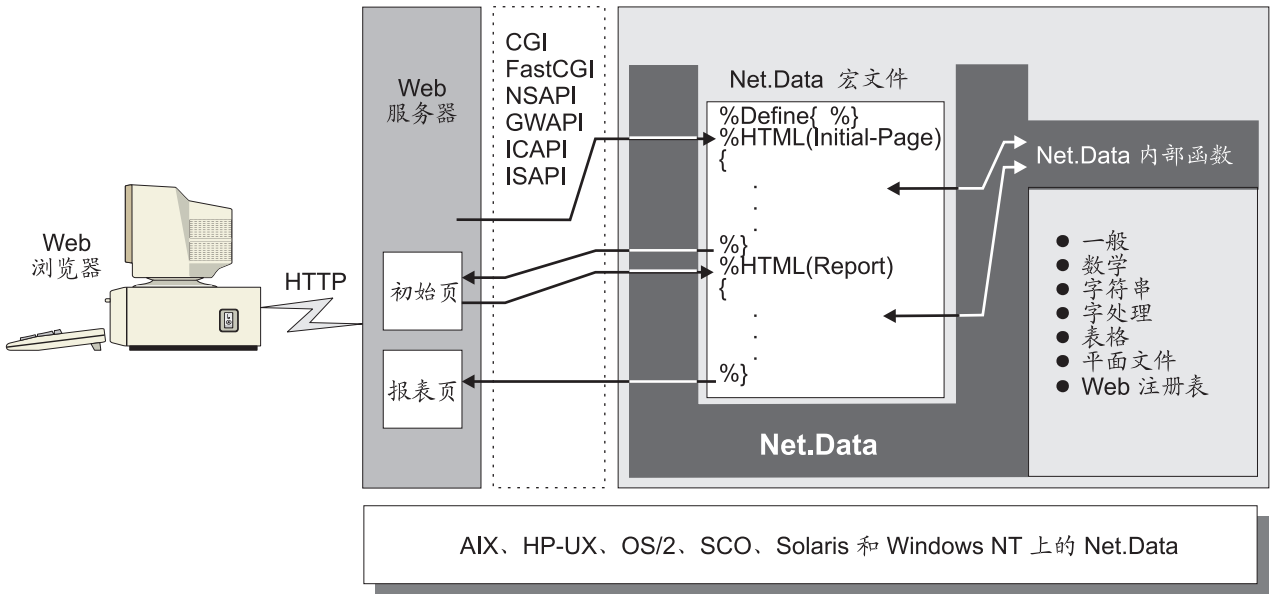


图 20. Net.Data 内部函数

定义函数

您可以定义您自己的函数，也可以使用 Net.Data 的内部函数。要在宏文件中定义您自己的文件(称为用户定义的函数)，可使用 FUNCTION 块或 MACRO_FUNCTION 块。

FUNCTION 块

定义一个子例程，它调用自一个 Net.Data 宏，由语言环境来处理，或调用一个外部程序。

MACRO_FUNCTION 块

定义一个子例程，它调用自一个 Net.Data 宏，必须由 Net.Data 而不是另一个语言环境来处理。块中的语句必须是 Net.Data 宏语言源语句。

对于 FUNCTION 块，MACRO_FUNCTION 块是可选的，它可以提高性能。MACRO_FUNCTION 块只能由 Net.Data 来处理，他不调用一种语言环境。*Net.Data 参考*中提供了这两种结构的细节。

语法如下所示:

FUNCTION 块:

```
%FUNCTION(type) function-name(usage parameter, ...) [RETURNS(return-var)] {
  executable-statements
  report-block
  message-block
%}
```

MACRO_ FUNCTION 块:

```
%MACRO_FUNCTION function-name(usage parameter, ...)  {
  executable-statements
    %}
```

其中:

type

标识了初始化文件中配置的语言环境。语言环境调用一个专用语言处理器(处理可执行语句)并提供 Net.Data 和语言处理器之间的标准接口。

Net.Data 附带提供了几种缺省的语言环境。

function-name

指定了 FUNCTION 或 MACRO_FUNCTION 块的名称。在宏文件以外的地方用函数调用来执行 FUNCTION 块 或 MACRO_FUNCTION 块。函数调用引用 *function-name*，前导以 at (@) 符号。请参阅第72页的『调用函数』，以获取细节。

可以用相同名称定义多个 FUNCTION 或 MACRO_FUNCTION 块，使得它们被同时处理。每个块都必须具有相同的参数表。当 Net.Data 调用函数时，将以在 Net.Data 宏中定义的顺序执行具有相同名称的所有 FUNCTION 块或有相同名称的所有 MACRO_FUNCTION 块。

usage 指定参数是输入 (IN) 参数、输出 (OUT) 参数还是两种类型 (INOUT)。这个指定指出了是将传送至或接收自 FUNCTION 块，MACRO_FUNCTION 块或两者都是。在被改为另一种用法类型之前，此用法类型适用于参数表中的所有后继参数。缺省类型是 IN。

parameter

指具有局部作用域的变量名称，将用在函数调用上指定的相应值来代替它。用参数的实际值来代替可执行语句或 REPORT 块中的参数引用，例如 $(*parm1*)。另外，参数传送至语言环境，并可以用该语言的语法或作为环境变量来被可执行语句访问。在 FUNCTION 块或 MACRO_FUNCTION 块之外，参数变量引用无效。

return-var

在 RETURNS 关键字之后指定此参数来标识特殊的 OUT 参数。返回变量的值赋给函数调用并在 Net.Data 宏处理中替换函数调用。如果您没有指定 RETURNS 子句，则函数调用的值是:

- NULL，如果来自调用向语言环境的返回码是零
- 返回码的值，当回归码非零时。

executable-statements

语言语句的集合，在替换变量和处理函数之后，它们传送至特定语言环境进行处理。 *executable-statements* 可包含 Net.Data 变量引用和 Net.Data 函数调用。在将可执行语句传送到语言环境之前，Net.Data 用实际值来代替这些变量引用或函数调用。*Executable-statements* 包含在 HTML 块中允许的那些可执行语句。

对于 FUNCTION 块，在可执行语句传送到语言环境之前，Net.Data 用变量值代替所有变量引用，执行所有函数调用并用结果值代替函数调用。每个语言环境处理语句的方式是不同的。关于指定可执行语句或调用可执行程序的更多信息，请参阅第63页的『可执行变量』。

对于 MACRO_FUNCTION 块来说，可执行语句是 HTML 语句和 Net.Data 宏语言结构的组合。在此情况下在此情况下，不涉及语言环境，因为 Net.Data 起语言处理器的作用，并估计和执行可执行语句。

report-block

定义 REPORT 块，以处理 FUNCTION 块的输出。请参阅第81页的『报表块』。

message-block

定义 MESSAGE 块，它处理 FUNCTION 块返回的任何消息。请参阅第79页的『消息块』。

在 Net.Data 宏中调用函数之前在 Net.Data 宏的最外层定义该函数。

在语言语句中使用特殊字符

当匹配 Net.Data 语言结构语法的字符在函数块的语言结构节中作为一部分语法上有效的嵌入程序码(例如 REXX 或 Perl)使用时，它们可能被作为 Net.Data 语言结构而被误解释，因而导致错误或宏中不可预测的结果。

例如，Perl 函数可能使用 COMMENT 块定界符 %{。 运行宏时，%{ 被作为 COMMENT 快的开头来解释。然后 Net.Data 查找 COMMENT 块的结尾，当它读到函数块结尾时就认为是找到了。Net.Data 然后继续查找函数块的结尾，但当找不到时，就发出一个错误。

使用以下方式之一来使用 COMMENT 块定界符字符，或使用任何其它 Net.Data 特殊字符作为嵌入程序代码的一部分，而不让它们被 Net.Data 作为特殊字符来解释:

- 使用 EXEC 语句来调用程序代码，而不是将代码放在内部。
- 使用一个变量引用来指定特殊字符。

例如，以下 Perl 函数包含表示一个 COMMENT 块定界符 %{ 的字符作为 Perl 语言语句的一部分:

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

要保证 Net.Data 将 %{ 字符作为 Perl 源码而不是作为 Net.Data COMMENT 块定界符，可以用以下方式之一重写函数:

- o 使用 %EXEC 语句:

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

- o 使用一个变量引用来指定 %{ 字符:

```
%define percent_openbrace = "%{"

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

调用函数

使用 at (@) 字符后跟 FUNCTION 块名称或 MACRO_FUNCTION 块名称来从 Net.Data 宏中调用一个函数:

@function_name([*argument*,...])

function_name

这是 FUNCTION 块的名称

或要调用的 MACRO_FUNCTION 块。除非是内部函数，否则必须在 Net.Data 宏中已定义函数。

argument

这是定义的变量、字面量字符串、变量引用或函数调用的名称。函数调用上的变元与 FUNCTION 块或 MACRO_FUNCTION 块上的参数相匹配，在处理 FUNCTION 块或 MACRO_FUNCTION 块时，对每个参数指定对应的变元值。变元与对应的参数必须具有相同数目和类型。

Net.Data 用以下顺序来处理与一个函数调用相关联的 FUNCTION 块、MACRO_FUNCTION 块或内部函数。

1. Net.Data 在 FUNCTION 块的可执行语句节中处理变量引用和函数调用。Net.Data 用变量的当前值来代替所有变量引用，并用函数调用的返回值来执行和代替所有函数调用。变量引用和函数调用是以被指定的顺序进行处理的。在该步骤中，Net.Data 不处理内部函数或 MACRO_FUNCTION 块。
2. 本机语言处理器处理可执行语句节。对于 FUNCTION 块，处理器对应于 FUNCTION 块上特定的语言环境，例如 SQL、REXX 或 Perl。 对于 MACRO_FUNCTION 块，Net.Data 起语言处理器的作用，并执行可执行语句。 内部函数不具有可执行语句。Net.Data 用函数名来处理内部函数。

   Net.Data 将函数的参数传送给本机语言处理器。 Net.Data 只将 IN 和 INOUT 参数的值传送给本机语言处理器，并只从本机语言处理器接受 OUT 和 INOUT 参数的返回值。

3. Net.Data 根据来自语言处理器的返回消息和返回码来设置隐式的 RETURN_CODE 和 DTW_DEFAULT_MESSAGE 变量。Net.Data 不为 MACRO_FUNCTION 块设置这些变量。
4. 如果 FUNCTION 块包含一个 REPORT 块，或指定要生成的缺省报表，Net.Data 将使用任何引用的输出参数的新值来处理报表节。 Net.Data 不为内部函数生成报表。
5. 如果 FUNCTION 块包含一个局部 MESSAGE 块，Net.Data 将处理 MESSAGE 块。当以下条件之一发生时，Net.Data 处理全局 MESSAGE 块:
   - 已指定一个全局 MESSAGE 块，但是局部 MESSAGE 块未处理返回码。
   - 调用了一个内部函数。

   Net.Data 不处理 MESSAGE 块或 MACRO_FUNCTION 块。
6. Net.Data 用函数的返回值来代替函数调用。对于 FUNCTION 块，此值是下面的某一个:

   RETURNS 参数值
   用 RETURNS 关键字来代替 FUNCTION 块。

   空字符串 ("")
   在 RETURN_CODE 是零时，不带 RETURNS 关键字来代替 FUNCTION 块。

   RETURN_CODE
   在 RETURN_CODE *非*零时，不带 RETURNS 关键字来代替 FUNCTION 块。

   对于 MACRO_FUNCTION 块，以处理可执行语句的结果代替函数调用。

   对于内部函数，值取决于内部函数的格式。

调用存储过程

存储过程是一个已编译程序，存储在 DB2 局部或远程服务器上，可以执行 SQL 语句。在 Net.Data 中是使用 CALL SQL 语句从 Net.Data 函数中来调用存储过程的。存储过程的参数传送自 Net.Data 函数参数表。通过以数据库服务器保持编译的 SQL 语句，可使用存储过程来改进性能和完整性。

本节描述以下主题:

存储过程语法

存储过程的语法使用 FUNCTION 语句、CALL 语句以及任选的 REPORT 块。

```
%function (dtw_sql) function_name ([IN datatype arg1, INOUT datatype arg2,
    OUT tablename...])
  CALL stored_procedure [(resultsetname....)]
[%REPORT(resultsetname...)]
```

其中:

function_name
是 Net.Data 函数的名称，它初启存储过程的调用。

stored_procedure
是存储过程的名称

datatype
是 Net.Data 支持的数据库数据类型，正如第74页的表 5中所显示的那样。参数表中指定的数据类型必须匹配存储过程中的数据类型。请参阅数据库文档以获取关于这些数据类型的更多信息。

tablename
是 Net.Data 表格的名称，结果集存储在表格中(仅当要在 Net.Data 表格中存储结果集时才使用)。如果指定的话，此参数名称必须匹配 `resultsetname` 的关联的参数名称。

resultsetname
一个名称，它将返回自存储过程的结果集与报表块相关联。如果指定的话，此参数名称必须匹配 `tablename` 的关联的参数名称。

表 5. 存储过程的数据类型

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

调用存储过程

要调用一个存储过程:

1. 定义一个函数，让它初启对存储过程的调用。

   ```
   %function (dtw_sql) function_name()
   ```

2. 任选地为存储过程指定任何 IN、INOUT 或 OUT 参数。包含一个表格名称以在 Net.Data 表格中存储结果集(如果您希望结果集存储在 Net.Data 表格中，只需要指定一个 Net.Data 表格)。

   ```
   %function (dtw_sql) function_name (IN datatype
   arg1, INOUT datatype arg2, OUT tablename...)
   ```

3. 使用 CALL 语句来标识存储过程名称。

   ```
   CALL stored_procedure
   ```

4. 如果存储过程将要生成一个结果集，则任选地指定一个 REPORT 块来定义 Net.Data 如何显示结果集。

   ```
   %REPORT
   ```

   例子:

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

```
    %row {  ....   %}
    ...
  %}
%}
```

5. 如果存储过程将要生成多个结果集:
   - 在 CALL 语句上指定一个或多个结果集名。

     ```
     CALL stored_procedure (resultsetname1, resultsetname2, ...)
     ```

   - 任选地指定一个或多个 REPORT 块来定义 Net.Data 如何显示结果集。

     ```
     %REPORT(resultsetname1)
     ```

     例子:

     ```
     %function (dtw_sql) mystoredproc (IN CHAR(30) arg1, OUT table1) {
        CALL myproc (table1, table2)
      %report (table2) {
           ...
           %row {  ....   %}
           ...
      %}
     %}
     ```

传送参数

您可以将参数传送给一个存储过程，并可以让存储过程更新参数值，以使新值传送回 Net.Data 宏。通过用 IN 关键字指定参数名来将参数传送至存储过程中。如果存储过程将要更新参数，则您必须用 INOUT 或 OUT 关键字来为返回值传送参数。为参数指定的数据类型必须匹配存储过程期望的类型。

例子 1: 将参数值传送给存储过程

```
%function (dtw_sql) mystoredproc (IN CHAR(30) valuein)  {
    CALL myproc
...
```

例子 2: 从存储过程返回一个值

```
%function (dtw_sql) mystoredproc (OUT VARCHAR(9) retvalue)  {
  CALL myproc
...
```

处理结果集

存储过程中可能返回一个或多个结果。结果集可存储在 Net.Data 表格中，以备在宏中进一步处理或使用 REPORT 块来显示。必须为存储过程所生成的每个结果集关联一个名称。这是通过在 CALL SQL 语句上指定参数来完成的。为结果集指定的名称就与一个 REPORT 块或 Net.Data 表格关联起来，允许您确定 Net.Data 如何处理每个结果集。你可以:

- 通过不为结果集定义报表块，来以 Net.Data 的缺省报表风格显示结果。
- 将结果集与 REPORT 块相关联，让 Net.Data 在报表中显示结果集。然后可用 Net.Data 变量、HTML 或其它函数来指定在浏览器中如何显示报表。
- 当你希望 Net.Data 以后使用宏文件中的数据时，将结果集存储在 Net.Data 表格中。例如，将 Net.Data 表格传送到另一个函数，以使它可使用数据进行计算并根据这些计算来显示结果。

如果一个存储过程生成多个结果集，就必须在 CALL SQL 语句上为每个结果集指定一个参数。

要返回单个结果集并用一个缺省报表来显示它:

使用以下语法:

```
%function (dtw_sql) function_name () {
    CALL stored_procedure ()
%}
```

例如:

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

要返回单个结果集并指定一个 REPORT 块以备显示处理:

使用以下语法:

```
%function (dtw_sql) function_name () {
     CALL stored_procedure
 %REPORT (resultsetname) {
  ...
 %}
%}
```

例如:

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

同样可使用以下语法:

```
%function (dtw_sql) function_name () {
    CALL stored_procedure (resultsetname)

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

例如:

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

要在 Net.Data 表格中存储一个结果集以备进一步的处理:

使用以下语法:

```
%function (dtw_sql) function_name (OUT tablename) {
   CALL stored_procedure (tablename)
%}
```

例如:

```
%define DTW_DEFAULT_REPORT = "NO"

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

注意，DTW_DEFAULT_REPORT 设置为 NO，因此没有为结果集生成缺省报表。

要返回多个结果集并用缺省报表格式显示它们:

使用以下语法:

```
%function (dtw_sql) function_name () {
    CALL stored_procedure (tablename1, tablename2)
%}
```

例如:

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

要返回多个结果集并在 Net.Data 表格中存储结果集以备进一步处理:

使用以下语法:

```
 %function (dtw_sql) function_name (OUT tablename1, tablename2) {
    CALL stored_procedure (resultsetname1, resultsetname2)
%}
```

例如:

```
%define DTW_DEFAULT_REPORT = "NO"

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

注意，DTW_DEFAULT_REPORT 设置为 NO，因此没有为结果集生成缺省报表。

要返回多个结果集并指定 REPORT 块以备显示处理:

每个结果集都与它自己的 REPORT 块相关联的。使用以下语法:

```
%function (dtw_sql) function_name () {
   CALL stored_procedure (resultsetname1, resultsetname2)
 %REPORT (resultsetname1)
   ...
   %row {  ....   %}
   ...
 %}
 %REPORT (resultsetname2)
   ...
   %row {  ....   %}
   ...
 %}
%}
```

例如:

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

 %REPORT(mytable1)
  ...
  %row {  ....   %}
  ...
 %}

 %REPORT(mytable2)
  ...
  %row {  ....   %}
  ...
 %}
%}
```

要返回多个结果集并为每个结果集指定不同的显示或处理选项:

可以使用唯一的参数名为每个结果集指定不同的处理选项。例如:

```
%function (dtw_sql) mystoredproc (OUT mytable2) {
    CALL myproc (mytable1, mytable2, mytable3)

 %REPORT(mytable1)
 ...
 %row {  ....   %}
 ...
 %}
%}
```

结果集 mytable1 由对应的 REPORT 块来处理，并如宏书写者所指定的那样显示。结果集 mytable2 存储在 Net.Data 表格 mytable2 中，现在可用于进一步的处理，例如传送给另一个函数。结果集 mytable3 是用 Net.Data 的缺省报表格式显示的，因为没有为它指定 REPORT 块。

返回多个结果集的准则和限制

在一个存储过程中返回多个结果集时，请使用以下准则和限制。

准则:

- 为每个结果集指定一个 REPORT 块。为 REPORT 块指定的表名参数必须匹配 CALL 语句上对应的表名参数。
- 以您希望处理结果集的顺序来为多个结果集指定 REPORT 块。
- 当没有为结果集指定 REPORT 块时，要指定缺省处理，可定义 DTW_DEFAULT_REPORT = ″YES″。当 Net.Data 构建 Web 页面时，当它为具有 REPORT 块的结果集显示了报表之后，将为结果集表格显示缺省报表。
- 要防止 Net.Data 显示不具有 REPORT 块的结果集，必须设置 DTW_DEFAULT_REPORT = ″NO″。
- 当在存储过程中使用 DTW_SET_TABLE_IN 变量时，存储过程所返回的第一个结果集分配给 DTW_SET_TABLE_IN 表格。

限制:

- 只有在具有 DTW_SQL DB2 环境变量的存储过程中才可以使用多个 REPORT 块。
- 报表变量是为整个函数设置的，并影响所有 REPORT 块的处理以及处理的结果集。您不能修改单独的 REPORT 块的一个报表变量。

- MESSAGE 块必须位于一个 REPORT 块列表的之前或之后，而不能在 REPORT 块的中间。
- 必须首先用 TABLE 语句定义表格变量，然后才可以在存储过程中使用它们
- 不能为相同的结果集指定多个 REPORT 块。
- 单个存储过程的结果集的最大数目是 32。

消息块

MESSAGE 块让您根据函数调用的成功或失败来确定在函数调用之后如何继续下去，并让您为函数的调用程序显示信息。Net.Data 使用以下消息块过程:

1. Net.Data 为对 FUNCTION 块的每个函数调用设置一个语言环境变量 RETURN_CODE。在对 MACRO_FUNCTION 块的函数调用上不设置 RETURN_CODE。
2. 当语言环境将一个返回码值传送回 Net.Data 时，Net.Data 将 RETURN_CODE 的值设置为返回码值。
3. 函数调用完成时，MESSAGE 块使用 RETURN_CODE 的值来确定如何继续下去。

一个 MESSAGE 块由一系列消息语句组成，每个消息语句指定一个返回码值、信息正文和一个要进行的操作。*Net.Data 参考*中的语言结构章中显示了 MESSAGE 块的语法。

MESSAGE 块可具有全局或局部作用域。如果 MESSAGE 块是在 FUNCTION 块中定义的，则它的作用域局部在该 FUNCTION 块中。如果它是在最外层指定的，则 MESSAGE 块是全局作用域，并且对于 Net.Data 宏中执行的所有函数调用都是活动的。如果您定义多个全局 MESSAGE 块，则最后定义的块是活动的。

Net.Data 使用这些规则来处理来自一个函数调用的 RETURN_CODE 变量的值:

1. 在局部 MESSAGE 块检查一个精确匹配；根据指定的来退出或继续。
2. 如果 RETURN_CODE 非 0，则检查局部 MESSAGE 块中的 `+default` 或 `-default`；根据 RETURN_CODE 的符号，按指定的来退出或继续。
3. 如果 RETURN_CODE 非 0，则检查检查局部 MESSAGE 块中的 `default`；按指定的来退出或继续。
4. 在全局 MESSAGE 块检查一个精确匹配；根据指定的来退出或继续。
5. 如果 RETURN_CODE 非 0，则检查全局 MESSAGE 块中的 `+default` 或 `-default`；根据 RETURN_CODE 的符号，按指定的来退出或继续。
6. 如果 RETURN_CODE 非 0，则检查检查全局 MESSAGE 块中的 `default`；按指定的来退出或继续。
7. 如果 RETURN_CODE 非 0，则发出 Net.Data 内部缺省消息和出口。

下例显示 Net.Data 宏的一部分，其中具有一个全局 MESSAGE 块和一个函数的 MESSAGE 块:

```
%{ 全局消息块 %}
%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
  %}
```

如果 *my_function()* 返回 RETURN_CODE 值为 50，Net.Data 将以此顺序处理错误:

1. 在局部 MESSAGE 块中检查确切匹配。
2. 在局部 MESSAGE 块中检查 +default。
3. 在局部 MESSAGE 块中检查 default。
4. 在全局 MESSAGE 块中检查确切匹配。
5. 在全局 MESSAGE 块中检查 +default。

当 Net.Data 找到一个匹配时，它向 Web 浏览器发送信息正文，并检查请求的操作。

当您指定了 continue 之后，Net.Data 继续处理 Net.Data 宏，然后才打印信息正文。例如，一个宏调用 my_functions() 5 次并在用 MESSAGE 块处理期间发现错误 100，则程序的输出是这样的:

```
.
.
.
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
```

在宏中生成 HTML

Net.Data 让您很容易地在应用程序用户浏览器中表达标准 HTML 页。以下各节描述宏的 HTML 和 REPORT 块，并显示如何在 Net.Data 宏中格式化 HTML。请参阅*Net.Data 参考*中的语言结构章节以获取关于这些块的语法信息。

HTML 块

Net.Data 宏文件包含 HTML 块，以及 HTML 块中的函数，它们在 Web 浏览器上生成 HTML 输出。在宏文件中，必须指定至少一个 HTML 块。HTML 块的内容控制 Net.Data 调用的其余部分。

任何有效的 HTML 语句都可以出现在 HTML 块中。另外，在 HTML 块中还可以使用 INCLUDE 语句、函数调用和变量引用。下例显示 Net.Data 宏中的 HTML 块的一般用法:

```
%DEFINE DATABASE="MNS96"

%HTML(INPUT) {
<H1>Hardware Query Form</H1>
<HR>
<FORM METHOD="POST" ACTION="/cgi-bin/db2www/equiplst.d2w/report">
```

```
<dl>
<dt>What hardware do you want to list?
<dd><input type="radio" name="hdware" value="MON" checked>Monitors
<dd><input type="radio" name="hdware" value="PNT">Pointing devices
<dd><input type="radio" name="hdware" value="PRT">Printers
<dd><input type="radio" name="hdware" value="SCN">Scanners
</dl>
<HR>
<input type="submit" value="Submit">
</FORM>
%}

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

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

您可以如下例所示来从 HTML 链接中调用 Net.Data 宏:

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

当应用程序用户单击此链接时，Web 浏览器调用 Net.Data，Net.Data 语法分析宏文件。当 Net.Data 开始处理在调用上指定的 HTML 块时(在此情况下为 HTML(INPUT) 块)，它开始处理其中的文本。Net.Data 对于不能识别为 Net.Data 宏语言结构的任何东西，都假定为 HTML 语句，并发送给浏览器来显示。

在用户作出选择并按了 Submit 按钮之后，Net.Data 运行 HTML FORM 元素的 ACTION 部分，这指出了对 Net.Data 宏的 HTML(REPORT) 块的调用。然后，Net.Data 象处理 HTML(INPUT) 块那样处理 HTML(REPORT) 块。

Net.Data 然后处理 `myQuery()` 函数调用，该函数调用依次调用 SQL FUNCTION 块。用在输入表中返回的值代替 SQL 语句中引用的 `$(hdware)` 变量之后，Net.Data 询问运行。在此点，Net.Data 再次开始将 HTML 报表发送给浏览器，浏览器根据 REPORT 块中指定的 HTML 语句来显示查询结果。

Net.Data 完成 REPORT 块的处理之后，返回至 HTML(REPORT) 块，并结束处理。

Net. Data 在每次被调用时只处理一个 HTML 块。但是，通过使用 HTML 链接和表，可以让人很容易地启动另一个 HTML 块上的另一个 Net.Data 调用，这都由您控制。

报表块

使用 REPORT 块语言结构来格式化并显示来自 FUNCTION 块的数据输出。虽然可以指定 HTML 标记、宏变量引用和函数调用的任何有效组合，但是此输出是典型的表格数据。通常可以任选地在 REPORT 块上指定表名。如果未指定表名，Net.Data 将使用

FUNCTION 块的参数表中的第一个输出表格中的表数据。如果在 FUNCTION 块上没有指定一个表格，Net.Data 将使用缺省表数据。

REPORT 块具有三部分，每部分都是可选的:

- 首部信息，包含在表格行数据之前显示的 HTML 数据。
- ROW 块，包含 HTML 和表格变量，在结果表格的每行上显示。
- 脚注信息，包含在表格行数据之后显示的信息。

要避免显示来自 ROW 块的任何表格输出，可让 ROW 块为空或整个省略它。

您可以在 REPORT 块中使用几个 Net.Data 提供的变量来访问 Net.Data 宏结果表格中的数据。第67页的『表格处理变量』中描述了这些变量。关于附加细节，请参阅 *Net.Data 参考*中的“报表变量”节。

要提供首部和脚注信息，必须在 ROW 块之前和之后提供文本。Net.Data 处理 ROW 块之前发现的任何东西，并将它们作为首部信息来对待，将 ROW 块之后发现的任何东西作为脚注信息来对待。象 HTML 块一样，Net.Data 将首部、ROW 和脚注块中未识别为宏语言结构的任何东西作为 HTML 对待，并将 HTML 数据发送给浏览器。

还可以在 REPORT 块中使用函数和变量。

要让 Net.Data 打印使用预格式化文本的缺省报表，就不要在宏文件中包含 REPORT 块。以下例子显示缺省报表格式:

```
 SHIPDATE   | RECDATE    | SHIPNO   |
-------------------------------------
 25/05/1997 | 30/05/1997 | 1495194B |
-------------------------------------
 25/05/1997 | 28/05/1997 | 2942821G |
-------------------------------------
```

要使用 HTML 标记来代替预格式化文本，可将 DTW_HTML_TABLE 设置为 YES。

要禁用缺省报表的打印，可将 DTW_DEFAULT_REPORT 设置为 NO，或指定一个空的 REPORT 块。例如:

```
%REPORT{%}
```

下例显示如何使用特殊变量和 HTML 标记来定制报表格式。它显示来自表格 CustomerTbl 的姓名、电话号码和传真号码:

```
%FUNCTION(DTW_SQL) custlist() {
     SELECT Name, Phone, Fax FROM CustomerTbl
%REPORT{
<I>Phone Query Results:</I>
<BR>
=====================
<BR>
%ROW{
   Name: <B>$(V1)</B>
<BR>
Phone: $(V2)
<BR>
Fax: $(V3)
<BR>
------------------------------
<BR>
   %}
   Total records retrieved: $(NUM_ROWS)
   %}
   %}
```

Web 浏览器中的结果报表如下所示:

```
Phone Query Results:
====================
Name: Doen, David
Phone: 422-245-1293
Fax: 422-245-7383
------------------------------
Name: Ramirez, Paolo
Phone: 955-768-3489
Fax: 955-768-3974
------------------------------
Name: Wu, Jianli
Phone: 525-472-1234
Fax: 525-472-1234
------------------------------
Total records retrieved: 3
```

Net.Data生成报表是通过:

1. 在报表的开头打印一次 *Phone Query Results:*。
2. 对于检索到的每一行，分别为 Name, Phone, and Fax 给定变量 V1、V2 和 V3 的值。
3. 为了提高可读性，在检索到的每行之后图一条线。
4. 在报表结尾打印一次字符串 *Total records retrieved:* 以及 NUM_ROWS 的值。

您可以使用存储过程的多个 REPORT 块来返回多个结果集合。在使用 DTW_SQL 语言环境时，只对存储过程支持多个报表块。请参阅第73页的『调用存储过程』以学习如何从一个存储过程返回多个结果集合。

宏文件中的条件逻辑和循环

Net.Data 让你使用 IF 和 WHILE 块来在 Net.Data 宏中结合条件逻辑和循环。

- 第83页的『条件逻辑』
- 第85页的『循环结构』

条件逻辑

使用 IF 块来在 Net.Data 宏中进行条件处理。在大多数高级语言中 IF 块类似于 IF 语句，因为 IF 块提供测试一个或多个条件的能力，然后基于条件测试的结果执行一个语句块。

您可以在宏中的几乎任何地方指定 IF 块并可以嵌套它们。 *Net.Data 参考*中的语言结构章中显示了 IF 块的语法。

IF 块的语法规则是由块在宏文件中的位置确定的。IF 块中语句的可执行块允许的元素取决于 IF 块自身的位置。包含 IF 块的块中的任何有效元素在该 IF 块中也有效。例如，如果您在一个 HTML 块中指定了一个 IF 块，则 HTML 块中允许的任何元素在 IF 块中也是允许的，例如 INCLUDE 语句和 WHILE 块。

```
  %HTML 块
...
  %IF block
...
    %INCLUDE
```

```
...
    %WHILE
```

类似地，如果您在 Net.Data 宏声明部分中的任何其它块之外指定 IF 块，则在 IF 语句中只允许那些在任何其它块之外也被允许的元素(例如 DEFINE 块或 FUNCTION 块)。

```
%IF
...
  %DEFINE
...
  %FUNCTION
```

如果 IF 块嵌套在一个 IF 块内，而后者在声明部分中任何其它块的外部，则它可以使用外部块可以使用的任何元素。 如果 IF 块嵌套在另一个嵌套在某个 IF 块的块中，则它遵循它所处的那个模块的语法规则。

在下面的例子中，嵌套的 IF 块必须遵循在它处于一个 HTML 块中时使用的规则。

```
%IF
...
  %HTML block
...
     %IF block
```

例外: 当 IF 块在 REPORT 块中时，不要在 IF 块中指定 ROW 块。

Net.Data 根据组成条件的项目的内容，用两种方式中的一种来处理 IF 块条件列表。缺省操作是将所有项目作为字符串对待，并如条件中所指定的那样执行字符串比较。但是，如果符合以下两个条件，Net.Data 将执行数值比较:

- 如果条件是一个二进制运算 (<、>、<=、>=、!=、==)。
- 如果条件中的两个项目都代表整数。这意味着项目是数字串的，任选地前导以一个 '+' 或 '-' 字符。字符串不能包含任何非数字字符，'+' 或 '-' 除外。

  有效字符串的例子:

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

  无效字符串的例子:

  ```
  -  20     (包含空格字符)
  234,000   (包含一个逗号)
  57.987    (包含一个十进制小数点)
  ```

Net.Data 在执行 IF 块的时侯求 IF 块，此时间可能与 Net.Data 初试读块的时间不同。例如，如果您在 REPORT 块中指定一个 IF 块，Net.Data 在读取包含 REPORT 块的 FUNCTION 块定义时不估计与 IF 块相关联的条件列表，而是在调用和执行它时进行。对于 IF 块的条件列表部分和要执行的语句块，都是这样的。

限制: Net.Data 不支持非整型数的数值比较。

例子: 此例显示一个宏文件，其中包含其它块中的 IF 块:

```
%{ 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)
  %}
%}
```

循环结构

在 Net.Data 宏中使用 WHILE 块来执行循环。类似于 IF 块，WHILE 块也提供测试一个或多个条件的能力，然后基于条件测试的结果执行一个语句块。 与 IF 不同的是，基于条件测试的结果，语句块可被执行多次。

可以在 HTML 块、REPORT 块、ROW 块 MACRO_FUNCTION 块和 HTML IF 块中指定 WHILE 块，并可以嵌套它们。*Net.Data 参考*中的语言结构章中显示了 WHILE 块的语法。

Net.Data 处理 WHILE 块的方式与处理 IF 块的方式精确相同，只在每此通过循环时重新估计条件列表。而且与任何条件循环结构相同，如果条件编码不正确的话，就有可能进入死循环。

例子: 该例显示具有一个 WHILE 块的宏文件:

```
%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)
%}
%}
```

使用大型对象

大型多媒体和文档对象是二进制文件(例如 BMP、TIF、GIF 和 PostScript 文件)，用于显示和打印。您可以在大多数操作系统中将这些文件存储在 DB2 数据库中，然后通过 Web 应用程序的 SQL 语言环境来访问它们。

Net.Data 支持以下类型的 LOB:

- 二进制大型对象 (BLOB)
- 字符大型对象 (CLOB)
- 双字节大型对象 (DBLOB)

LOB 通常在最先几个字节中包含一个文件签名，以指出文件包含什么类型的信息。如果 Net.Data 识别到一个 LOB，Net.Data 将把扩展名添加到临时文件以及表示其名称的 Net.Data 宏变量。如果在宏文件中不具有 REPORT 块，Net.Data 将把扩展名 .txt 添加至 CLOB 文件。Net.Data 识别以下 LOB 格式:

- 位图 (.bmp)
- 图形图象格式 (.gif)
- TIFF 文件格式 (.tif)
- Postscript (.ps) 它们必须作为 BLOB 而不是 CLOB 存储。

限制: Net.Data 不能识别其它文件类型并且不支持它们。对于任何大型对象，Net.Data 不支持 UPDATE 和 INSERT SQL 语句。

规划提示: 当查询返回一个 LOB 时，Net.Data 将把它保存在 HTML_PATH 配置变量指定的目录中。在使用 LOB 时请考虑系统限度，因为它们会很块地消耗资源。诸如声频文件等一些 LOB 需要特殊的硬件和软件。

例子 1: 例子一个内部图形

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

例子 2: 在以下例子中，应用程序用户必须单击文件名来调用察看器，因为应用程序使用一个 .WAV 文件。Net.Data 不能识别此文件类型，因此使用了一个 EXEC 变量来将扩展名附加到文件。

```
%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()
%}
```

queryData 函数返回以下 HTML:

```
<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>
```

先前例子中的 REPORT 块使用隐式表格变量 V1、V2 和 V3。

- V1 是人名，是普通文本。
- V2 是 .GIF 文件中的人员照片。图象是内插的。 Net.Data 自动包含临时 LOB 目录和 .GIF 扩展名。
- V3 是 .WAV 文件中人的语音的示例。当 Net.Data 遇到未被识别的文件格式(例如 .WAV 文件)时，它将文件写入临时 LOB 目录，不带扩展名。此例显示怎样通过用 EXEC 变量添加扩展名来处理此文件类型。当分辨出变量 *$(V3)* 时，它在文件名之前添加 `/tmplobs/` 。例如，`/tmplobs/sound2a`。对付此种文件的一个方式是写一个 REXX 程序，以将斜杠改为反斜杠并重新命名文件。当应用程序用户单击 Voice 示例时，将播放语音。

不是所有 Web 浏览器都支持图形和声音的。要支持这里描述的功能，可能需要特殊的硬件和软件，例如语音卡和驱动程序。

第6章 使用内部函数

Net.Data 提供了大量的内部函数来简化 Web 页面的开发。这些函数已经由 Net.Data 定义好了，因此您不需要在 FUNCTION 块中再对它们进行定义。在宏文件中，您可以在任何能够调用用户自定义函数的地方简单地调用这些函数。

内部函数可以三种方式返回它们的结果。您可以通过前缀来分辨出它们各自是如何返回结果的:

- **DTW_**、**DTWF_** 和 **DTWR_**: 调用结果在一个输出参数中返回，或者不返回结果。(**DTWF_** 是用于平面文件函数的前缀。**DTWR_** 是用于 Web 注册表函数的前缀。)
- **DTW_r**、**DTWF_r** 和 **DTWR_r**: 函数调用的结果将替换宏中的函数调用，这就和指定了 RETURNS 关键字的用户自定义函数中用 RETURNS 关键字的值替换函数调用是相同的。
- **DTW_m**: 在传递给函数的参数中返回多个结果。

有些内部函数并不具有每一类型。有关的更多信息，请参阅*Net.Data 参考*。

下面的列表对 Net.Data 内部函数提供了一个高级概述。使用这些函数可以执行通用、数学、字符串、字处理或表格处理功能。请参阅 *Net.Data 参考*以获取每个函数的语法说明和例子。

通用函数
这个函数集合通过改变数据或访问系统服务来帮助您开发 Web 页面。您可以使用它们来查询和设置环境变量、使用 HTML 退出代码并从系统获取其它有用的信息。

数学函数
这些函数执行数学运算，使您能够计算或改变数字数据。除了标准的数学运算以外，您还可以执行按模除法、指定结果精度并使用科学记数法。

字符串函数
这些函数可以让您处理字符串中的字符。您可以更改字符串的大小写、插入或删除字符、给另一个变量指定字符串值、增加其它有用的函数。

字处理函数
这些函数可以让您处理字符串中的单词。这些函数大部分和字符串函数以相同的方式作用，但它们是对整个单词进行作用。例如，它们可以让您计数一个字符串中的单词个数、删除单词、在字符串中搜索某个单词。

表格函数
您可以使用这些函数来生成报表或表格(这些报表或表格使用 Net.Data 表格变量中的数据)。表格变量中包含了一个值数组以及相关的列名。它们提供了一种便利的方式将一组值传递给一个函数。

平面文件函数
使用平面文件接口 (FFI) 可以打开、读取和处理平面文件源(文本文件)中的数据，也可以将数据存储到平面文件中。

Web 注册表函数
使用 Web 注册表函数来维护注册表及其包含的条目。

Web 注册表是一个文件，由 Net.Data 维护此文件的一个关键字，允许您方便地添加、检索和删除其中的条目。

第7章 使用语言环境

Net.Data 提供了用于访问数据源以及执行包含商业逻辑的应用程序的语言环境。例如，SQL 语言环境可以让您将 SQL 语句传递到一个 DB2 数据库，REXX 语言环境可以让您调用 REXX 程序。您还可以使用 SYSTEM 语言环境来执行一个程序或发出一条命令。

使用了 Net.Data，您就能够以一种可插入的方式来添加用户编写的语言环境。每个用户编写的语言环境都必须支持 Net.Data 定义的一系列接口，必须作为动态链接库 (DLL) 或共享程序库实现。您必须在 Net.Data 初始化文件中添加一个 ENVIRONMENT 语句以使一个 DLL 与您所编写的语言环境相关联。Net.Data 在首次遇到对指定语言环境名的 FUNCTION 块的函数调用时，它将装入并执行一个 DLL。对指定同一个语言环境名的 FUNCTION 块的后继函数调用都只是使得 Net.Data 执行 DLL，因为这些 DLL 都只装入一次。

第91页的图 21显示了 Web 服务器、Net.Data 以及 Net.Data 语言环境之间的关系。

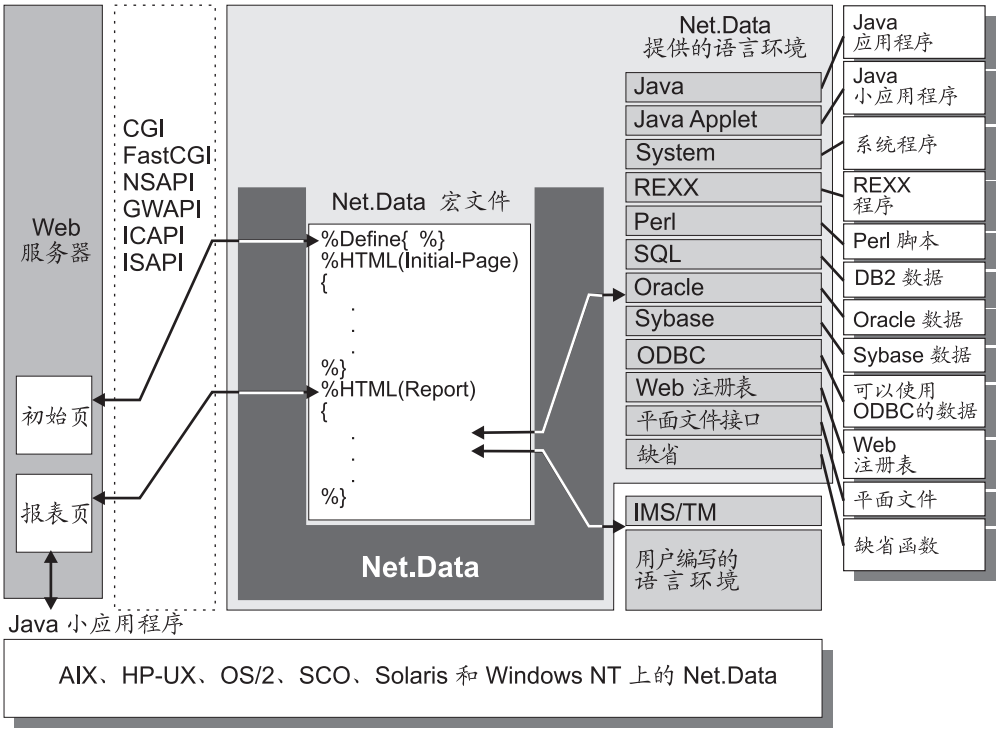


图 21. Net.Data 语言环境

有关 Net.Data 提供的语言环境以及如何创建用户编写的语言环境的完整细节，请参阅 *Net.Data 语言环境参考*。

第8章 使用 Java 小服务程序和 JavaBean 来调用 Net.Data

Net.Data 提供了一些基于 Java 的类，您可以使用这些类在 Net.Data 支持的任何支持 Java 功能的操作系统中来调用 Net.Data 宏文件、运行 Net.Data 函数或在 Net.Data 中执行 SQL 语句。

本章将描述以下内容的概念和任务:

第93页的『Net.Data 小服务程序』

第99页的『Net.Data JavaBean』

Net.Data 小服务程序

Net.Data 为小服务程序和 NetObjects Fusion 插件提供了可以在 Java 环境中使用的 Net.Data 函数。使用了这些小服务程序和插件，您可以:

- 从 URL 并作为服务器方包含文件 (SSI) 运行 Net.Data 宏文件
- 从 URL 并作为 SSI 运行 Net.Data 函数
- 使用 NetObjects Fusion (NOF) 来管理宏文件，可以更好地与 Web 站点管理相集成，并且提供了一个易于使用的图形用户界面来开发简单的宏。请参阅第139页的『附录 C. 使用 NetObjects Fusion NOF 插件和 Net.Data 小服务程序』，以学习如何使用 NOF 插件。

本节将描述以下小服务程序的主题:

- 第93页的『关于 Net.Data 小服务程序』
- 第94页的『设置小服务程序』
- 第94页的『运行 Net.Data 小服务程序』

关于 Net.Data 小服务程序

Net.Data 提供了小服务程序来帮助您开发和管理使用 Java 环境的宏。小服务程序是一些 Java 类，它们执行类似于 CGI 程序或 Web 服务器 API 插件的功能。支持 Java 小服务程序的 Web 服务器使用小服务程序来构建 HTML 页面。小服务程序没有它们自己的图形用户界面，但它们的类却可以动态地装入或通过访问网络来装入，并且可以使用一个 URL 地址(远程)或由一个类名来调用。小服务程序在 Windows NT 和 AIX 操作系统中可用。

Net.Data 小服务程序是基于 Java 的包装，它们使用本机的 DLL 文件运行 Net.Data 版本 2 宏文件或直接请求。这些小服务程序允许您运行现有的宏文件 (MacroServlet) 或单独的函数 (FunctionServlet)。要使用这些小服务程序， Net.Data 版本 2 是必需的。Net.Data 小服务程序可以运行数据库语言环境或非数据库语言环境，还可以与“现场连接”一起运行。

API 所附带的小服务程序是与应用程序一起使用的。小服务程序 API 的文档与 Net.Data 在一起。请参阅 <*inst_dir*>/servlets/NetDataServlets.jar 文件以获取 API 文档。

Net.Data 提供了两个小服务程序:

宏小服务程序 (com.ibm.netdata.servlets.MacroServlet)

执行现有的 Net.Data 宏文件。

使用宏小服务程序类似于通过 CGI-BIN 接口来运行 Net.Data 宏文件，除了您通过 Java 小服务程序运行宏文件以外。宏小服务程序要求安装 Net.Data 版本 2 或更高版本。

使用宏小服务程序的好处包括:

- SQL 查询是使用现场连接管理器通过 ODBC 运行的，以便获取增加的性能。
- 您可以通过服务器方的包含文件 (SSI) 来运行宏文件，以便在 HTML 文件中嵌入多个宏。

宏小服务程序还允许对多机种数据库(例如 DB2、Oracle 和 Sybase) 以及各种语言环境(例如 Perl、REXX 和 Java) 的本机访问。

函数小服务程序 (com.ibm.netdata.servlets.FunctionServlet)

通过小服务程序接口执行 Net.Data 函数或 SQL 语句，例如 %FUNCTION DTW_SQL()。请参阅第48页的『不使用宏文件对 Net.Data 进行调用(直接请求)』以获取更多信息。

函数小服务程序要求安装 Net.Data 版本 2。

设置小服务程序

请参阅您的 Web 服务器文档以获取有关登记和使用小服务程序的指导。Net.Data 小服务程序包含在 Net.Data <*inst_dir*>/servlets/NetDataServlets.jar 文件中。对于您的 Web 服务器来说，可能需要在 CLASSPATH 环境变量中添加 <*inst_dir*>/servlets/NetDataServlets.jar。

运行 Net.Data 小服务程序

Net.Data 小服务程序可以从 URL 运行，也可以作为 HTML 文件内的 SSI 运行。可以使用 NetObjects Fusion 插件将 Net.Data 小服务程序结合到您的 NOF 站点中。以下章节将讨论如何通过键入小服务程序的语法来修改和运行小服务程序。请参阅第139页的『附录C. 使用 NetObjects Fusion NOF 插件和 Net.Data 小服务程序』以学习如何使用 NetObjects Fusion 来修改和运行小服务程序。

运行宏小服务程序

在 HTML 文件内，使用以下某个语法选项输入小服务程序参数:

1. URL:

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

   例如:

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

2. SSI:

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

例如:

```
<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>
```

参数:

macro_value
现有 Net.Data 宏文件的全限定路径

block_value
在指定的 Net.Data 宏文件中要执行的 HTML 块的名称; 缺省为 report (可选)。

parmnn
宏文件所需要的任何附加参数, 例如

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

valuenn
宏文件所需要的任何附加值, 例如

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

HTMLPATH 参数: 如果您获取一个指示丢失了 HTMLPATH 参数的错误信息, 则向小服务程序调用命令中添加 HTMLPATH 参数:

- URL:

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

例如:

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

- SSI:

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

例如:

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

INBUFLEN 和 OUTBUFLEN 参数: 如果对宏文件的输入大于 1 KB，则必须指定 INBUFLEN 参数。如果宏文件结果大于 32 KB，则必须指定 OUTBUFLEN 参数。如果在必需的时候不能指定这些参数，则有可能导致不可预测的结果。

- URL:

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

  例如:

```
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="macro_value">
      <param name="BLOCK" value="block_value">
<param name="INBUFLEN" value="input_buffer_size">
<param name="OUTBUFLEN" value="output_buffer_size">
<param name="parmnn" value="valuenn">
 </servlet>
```

  例如:

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

运行函数小服务程序

函数小服务程序可以使用直接请求调用 Net.Data 来执行函数(例如 REXX 函数)或 SQL 语句。对小服务程序指定的参数取决于您是在执行函数还是在执行 SQL 语句。在 HTML 文件内，使用以下某个语法选项输入小服务程序参数:

1. 对于 URL:
   - ***要调用函数:***

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

     例如:

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

   - ***要调用 SQL 语句:***

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

     例如:

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

2. SSI:
   - ***要调用函数:***

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

例如:

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

- *要调用 SQL 语句:*

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

例如:

```
<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>
```

参数:

lang_env_name
要调用来处理函数、SQL 语句或存储过程的 Net.Data 语言环境(例如 DTW_SQL、DTW_REXX)。此参数要求使用 FUNC 或 SQL。

program_name
包含要执行的函数的程序名。例如 my_rexx，其中 my_rexx 是 REXX 可执行程序的名称。使用 *parmnn* 和 *valuenn* 参数来为函数指定输入参数。

database_name
与 DATABASE 参数相关联的数据库的名称。指定的

SQL_stmt_name
访问数据库的 SQL 语句或存储过程名。例如，"select * from employee"。使用 *parmnn* 和 *valuenn* 参数来为 SQL 语句或存储过程指定输入参数。

parmnn
宏文件所需要的任何附加参数，例如 <param name="field1" ...。

valuenn
宏文件所需要的任何附加值，例如 ... value="custnum"。

HTMLPATH 参数: 如果您获取一个指示丢失了 HTMLPATH 参数的错误信息，则向小服务程序调用命令中添加 HTMLPATH 参数:

- URL:

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

例如:

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

- SSI:

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

例如:

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

其中 *html_path* 指定了 Web 服务器根 HTML 目录的路径; 例如: `htmlpath=e:\html`。

INBUFLEN 和 OUTBUFLEN 参数: 如果对宏文件的输入大于 1 KB, 则必须指定 INBUFLEN 参数。如果宏文件结果大于 32 KB, 则必须指定 OUTBUFLEN 参数。如果在必需的时侯不能指定这些参数, 则有可能导致不可预测的结果。

- URL:

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

例如:

```
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="lang_env_name">
      <param name="FUNC" value="program_name">
<param name="INBUFLEN" value="input_buffer_size">
<param name="OUTBUFLEN" value="output_buffer_size">
<param name="parmnn" value="valuenn">
 </servlet>
```

例如:

```
<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 JavaBean

Net.Data 提供了 JavaBeans，它们可以在 Java 环境中使用，并且不需要在运行的 Web 服务器。JavaBean 是一个面向对象的程序设计接口，允许您构建可重用的应用程序或程序构造块。这些对象可以在那些建立在支持 Java 功能的操作系统上的网络中使用。

通过使用本机的 Net.Data DLL，JavaBean 调用 Net.Data，并填充返回码和包含 Net.Data 输出(结果)的字符串。因为 JavaBean 使用本机的 DLL，所以您就不必要让一个 Web 服务器保持运行状态以便使用 Net.Data 功能。

设计技巧: Net.Data JavaBean 返回的结果就是您的宏文件或函数所返回的；一般来说，这是 HTML。请考虑将结果传递到一个能够理解 HTML 的、类似于 HTML 的 JavaBean 并显示结果。

使用 JavaBean 之后，您可以:

- 运行 Net.Data 宏文件
- 通过 Net.Data 运行 SQL 语句

本节将描述以下 JavaBean 主题:

- 第99页的『关于 Net.Data JavaBean』
- 第99页的『设置与运行 Net.Data JavaBean』

关于 Net.Data JavaBean

Net.Data 提供了 JavaBean 来帮助您开发和管理使用 Java 环境的宏。JavaBean 是提供以下接口的 Java 对象:

- 当在 JavaBean 开发环境(例如 Lotus BeanMachine) 中使用时，您可以使用该环境所提供的定制程序将期望的组件组合在一起，从而处理和显示宏文件或 SQL 语句的结果并产生一个 Java 小应用程序。
- 使用 API 时，您可以使用 JavaBean 向您自己的 Java 小应用程序或应用程序提供 Net.Data 功能。API 文档位于 *<inst_dir>*/beans/NetDataBeans.jar 中。

Net.Data 提供了两类 JavaBean:

Net.Data 宏 JavaBean
为通过 Net.Data 执行现有的 Net.Data 宏提供了一个基于 Java 的接口。

Net.Data SQL JavaBean
为通过 Net.Data 执行 SQL 语句提供了一个基于 Java 的接口。

Net.Data JavaBean 是基于 Java 的包装，它们使用本机的 DLL 文件通过 Net.Data 运行。它们都要求安装 Net.Data 版本 2 或更高版本以及 JDK 版本 1.1 或更高版本。

设置与运行 Net.Data JavaBean

本节将描述如何使用 JavaBean 开发工具(例如 Bean Machine) 来设置和运行 Net.Data JavaBean。使用开发工具的步骤是类似的，因此您可以使用自己选择的工具。

- 第100页的『设置宏 Bean』
- 第100页的『运行宏 Bean』

- 第100页的『设置 SQL Bean』
- 第101页的『运行 SQL Bean』

设置宏 Bean

Net.Data 宏 bean com.ibm.netdata.beans.NetDataMacro 可以让您使用 Java 来运行现有的宏文件。要使用这个 bean，您需要为 bean 指定 Net.Data 特性，这样，它就可以与宏文件一起使用了。

要使用 JavaBean 开发工具设置 Net.Data JavaBean:

1. 添加或导入文件 *<inst_dir>*/beans/NetDataBeans.jar 到您的 JavaBean 开发工具中。
2. 通过使用开发工具的接口设备，设置以下输入特性:

   宏 指定要执行的现有宏文件的名称。例如: MyMacro.mac

   块 指定要执行的 HTML 块部分的名称; 缺省为 report。

   HTML 路径
   指定 Net.Data db2www.ini 文件的路径。

   参数 指定运行宏时要使用的参数名及其值。

   语法:

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

运行宏 Bean

要使用 JavaBean 开发工具运行 Net.Data JavaBean:

- 选择 JavaBean 开发工具提供的运行或执行操作来运行宏。
- 宏运行之后，您可以参考以下输出特性:

  RC 指定从 Net.Data 返回的返回码。

  结果 指定 Net.Data 宏文件执行后返回的数据。

设置 SQL Bean

Net.Data SQL bean，com.ibm.netdata.beans.NetDataSQL，可以让您使用 Java 来通过 Net.Data 运行 SQL 语句。要使用这个 bean，您需要为 bean 指定 Net.Data 特性，这样，它就可以与宏文件一起使用了。

要使用 JavaBean 开发工具设置 Net.Data JavaBean:

1. 添加或导入 NetDataBeans.jar 文件到您的 JavaBean 开发工具中。
2. 通过使用开发工具的接口设备，设置以下输入特性:

   语言环境
   指定要使用的语言环境; 缺省为 DTW_SQL。

   SQL 指定要运行的 SQL 语句; 缺省为 select * from employee。

   DATABASE
   指定要使用的数据库; 缺省为 SAMPLE。

HTML 路径

指定 Net.Data db2www.ini 文件的路径。

参数 指定运行 SQL 语句时要使用的参数名及其值。

语法:

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

运行 SQL Bean

要使用 JavaBean 开发工具运行 Net.Data JavaBean:

- 选择 JavaBean 开发工具提供的运行或执行操作来运行宏。
- SQL 语句运行之后，您可以参考以下输出特性:

  RC 指定从 Net.Data 返回的返回码。

  结果 指定从 SQL 语句返回的数据。

第9章 Net.Data 高速缓存

高速缓存有助于您提高应用程序用户的响应时间。Net.Data 将来自对 Web 服务器的请求存储在本地以备快速检索，直到刷新信息时为止。 本章描述 Net.Data 高速缓存的概念、任务和限制。

关于 Web 高速缓存

许多软件组件为 Web 应用程序执行高速缓存。这里是高速缓存应用程序的一些例子:

- Web 浏览器在内存或磁盘中本地保存 Web 页面和相关对象(例如图象和声频文件以及 Java 小程序)，当在用户重复访问相同页面时保存网络时间。
- Web 代理服务器高速缓存在本地服务器上保存 Web 页面和相关对象(近一组用户组)，以减少对远程 Web 服务器的网络访问时间，例如减少 Web 服务器检索所请求项目的时间数。 Web 代理服务器高速缓存还允许多个用户之间有效共享公共访问页面。
- Web 服务器在内存中高速缓存频繁检索的页和相关对象，以在用户重复检索相同页时节省磁盘存取时间。
- 数据库管理系统在内存中高速缓存通常保持在磁盘上的数据项，以在重复检索相同数据项时节省磁盘访问时间。

所有这些组件都各自完成它们的高速缓存过程，但是整个结果为用户提高了响应时间。为了确定何时刷新高速缓存的项目，Web 组件(浏览器、代理服务器和 Web 服务器)通常考虑以下不同选项:

- 浏览器和服务器配置选项
- 从 Web 服务器中与 Web 页面和相关项目一起返回的 HTTP 的内容，特别是满期日期信息

关于 Net.Data 高速缓存

Net.Data 本身为 Net.Data 宏生成的频繁访问页面和相关数据项提供自己的高速缓存功能。通过从 Net.Data 高速缓存中传递页面，可以节省为了创建页面而运行 Net.Data 宏和访问数据库的时间。

对于每个服务器，可以使用一个高速缓存管理器。**建议**： 为 Net.Data 的许多实例使用一个高速缓存管理器，每个高速缓存管理器对应多个高速缓存。

第104页的图 22显示 Net.Data 使用高速缓存管理器来管理来自一个宏文件的 HTML 输出的高速缓存过程。此输出可能包含来自数据库的数据。

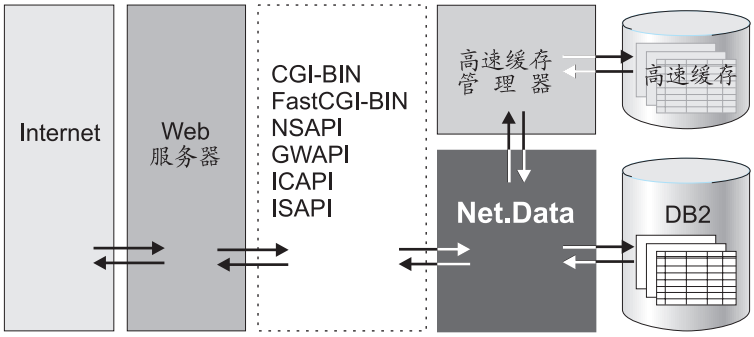


图 22. Net.Data 高速缓存

Net.Data 高速缓存术语

Net.Data 文档使用以下项目来描述 Net.Data 高速缓存。

高速缓存

一类包含最近访问过的数据的内存，是为了加快对相同数据的后继访问而设计的。高速缓存经常是用来对网络中可以访问的、频繁使用的数据保留一个本地副本。 在 Net.Data 中，指这样的本地内存: 它包含 Net.Data 所生成的 HTML Web 页面，以让 Net.Data 宏重新使用。在高速缓存中存储了页之后，Net.Data 就不必重新生成高速缓存中的信息。每个高速缓存都是由高速缓存管理器来管理的，高速缓存管理器负责管理多个高速缓存，并可以服务于 Net.Data 的多个实例。

高速缓存标识符

一个标识特定高速缓存的字符串。

高速缓存管理器

为一台机器管理高速缓存的程序。 它可以管理多个高速缓存。

高速缓存管理器配置文件

此文件包含一些设置，Net.Data 使用这些设置来确定对记录、跟踪、高速缓存大小和其它选项的设置。它包含对高速缓存管理器和特定高速缓存管理器所管理的所有高速缓存文件的设置。在 Net.Data 中封装时，文件名是 `cachemgr.cnf`。

Net.Data 高速缓存概念

根据系统上具有多少 HTTP 服务器以及每个 HTTP 服务器是否运行 Net.Data 的自身副本 (使用各自的 Net.Data 配置文件)，您可以使 Net.Data 的所有副本与一个或多个高速缓存管理器相关联。一个高速缓存管理器可以支持许多内存中的高速缓存，每个高速缓存具有一个*高速缓存标识符*。 第105页的图 23显示一个高速缓存管理器，它处理多个宏文件并管理两个高速缓存。

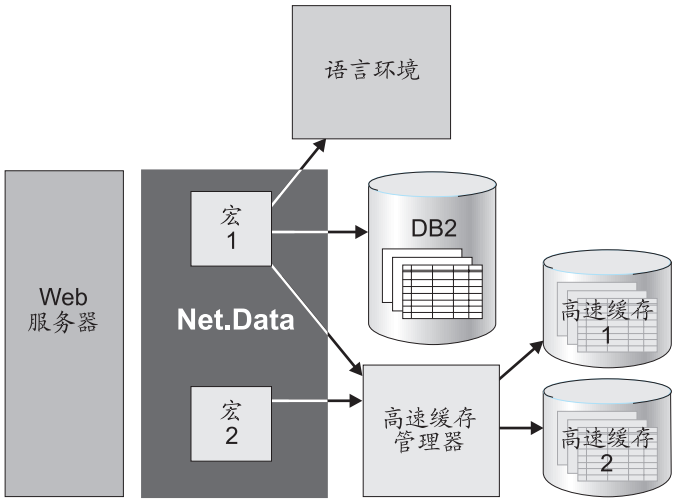


图 23. 高速缓存管理器处理多个宏文件和高速缓存

任何数目的项目(称为*高速缓存的页*都可以放在一个高速缓存中。每个高速缓存的页具有唯一的标识符，例如一个统一资源定位器 (URL)。一个页面是一个完整的 HTML 页面或它的一个分段。

当 Net.Data 接收一个高速缓存化的数据的请求(例如，来自内部函数 DTW_CACHE_PAGE)时，将发生以下步骤:

1. Net.Data 连接至高速缓存管理器。
2. Net.Data 检查数据是否已经高速缓存。
   - 如果数据已高速缓存并且未到期，Net.Data 将从高速缓存管理器请求页面，将它发送到浏览器并停止执行宏。
   - 如果数据未高速缓存，Net.Data 将继续处理宏，然后将生成的 HTML 页发送到 Web 浏览器和高速缓存管理器，在高速缓存管理器中进行高速缓存。
3. Net.Data 断开与高速缓存管理器的连接

当宏文件成功完成处理之后，高速缓存管理器高速缓存 HTML 输出，确保只高速缓存成功生成的 Web 页。数据直到发送到浏览器才被高速缓存，用户看到的数据与高速缓存的数据相同。

当 Net.Data 遇到一个错误并从宏中退出时，高速缓存管理器:

- 不接受部分或有故障的页
- 在高速缓存中保存现存页

Net.Data 高速缓存的限制

Net.Data 高速缓存具有以下限制:

安全性 高速缓存管理器不提供安全性。例如，是否一个用户运行一个宏和高速缓存数据库结果的一个页面。另一个数据库用户可以检索高速缓存的页。

直接请求

Net.Data 的直接请求调用不能使用 Net.Data 高速缓存。

Net.Data 高速缓存界面

Net.Data 提供了一个灵活的界面集，以供您在为应用程序配置和设置高速缓存时来使用。第106页的表 6描述了使用 Net.Data 高速缓存功能的各种选项以及这些选项描述的地方。

表 6. Net.Data 高速缓存界面

界面	说明	转至 ...
高速缓存管理器配置选项	您可以在高速缓存管理器配置文件的高速缓存管理器节中为高速缓存管理器指定许多选项，例如记录和跟踪。	第107页的『定义高速缓存管理器』
高速缓存配置选项	在 Net.Data 的高速缓存管理器的单个实例中，您可以定义许多高速缓存来保存高速缓存项。每个高速缓存具有自己的特性集(例如大小和位置)和高速缓存标识符。在高速缓存管理器配置文件中的高速缓存节中定义了这些特性。每个节是由高速缓存标识符来标识的。	第109页的『定义高速缓存』
Net.Data 初始化选项	如果 Net.Data 和相应的高速缓存管理器在各自系统上运行，则必须在 Net.Data 初始化文件中指定高速缓存管理器系统和端口号。	第10页的『高速缓存管理器配置变量』
Net.Data 高速缓存内部函数	可以使用 Net.Data 内部函数来处理 Net.Data 高速缓存的内容。在适当的宏函数中指定高速缓存标识符来选取具有最合适特性的高速缓存。	请参阅*Net.Data 参考*的内部函数一章

高速缓存管理器的规划

在规划 Net.Data 高速缓存功能的使用时，必须考虑:

- 高速缓存什么页有好处，以及将获取的性能改进
- 何时高速缓存项目
- 何时刷新高速缓存中的项目，以及要使用的刷新方式

要使用 Net.Data 高速缓存，必须完成以下步骤，即需要了解您希望如何使用高速缓存。

建议: 在着手于使用高速缓存的较大应用程序之前，必须先规划和设计应用程序的原型，然后才将它投入生产。

- 安装包含了高速缓存功能的 Net.Data。
- 配置高速缓存管理器。请参阅第107页的『配置高速缓存管理器和 Net.Data 高速缓存』。
- 确定您将如何把 Net.Data 应用程序投入生产。
- 检查各种 Net.Data 高速缓存日志，以确定是否应当改进高速缓存的使用和配置的方式。

高速缓存错误

当 Net.Data 遇到内部错误，使它在完成处理之前退出宏文件，则高速缓存管理器不高速缓存 Web 页。高速缓存管理器不高速缓存不完整的或包含 Net.Data 错误的页面。这些错误类型包含宏语法错误和 SQL 错误。

有错误的页在以下条件下可被高速缓存:

- Net.Data 遇到一个错误而 Net.Data 由于在消息块中有 CONINUE 指令而继续执行宏文件并正常终止。
- 错误发生在 Net.Data 的错误确定作用域之外，例如数据库滚回。

高速缓存标识符

在设计应用程序的高速缓存时，需要规划两种类型的标识符。

- **高速缓存的标识符**: 此标识符是高速缓存标识符，并在定义高速缓存的配置文件节中指定名称。可以使用许多方法来分类和命名高速缓存。例如，通过应用程序来命名高速缓存。对于每个 Net.Data 应用程序，都可以具有一个高速缓存，给予每个高速缓存一个名称，该名称推导自它服务的 Net.Data 宏。
- **高速缓存的页面的标识符**: 此标识符是高速缓存的页的标识符，并指定要高速缓存的页面的名称。高速缓存的页的标识符可以是任何字符串，例如 URl 地址。用 DTW_CACHE_PAGE() 内部函数可以指定标识符。请参阅*Net.Data 参考*的内部函数一章以获取语法和例子。

配置高速缓存管理器和 Net.Data 高速缓存

高速缓存管理器管理系统中一个或多个高速缓存。这些高速缓存中的每一个都包含动态生成的 HTML 页的内容。要配置高速缓存管理器和每个高速缓存，必须更新高速缓存管理器配置文件 `cachemgr.cnf` 中的关键字值。

高速缓存管理器配置文件中包含两种类型的节: 高速缓存管理器节和高速缓存定义节。下列步骤描述如何为应用程序定制这两个类型的节。

定义高速缓存管理器

通过指定允许的关键字的值，来定义高速缓存管理器节。所有关键字都是可选的；除非您不希望接受缺省值，否则不必指定它们。

要定义高速缓存管理器:

1. 指定高速缓存管理器日志文件的名称。日志显示所有高速缓存的所有事务的动作，并供调试和问题分析来使用。

   缺省情况是将消息写至控制台。

   语法:

   ```
   log=path
   ```

   其中，*path* 是高速缓存文件的路径与文件名称。

   提示: 要指定每个高速缓存的日志文件，可在高速缓存定义节中使用 **tran-log** 关键字。

2. 指定高速缓存管理器用于接受请求的 TCP/IP 端口号码。只对于从远程机上连接高速缓存管理器时，才使用该端口号码。

   此值必须匹配由 Net.Data 初始化文件中的 CACHE_PORT 配置变量所指定的端口号码。用以下方法确定缺省值:

   a. 高速缓存管理器在路径 */etc/services* 中检查与名称 `ibm-cachemgrd` 关联的值。如果找到此值，高速缓存管理器将使用此值。如果找不到，则使用下一种方法。

   b. 高速缓存管理器使用缺省端口 7175。

   语法:

   ```
   port=port_number
   ```

   其中 *port_number* 是唯一的 TCP/IP 端口号。

3. 以秒为单位指定一个最大时间长度，在此时间内，高速缓存管理器应当允许暂挂的读取操作保持活动。如果超过此时间，高速缓存管理器撤消连接。

   缺省值是 30 秒。

   语法:

   ```
   connection-timeout=seconds
   ```

   其中 *seconds* 是秒数，用于暂挂的读取操作应当活动的时间长度。

4. 指定是否记录消息。

   缺省值是 **no** 或 **off**。

   语法:

   ```
   logging=yes|on|no|off
   ```

   其中:

   yes|on
   指出记录是需要的。

   no|off 指出不应执行记录。

5. 指定是否环绕日志。

   缺省值是 **no**。如果指定为 **yes**，则在达到最大大小时当前记录将关闭(参见下面的 **log-size**)，文件具有文件类型 .old，新的日志被打开。只维护一代日志文件(现存的 .old 文件被覆盖)。

   语法:

   ```
   wrap-log=yes|no
   ```

其中:

yes 指定环绕日志。

no 指定不环绕日志。

6. 以字节为单位指定最大大小，如果指定了环绕记录则允许日志多至这般大小。

   缺省值是 64000。

   语法:

   ```
   log-size=bytes
   ```

   其中 *bytes* 是最大大小的字节数。

7. 指定要写入日志的消息级别。当这些值包含在 *trace_flag_definitions* 列表中时，则已被设置，不再需要进行任何设置。

   缺省值是只记录高速缓存管理器启动和关闭消息。

   语法:

   ```
   trace-flags=trace_flag_definitions
   ```

   其中:

   D_ALL
   启用所有跟踪标志。

   D_NONE
   禁用所有跟踪标志

例子: 启用指定所有跟踪标志的跟踪标志:

```
trace=flags=D_ALL
```

节的例子: 一个有效的配置管理器节:

```
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
}
```

定义高速缓存

通过指定允许的关键字的值，来定义高速缓存定义节。 大部分关键字都是可选的，除非您不希望使用缺省值，否则不必指定它。

要定义一个高速缓存:

1. 指定要保持高速缓存页的路径和目录名。就启动来说，包含此目录的文件系统必须至少象 **fssize** (参见下面) 的值一样大；否则高速缓存将不能启动。可将此值指定为一个绝对路径名或对应于高速缓存管理器启动的路径的相对路径名。

   必需的。

   语法:

   ```
   root=path_name
   ```

其中:

path_name
是高速缓存页所存储的绝对或相对的目录和路径名。

2. 指定在高速缓存管理器启动时是否激活当前高速缓存。

   不是必需的; 缺省值是 **yes**。如果设置为 **no**, 则高速缓存被定义在高速缓存管理器中但是不激活。以后您可以用 **cacheadm** 命令来激活它。

   语法:

   ```
   caching=yes|no
   ```

   其中:

   yes 指出当高速缓存管理器启动时将激活高速缓存。

   no 指出当高速缓存管理器启动时将不激活高速缓存。

3. 指定由当前高速缓存中的页在文件系统中所使用的最大空间。超过最大空间数时, 高速缓存管理器从最旧的页开始删除足够的页, 以将高速缓存所占有的总空间限制在一个极限之内。您可以通过将该值设置为一个很大的数目来有效禁用对条目的自动清除; 但是如果超过物理文件系统空间, 则试图将新的页面添加到高速缓存将会失败。

   不是必需的; 缺省值是 0 (没有高速缓存至磁盘)。

   语法:

   ```
   fssize=nnB|nnKB|nnM
   ```

   其中:

   *nn***B** 是字节数; 例如 5000B。

   *nn***KB** 是千字节数; 例如 640KB。

   *nn***MB** 是兆字节数; 例如 30MB。

4. 指定由该高速缓存中的所有页面使用的最大内存数。超过最大内存数时, 高速缓存管理器从最旧的页开始删除足够的页, 以将高速缓存所占有的总内存限制在一个范围之内。 您可以通过将该值设置为一个很大的数目来有效禁用对页面的自动清除; 但是如果 **cachemgrd** 过程消耗太多内存, 操作系统可能会终止它。

   不是必需的; 缺省值是 1MB。

   语法:

   ```
   mem-size=nnB|nnKB|nnMB
   ```

   其中:

   *nn***B** 是字节数; 例如 5000B。

   *nn***KB** 是千字节数; 例如 640KB。

   *nn***MB** 是兆字节数; 例如 30MB。

5. 指定了一个页面可在高速缓存中保持的最大时间长度。超过此值时, 高速缓存管理器将页面标记为期满, 但是除非到达 **fssize** (如果高速缓存在磁盘上) 或 **memsize** (如果高速缓存在内存中) 极限, 否则不删除该页面。如果到达 **memsize** 或 **fssize** 极限, 则高速缓存管理器在所有其它页面之前删除标记为期满的页。可以使用 **check_expiration** 关键字来禁用**lifetime**检查。

   必需的: 否; 缺省值是 5 分钟。

语法:

```
lifetime=time_length
```

其中:

*nn***S** 秒数; 例如 600S。

*nn***M** 分钟数; 例如 20M。

*nn***H** 小时数; 例如 30H。

6. 指定是否将高速缓存页标记为期满并执行寿命检查。

   不是必需的; 缺省值是 **yes**, 缺省寿命长度是 60 秒。还可将此值设置为一个时间长度, 以指出 **yes** 值并说明一个项目可在高速缓存中保持的最大时间长度。设置为 **no** 时, 高速缓存页不标记为期满, 并且不执行寿命检查。

   语法:

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

   其中:

   yes 指出高速缓存管理器执行寿命检查, 并且高速缓存页标记为期满。

   *nn***S** 是秒数; 例如 600S。

   *nn***M** 是分钟数; 例如 20M。

   *nn***H** 是小时数; 例如 30H。

   no 指出高速缓存管理器不执行寿命检查, 并且高速缓存页不标记为期满。

7. 指定一个高速缓存的页面可在内存高速缓存中占用的最大空间量。如果对于内存来说页面太大, 则选择文件高速缓存。如果存在足够的空间, 高速缓存管理器将在文件高速缓存中擦除高速缓存页面。如果页面不匹配文件高速缓存, 则高速缓存的试图将失败。如果页面小于 datum_memory_limit 值 (cacheobj-memory-limit), 但是如果高速缓存没有足够的空间, 则将从内存高速缓存中删除最早的高速缓存页面以容纳新的页面。

   不是必需的; 缺省值是 1KB。

   语法:

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

   其中:

   *nn***B** 是字节数; 例如 5000B。

   *nn***KB** 是千字节数; 例如 640KB。

   *nn***MB** 是兆字节数; 例如 30MB。

8. 指定一个高速缓存页面可在文件高速缓存中占用的最大空间量。 如果一个页面小于 datum_disk_limit, 但是在文件高速缓存中没有余留空间, 则将从文件高速缓存中删除最早的高速缓存页面以容纳新的页面。

   不是必需的; 缺省值是 1KB。

   语法:

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

   其中:

*nn***B** 是字节数；例如 5000B。

*nn***KB** 是千字节数；例如 640KB。

*nn***MB** 是兆字节数；例如 30MB。

9. 指定了创建统计记录的时间。如果设置为 0，则不写统计记录。

   不是必需的；缺省值是 0 (不统计)。

   语法:

   ```
   stat-interval = nnS|nnM|nnH
   ```

   其中:

   *nn***S** 是秒数；例如 600S。

   *nn***M** 是分钟数；例如 1M。

   *nn***H** 是小时数；例如 3H。

10. 指定一个文件的路径和名称，该文件用于记录当前高速缓存的统计值。

    当 **stat-interval** 的值大于 0 时是必需的。

    语法:

    ```
    stat-files=filename
    ```

    其中 *filename* 是记录统计文件的路径和名称。

11. 指定每次写日志文件时，统计计数器是否应当复位。

    不是必需的；缺省值是 **yes**。

    语法:

    ```
    reset-stat-counters=yes|no
    ```

    其中:

    yes 复位统计计数器。

    no 不复位统计计数器。

12. 定义存放每个高速缓存的事务日志的路径与文件名称。高速缓存事务日志文件独立与高速缓存管理器日志文件，后者用于记录整个高速缓存管理器的活动。

    必需的；如果不指定，就不创建高速缓存的事务日志。

    语法:

    ```
    tran-log=filename
    ```

    其中 *filename* 是每个高速缓存的事务日志的路径和文件名。

13. 指定是否在高速缓存管理器首次启动时打开高速缓存的事务日志。除非已通过 tran-log 参数指定了一个有效事务日志文件，否则忽略此参数。如果在高速缓存管理器配置文件中已经指定 tran-log 值，则当高速缓存管理器守护程序正在运行时，可使用 **cacheadm** 命令来激活事务日志。

    不是必需的；缺省值是 **no**。

    语法:

    ```
    tran-logging=yes|on|no|off
    ```

    其中:

yes|on
指出记录是需要的

no|off 指出不应执行记录。

14. 指定是否应当环绕事务登记。

    不是必需的；缺省值是 **yes**。 如果指定为 **yes**，则在达到最大大小时当前日志将关闭(参见下面的 **tran-log-size**)，具有文件类型 .old，新的日志被打开。 只维护一代日志(现存的 .old 文件被覆盖)。

    语法:

    ```
    wrap-tran-log=yes|no
    ```

    其中:

    yes 指定环绕日志。

    no 指出不环绕日志。

15. 以字节为单位指定最大大小，如果指定了 **wrap-tran-log** 则允许事物日志多至这般大小。

    不是必需的；缺省值是 64000。

    语法:

    ```
    tran-log-size=bytes
    ```

    其中 *bytes* 是最大大小的字节数。

节的例子: 高速缓存的一个有效的高速缓存定义节:

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

启动和停止高速缓存管理器

以下章节描述了如何启动和停止高速缓存管理器。

- 第114页的『启动高速缓存管理器』
- 第114页的『停止高速缓存管理器』

启动高速缓存管理器

使用 cachemgrd 命令来启动高速缓存管理器守护程序。

语法:

参数:

cachemgrd
命令关键字。

config_file
指定一个文件名称，在该文件中定义高速缓存管理器以及高速缓存管理器所管理的每个高速缓存。Net.Data 装运的配置文件是 cachemgr.cnf。

例子:

```
cachemgrd -c myconfig.cfg
```

停止高速缓存管理器

使用 cacheadm 来停止高速缓存管理器。

语法:

参数:

cacheadm
命令关键字。

hostname
指定高速缓存所运行的机器的名称，如果该机器不同于发出 cacheadm 命令的机器的话。

port_num
指定高速缓存端口号码，如果该号码不同于缺省值 (7175) 的话。

terminate
指定要停止高速缓存管理器。

例子:

```
cacheadm hostname host1 port 7178 terminate
```

高速缓存 Web 页

利用使用 DTW_CACHE_PAGE 内部函数来高速缓存一个 Web 页。当 Net.Data 在宏文件中发现 DTW_CACHE_PAGE 函数时，它与高速缓存管理器联系并开始在内存中保存宏文件的 HTML 输出。在 Net.Data 成功地处理了一个宏之后，HTML 输出就发送给浏览器，并且高速缓存管理器如第115页的图 24中所示地在一个事物中将输出进行高速缓存。

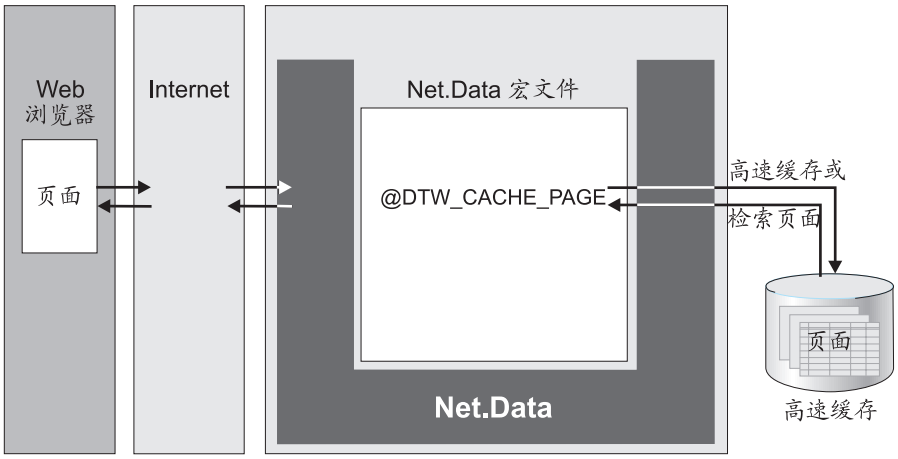


图 24. DTW_CACHE_PAGE 函数初启高速缓存

高速缓存一个页面

使用 Net.Data DTW_CACHE_PAGE() 内部函数来指定要写至高速缓存的 Net.Data 生成的页面。

一旦确定在高速缓存中已经不存在页面或页面已经过期，DTW_CACHE_PAGE() 函数将高速缓存函数语句之后的宏文件的所有输出。如果页面不存在于高速缓存中或超过指定的年龄，Net.Data 将把输出页面发送回浏览器，从宏执行中生成新的输出页面，并将页面存储在高速缓存中。

如果高速缓存管理器找到高速缓存页面并且该页面仍然是当前的，则它显示高速缓存的内容，并且 Net.Data 退出宏。此行为保证了在从高速缓存中检索了 Web 页面之后，不再作不需要的处理。

性能提示: 把 DTW_CACHE_PAGE() 放在最先，或作为宏文件中的第一条语句，以将执行宏文件的代价降到最低。

要高速缓存一个页面:

1. 在宏文件的 HTML 块中，在 HTML 编码之前，插入以下函数语句:

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

   使用该函数来指出 Net.Data 将对跟随此语句之后的宏中的所有 HTML 输出进行高速缓存。如果您希望高速缓存所有 HTML 输出，则将该语句放在宏文件的最前面。

   参数:

   cache_id
   标志放置此页的高速缓存的字符串。您可以将高速缓存标识符与宏或宏组相关联。

   cached_page_id
   一个包含了一个标识符的字符串，该标识符用于标识后继 @DTW_CACHE_PAGE 高速缓存请求中的高速缓存中的页，例如页面的 URL。

   age 包含时间长度(以秒计)的字符串变量，是在认为页面过期时指定的。 如果请

求的页在高速缓存中停留的时间长于值 *age*，Net.Data 将执行页面，并高速缓存所生成的页面，代替过期页面。如果请求的页在高速缓存中停留的时间等于或小于 age 的值，Net.Data 将在高速缓存中检索页面并将它发送到浏览器。在此情况下，Net.Data 立即结束宏执行。

status 由 Net.Data 返回的一个字符串变量，它指出页面是否已经正确高速缓存。

例子:

```
%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>
 %}
```

高级高速缓存: 动态确定是否高速缓存

DTW_CACHE_PAGE() 从宏文件中的位置初启高速缓存。您通常将函数放在宏文件的开头，以获取最佳性能并确保所有 HTML 都已高速缓存。

对于高级的高速缓存应用程序，当您需要在处理期间决定在某个特定点作高速缓存时，可以把 DTW_CACHE_PAGE() 函数放在 HTML 输出段，而不是在宏文件的开始。例如，您可能需要根据从查询或函数调用返回的行数来作高速缓存决定。

例子: 因为作高速缓存的决定依赖于 HTML 输出的期望尺寸，所以把函数放在 HTML 块中

```
 % 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>
%}
```

在此例中，此页根据 HTML 输出的期望尺寸作高速缓存或检索。只有当数据库表格包含多于 100 行时，才认为 HTML 输出页是值得作高速缓存的。Net.Data 总是在执行这个宏之后，把 OUTPUT 块中的文本(This is the customer list)发送给浏览器；此文本永不作高速缓存。跟在函数调用后的行 (@count_rows()) 在满足 IF 块的条件时作高速缓存或检索。 两部分共同形成一个完整的 Net.Data 输出页。

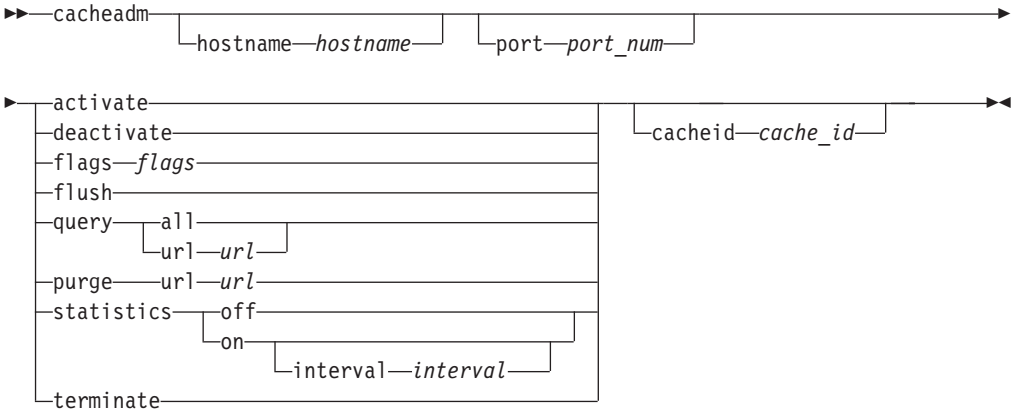
CACHEADM 命令

对于以下任务，使用 CACHEADM 命令:

- 停止高速缓存管理器
- 闪断特定的高速缓存
- 查询特定的高速缓存
- 启用或禁用记录
- 记录标志
- 启动和停止统计值的收集

所有参数都可以缩写为最小唯一字符集。

语法:



参数:

activate

激活指定的高速缓存。如果高速缓存已是活动的，高速缓存管理器就不做任何事。

cache_id

一个字符串变量，识别页面所位于的高速缓存。例如: cache1。

deactivate
释放指定的高速缓存。如果高速缓存已是不活动的，高速缓存管理器就不做任何事。 完成所有暂挂操作并且不接受新的。最后一个操作完成时，高速缓存管理器将高速缓存标记为不活动。

flags
指定应当打开还是关闭列出的标志。

D_ALL
打开所有跟踪标志。

D_NONE
关闭所有跟踪标志。

flush
闪断一个由 *cache_id* 参数指定的高速缓存，它是该参数必需的。此参数从指定的高速缓存中无条件删除所有项目。

hostname
指定高速缓存所运行的机器的名称，如果该机器不同于发出 `cacheadm` 命令的机器的话。例如: `myhost`。

port_num
指定高速缓存端口号码，如果该号码不同于缺省值 (7175) 的话。 在系统中，此数目必须是唯一的。

purge
指定要从高速缓存中清除的特定页。如果已指定 *url*，高速缓存管理器将用匹配 *url* 的关键字来清除页面。如果已指定从属关系，高速缓存管理器将清除具有关联从属关系的所有项目，并将其关键字写入标准输出流 `stdout`。

query
根据指定的参数来返回高速缓存数据:

- 如果只指定高速缓存标识符，则返回关于此高速缓存的信息。
- 如果指定了 *url*，则返回关于特定的高速缓存的页的信息。
- 如果指定了 `all`，则返回关于所有页的信息。

  其它程序使用 `all` 选项来格式化或解释结果。每行包含以下信息:

  – 页面关键字
  – 页面年龄
  – 页面长度
  – 页面建立日期
  – 页面到期日期
  – 页面最后引用的日期

  所有日期都使用标准的 UNIX 整数时间格式。

  性能提示: 选项高速缓存查询全部可能会影响性能，应小心使用。

statistics
启用或禁用对为特定高速缓存收集的统计值的记录，并需要 *cache_id* 参数。如果当 `statistics` 参数设置为 `on` 时指定了一个间隔，则 Net.Data 将更新间隔设置或复位为特定的秒数。

terminate
指定要停止高速缓存管理器。

tranlogging
启用或禁用特定高速缓存的事务处理登记，并需要 *cache_id* 参数。 仅当用 tran-log 参数在高速缓存管理器配置文件中指定了一个有效的高速缓存事物日志之后，此参数才起作用。

url 全局相关位置 (URL) 地址指定 Web 服务器上文件的位置。例如 http:www.ibm.com/mydir/page1。

高速缓存日志

根据内部操作，可保持几种类型的统计值，并任选地写入高速缓存日志。可以为每个高速缓存各自维护一个独立的日志，也可以将所有统计值写入相同日志。本章讨论以下高速缓存日志主题:

- 第119页的『配置日志』
- 第119页的『高速缓存日志的格式』

配置日志

要记录统计值，必须配置高速缓存管理器配置文件。

要配置日志:

在高速缓存管理器配置文件中的高速缓存节中指定 stat-files 和 stat-interval 关键字。

您可以修改统计值设置，而不必停止、重新配置和重新启动高速缓存管理器。

要修改统计值收集设置:

指定 cacheadm statistics 命令。但是请注意，在重新启动高速缓存管理器时，用 cacheadm statistics 命令作的更改不保存。

高速缓存日志的格式

统计日志是一个普通 ASCII 文件，可以通过电子表格或数据库程序来处理或调入它。可写入三种类型记录:

- 初始化记录记录了为特定高速缓存收集的统计值的启动。这些记录具有以下格式:

  mm/dd/yy hh:mm:ss id Initialization: interval *n* seconds

  其中:

mm/dd/yy	是收集的统计值启动时的月、日、年
hh:mm:ss	是收集的统计值启动时的时、分、秒
id	是与记录关联的高速缓存的名称
n	是集合间隔

- 终止记录记录了为特定高速缓存收集的统计值的终止。 这些记录具有以下格式:

mm/dd/yy hh:mm:ss id Termination

其中:

mm/dd/yy 是收集的统计值停止时的月、日、年

hh:mm:ss 是收集的统计值停止时的时、分、秒

id 是与记录关联的高速缓存的名称

- 统计记录是以空格定界的数目集，显示高速缓存中的活动。这些记录具有以下格式:

  mm/dd/yy hh:mm:ss id statistics

  其中:

  mm/dd/yy 是收集的统计值创建时的月、日、年

  hh:mm:ss 是收集的统计值创建时的时、分、秒

  id 是与记录关联的高速缓存的名称

  <*statistics*> 如 第120页的表 7 所示，是一个以空格定界的统计值列表，为该高速缓存而收集:

表 7. 统计值列表

字段号	内容	说明	计数器重新初始化为零
1	读取	基于高速缓存执行的读出操作的数目	是
2	写入	基于高速缓存执行的写操作的数目	是
3	关闭	在高速缓存中的对象上执行的关闭操作的数目	是
4	打开读取	在高速缓存中的对象上执行的打开读取操作的数目	是
5	打开写入	在高速缓存中的对象上执行的打开写入操作的数目	是
6	打开写入查询	在高速缓存中的对象上执行的打开写入查询操作的数目	是
7	读取命中	在高速缓存的对象上的读取命中的数目	是
8	写入命中	在高速缓存的对象上的写入命中的数目	是
9	写入查询命中	在高速缓存的对象上的写入查询命中的数目	是
10	初始化	与该高速缓存确定的新会话的数目	是
11	终止	与该高速缓存终止的会话的数目	是
12	清除	从该高速缓存中删除的对象的数目	否
13	使用的内存	此高速缓存的内存部分中的对象所使用的内存数	否
14	使用的磁盘	此高速缓存的磁盘部分中的对象所使用的磁盘空间数	否
15	可用内存	供此高速缓存的内存部分中的对象所使用的有效内存数	否
16	可用磁盘	供此高速缓存的磁盘部分中的对象所使用的有效磁盘空间数	否
17	内存对象数目	此高速缓存的内存部分中的对象数目	否
18	文件对象数目	此高速缓存的磁盘部分中的对象数目	否
19	会话数目	基于高速缓存的当前活动的会话数目	否

第10章 改进性能

改进性能是调整系统时的一个重要部分。本章将讨论改进性能的策略。将讨论以下主题:

使用Web 服务器 API 改进性能

您可以用 Web 服务器 API 取代 CGI 来调用 Net.Data, 从而改进性能。当 Net.Data 在 Web 服务器 API 方式中运行时, Net.Data 将在 Web 服务器的进程内部作为线程执行, 消除了将 Net.Data 作为 CGI 进程调用造成的系统开销。使用了 Web 服务器 API, Net.Data 将在服务器的进程内部作为多个线程运行。

缺省情况下, Web 服务器将 Net.Data 作为 CGI 程序调用, 每个 Net.Data 进程在单个进程中运行。为了改进性能, Net.Data 为 Web 服务器 API 提供了配置选项。

Net.Data 支持以下列表中的 Web API, 这取决于您的操作系统:

GWAPI 插件和 ICAPI 插件
: Lotus Domino Go Webserver API 插件作为 IBM Internet Connection Secure Sever 插件的改进型

ISAPI 插件
: Microsoft Internet Server API 插件

NSAPI 插件
: Netscape Server API 插件

请参阅 *Net.Data 参考*中的操作系统参考附录, 以确定对于您的操作系统支持哪个 Web 服务器 API。请参阅第26页的『配置 Net.Data 以便与 Web 服务器 API 一起使用』, 以学习如何配置 Net.Data 和 Web 服务器以使它们与 API 一起使用。

考虑: 使用 Web 服务器 API 提供了改进的性能, 而没有隔离应用程序。因为 Net.Data 以多线程的方式运行, 因此用户开发的语言环境中的错误、不适当的调用、甚至数据库的停机都可能引起 Web 服务器的问题, 并存在使服务器关闭的潜在可能。在确定是否使用某个 Web 服务器 API 来取代 CGI 或 FastCGI 时, 需要确定对于您的应用程序来说, 性能和应用程序隔离中哪一个的优先级较高。

要求:

- 如果在 GWAPI、ICAPI、ISAPI 或 NSAPI 方式中运行 Net.Data, 您必须重新启动 Web 服务器, 这样 Web 服务器就可以重新装入 Net.Data 并将它作为进程运行。

- 如果您在 Web 服务器以 API 方式调用 Net.Data 之后对初始化文件进行更改，那么您必须重新启动 Web 服务器。否则，对于 Net.Data 初始化文件 (db2www.ini) 的任何更改都没有作用。在 API 方式中，Net.Data 只读取初始化文件一次，从而减少对性能的系统开销。
- 在 API 方式中运行时，Oracle 和 Sybase 语言环境需要“现场连接”。

要调用 Web 服务器 API:

对于 ICAPI 和 GWAPI:

语法:

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

参数:

server_name
服务器的名称。

macro_name
宏文件在 MACRO_PATH 指定的目录下的相对路径名。

html_block
要处理的宏文件中的 HTML 块的名称。

例子:

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

对于 ISAPI:

语法:

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

参数:

server_name
服务器的名称。

server_HTML_root_directory
Web 服务器 HTML 根目录的名称。

dll_name
Net.Data 的 ISAPI .dll 文件名，dtwisapi.dll。

macro_name
宏文件在 MACRO_PATH 指定的目录下的相对路径名。

html_block
要处理的宏文件中的 HTML 块的名称。

例子:

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

对于 NSAPI:

语法:

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

参数:

server_name
服务器的名称。

macro_name
宏文件在 MACRO_PATH 指定的目录下的相对路径名。宏文件的扩展名，例如 `.d2w`，必须在 Web 服务器配置文件中定义。请参阅第26页的『配置 Net.Data 以便与 Web 服务器 API 一起使用』，以获取更多信息。

html_block
要处理的宏文件中的 HTML 块的名称。

例子:

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

使用 FastCGI 来改进性能

FastCGI 提供了改进的性能和 CGI-BIN 的可靠性。使用 FastCGI 允许您以 API 服务器的速度、更可靠的方法(使用分开的内存空间)来运行宏。 Net.Data 调用缺省情况是与 CGI 一起运行。

您可以在所有支持 FastCGI 的服务器上同时使用 Net.Data 和 FastCGI。

请参阅第22页的『为 FastCGI 配置 Net.Data』，学习如何配置 FastCGI。

您可以调节 FastCGI 来运行适当数量的进程，从而处理与进程配置参数一起进入的请求数。例如，一个用户平均每秒有 100 个请求，每个请求需要半秒钟来进行处理。则将进程参数设置为 50。

FastCGI 受到以下语言环境的支持:

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

要求: 在 FastCGI 方式中运行时，Oracle 和 Sybase 语言环境需要 “现场连接”。

要调节同步进程的个数:

1. 打开定义进程配置参数的配置文件。
   - 对于 Apache，这是 `httpd.conf` 文件。
   - 对于 ICS 或 Lotus Domino Go Webserver，这是 `lgw_fcgi.conf` 文件。
2. 更改指定进程数的配置参数值:
   - 对于 Apache: `Process=`*`num`*。
   - 对于 ISC: `NumProcess=`*`num`*。

其中 *num* 是进程个数。

使用“连接管理”改进性能

Net.Data 提供了一个称为“现场连接”的组件来管理数据库和 Java 虚拟机连接。“现场连接”维护持续的连接，以改进性能。有些 Net.Data 的操作需要很长的启动时间。例如，一个进程在发出数据库查询之前，必须先向 DBMS 标识它自己并连接到数据库。这通常在访问数据库的 Net.Data 宏所需的进程时间中占很大的部分。另一个在启动时需要很多开销的例子是运行 Java 应用程序(非 Java 小应用程序)所需的 Java 虚拟机。由于 CGI 程序操作的方式，在每次请求 Web 服务器时都需要这些启动时的花费。Net.Data 在 OS/2、Windows NT 和 UNIX 操作系统上提供了“现场连接”并维护持续连接。

以下章节描述了“现场连接”。

- 第124页的『关于现场连接』
- 第125页的『现场连接的优点』
- 第125页的『我应当使用现场连接吗?』
- 第126页的『启动连接管理器』
- 第126页的『Net.Data 和现场连接进程流』

关于现场连接

“现场连接”可以通过消除启动时的系统开销来动态地改进性能。这些节省来自于连续运行一个或多个执行启动功能的进程。然后，这些进程将等待服务请求。如果您将 Net.Data 用作 CGI 或 FastCGI 程序，或者用作 Web 服务器 API 插件，那么您可以运行“现场连接”。

“现场连接”由连接管理器和 cliette 组成。*Cliette* 是连接管理器所启动的一些进程，它们在服务器运行期间保持活动状态。Cliette 处理数据并与您在初始化文件中用 CLIETTE 关键字指定的 Net.Data 语言环境通信。每个类型的 cliette 处理一个专门的语言环境函数，例如 DB2 cliette 连接到 DB2 数据库并建立在 Net.Data 调用任何 Net.Data 宏之前执行 SQL 调用的操作。这个可执行文件是在现场连接配置文件中命名的，称为 `dtwcm.cnf`。第125页的图 25显示了“现场连接”、宏文件以及语言环境之间的交互。

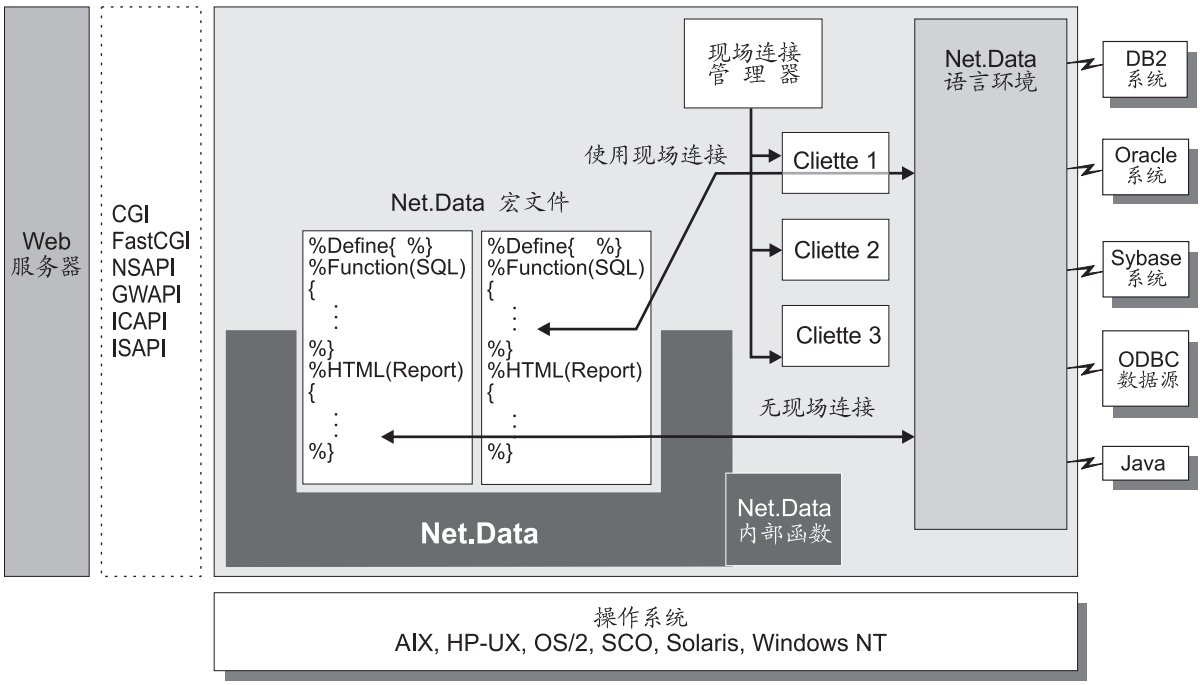


图 25. 使用 Cliette 的现场连接

以下章节更详细地描述了“现场连接”。要学习如何配置“现场连接”，请参阅第18页的『配置现场连接』。

现场连接的优点

使用“现场连接”的主要优点是:

- **改进性能**

  重新使用连接要比建立新的连接有效得多。通常，如果您请求小的 SQL 语句(例如，对少于 100 000 行的数据库的简单查询)，或者如果您的数据库连接很困难(例如，远程服务器)，那么连接的时间是占很大比重。

- **多数据库访问**

  “现场连接”允许一个 Net.Data 宏同时连接到多个数据库。这一点是可能的，因为每个数据库都有唯一的 cliette，因此 Net.Data 只需和多个 cliette 通信即可。

我应当使用现场连接吗?

您可以以 CGI、FastCGI 或 API 方式使用“现场连接”来与您的数据库通信。另外，如果应用程序需要来自多个数据库的数据，那您还可以从“现场连接”受益。

现场连接和 API 插件一起使用，可以改进许多系统的性能(取决于系统的负载和配置)。您应当在自己的系统上试验一下，确定哪一个配置是最佳的。

通过使用 ACTIVATE DATABASE 或 START DATABASE 命令来节省建立数据库连接的时间，许多应用程序可以改进性能而不必使用“现场连接”。请参阅数据库的文档，以获取有关您的数据库所使用的命令的细节。还请检查一下您的操作系统文档，是否有其它能够帮助改进性能的步骤。

要求: 在 CGI、FastCGI 和 API方式中运行时，Oracle 和 Sybase 语言环境需要“现场连接”。

启动连接管理器

连接管理器是与 Net.Data 一起发行的一个独立的可执行文件，名为 dtwcm。在启动 Web 服务器时启动连接管理器。

启动连接管理器时，它将读取配置文件并启动一组进程。在每个进程中，连接管理器都将开始一个特定 cliette 的执行。要学习如何配置“现场连接”，请参阅第18页的『配置现场连接』。

要在 Windows NT 和 OS/2 中启动连接管理器:

1. 从命令行更改到 <*inst_dir*>\connect\ 目录。
2. 输入 dtwcm。

其中 <*inst_dir*> 是 Net.Data 的安装目录。

要在 AIX 中启动连接管理器:

1. 从命令行更改到 /usr/lpp/internet/db2www/db2/ 目录。
2. 输入 dtwcm。

要带消息选项启动连接管理器:

缺省情况下，连接管理器的消息是被关闭的。如果您希望显示连接管理器的消息，可以在启动连接管理器时使用 -d 选项。

从命令行输入: dtwcm -d

使用 -d 选项之后，要想再次关闭消息，就必须重新启动连接管理器。

要自动将连接管理器作为 Windows NT 的服务启动:

在 Windows NT 上，您可以指定将连接管理器作为 Windows NT 的服务启动，而不是从命令行启动。将连接管理器作为 Windows NT 的服务运行可以使连接管理器在每次启动机器时自动启动。

技巧: 在将连接管理器设置为自动启动之前，先从命令行启动，以确保现场连接配置文件是正确的。

1. 从 Windows NT 任务栏，选择**开始->设置->控制面板->服务**。
2. 选择 **Net.Data 连接管理器**，然后单击**启动**按钮。
3. 选择**自动启动类型**，然后单击**确定**。

Net.Data 和现场连接进程流

在配置并启动了数据库、Web 服务器以及连接管理器之后，Net.Data 处理一般将在启用“现场连接”时调用这些步骤:

1. Web 服务器接收请求并启动一个 FastCGI、CGI 或 API 进程来运行 Net.Data。
2. Net.Data 开始处理 Net.Data 宏。
3. 当 Net.Data 遇到使用“现场连接”的函数调用时，它将确定需要初始化文件中哪些类型的 cliette。对于 DB2，cliette 类型通常是根据 DB2 数据库名称所取的一个名称，例如 DTW_SQL:CELDIAL。

4. Net.Data 向连接管理器要求该类型的 cliette。
5. 连接管理器查找可用的该类 cliette。如果没有可用的 cliette，连接管理器将把请求放入队列，当有正确的 cliette 类型可用时再进行处理。
6. 当有一个 cliette 可用时，连接管理器将告诉 Net.Data 如何与该 cliette 通信。
7. Net.Data 让 cliette 处理函数。
8. 处理从步骤 3 开始重复，直至 Net.Data 宏处理完成为止。
9. 释放所有的 cliette。

如果初始化文件中指定了一个 cliette，但连接管理器没有运行，Net.Data 将装入 DLL 并处理该宏。如果使用 API，那么很可能会收到错误信息，这时就应启动连接管理器。

使用 Net.Data 高速缓存来改进性能

您可以通过频繁地高速缓存请求的 Net.Data Web 页面来改进系统性能。从高速缓存中检索页面可以降低使用的处理器时间，因为 Net.Data 宏既不是被解释的，也不是由任何 Net.Data 语言环境调用执行的。如果 Net.Data 页面是从数据库或文件持有的信息中生成的，则很可能降低磁盘 IO 访问。节省的结果是，您可能还会发现吞吐量与响应时间也可以降低。

Net.Data 高速缓存在一个单独的进程内提供了多高速缓存的多线程管理。在多台机器上的多个进程共享单个高速缓存的网络环境中，高速缓存可以很好地工作。 Net.Data 高速缓存使用高速缓存管理器组件来管理 Net.Data 高速缓存。

根据您所定义的高速缓存和现有系统中内存的大小，您可能还需要一些内存才能避免额外的页面调度。

请参阅第103页的『第9章 Net.Data 高速缓存』，以学习如何设置和使用 Net.Data 高速缓存。

Net.Data 错误记录: 性能考虑

Net.Data 提供了一个错误记录，这样您就可以监视 Net.Data 系统的错误或性能问题。

在使用 Net.Data 错误记录时，您可能会注意到如果有许多消息要写入错误记录，则会对系统的性能产生影响。例如，每当用户访问 Net.Data 找不到的宏时，Net.Data 把消息作为输出传送到错误记录中。

为了减少对性能的影响，请使用 DTW_LOG_LEVEL 关键字检查 Net.Data 宏文件中设置的错误记录的记录等级。如果这个等级设置为 WARNING，可以考虑将等级降低为 ERROR 来获取较小的性能改进，或者将其设置为 OFF 来获取较大的性能改进。

优化数学函数

可以使用 DTW_OPTIMIZE_MATH 配置变量来改进数学函数的性能。此变量提供了 C 语言风格格式化的数学函数，从而加快了对这些函数的处理。当设置 YES 时，Net.Data 使用 C 风格的格式。NO 是缺省值。请参阅第12页的『DTW_OPTIMIZE_MATH: 优化数学函数变量』，以学习如何配置此变量。

语法:

```
DTW_OPTIMIZE_MATH=NO|YES
```

考虑: Net.Data 版本 1 使用 REXX 风格对数学函数进行格式化。使用 DTW_OPTIMIZE_MATH 配置变量时，您的输出可能不同。尾部的空格不显示。如果与 Net.Data 版本 1 宏文件一致的输出对于您的应用程序很重要，则应考虑一致性与性能的价值。

第11章 记录 Net.Data 错误信息

Net.Data 将错误信息写入到 Net.Data 错误记录文件 `netdata.log` 中。错误记录的最大尺寸被 Net.Data 固定为 500 KB，大约 3000 个记录项。

您可以浏览错误记录文件或归档副本，定期地确定 Net.Data 系统中是否存在问题。

本章将讨论以下记录主题:

计划监控记录

在记录错误时，需要计划以下问题:

- 确定磁盘空间:

  如果您计划使用错误记录，则必须允许对错误记录使用额外的磁盘空间。

- 配置 Net.Data:

  如果您计划控制整个 Net.Data 系统的错误记录，则应在 Net.Data 初始化文件中设置一个配置变量: DTW_LOG_DIR

  这个变量是错误记录必需的，即使已经在宏中将 DTW_LOG_LEVEL 变量设置为 error 或 warning。请参阅第11页的『DTW_LOG_DIR: 错误记录位置变量』，以学习如何更新初始化文件。

- 编写 Net.Data 宏:

  使用宏文件中的 DTW_LOG_LEVEL 关键字来指定记录的等级。

- 运行 Net.Data:

  如果您在使用错误记录，那么您可以检查 Net.Data 系统中错误的错误记录和档案文件。

- 调节:

  请注意，记录将会影响性能。请参阅第127页的『Net.Data 错误记录: 性能考虑』，以获取有关性能问题的信息。

控制记录文件中的记录数

您可以使用 DTW_LOG_LEVEL 关键字来指定记录的等级。在 Net.Data 宏文件中定义这个关键字。它有三个设置:

off Net.Data 不记录错误。这是缺省。

error Net.Data 记录错误信息。

warning
Net.Data 记录警告和错误信息。

不记录的消息类型

Net.Data 不记录由宏文件中的 MESSAGE 部分显式处理的错误。

记录文件的尺寸和循环

记录文件的最大尺寸可达 500 KB。在这种情况下，大约可以装入 3000 个记录项。

当记录文件到达最大尺寸时，该文件将被归档到 netdata.log*MMMDDYYYY_nn*

其中:

MMM 月份 (Jan-Dec)

DD 日期

YYYY 年

nn 一个从 01 到 99 的数字，用于唯一地标识某一天的每个档案文件。

在原来的文件中继续记录。

记录文件格式

记录文件条目具有以下格式:

[DD/MMM/YYYY:HH:MM:SS][MACRO][BLOCK][PID#][TID#]error_message

参数:

DD 日期

MMM 月份 (Jan-Dec)

YYYY 年

HH 小时 (00-23)

MM 分钟 (00-59)

SS 秒 (00-59)

MACRO
生成错误信息的宏文件

BLOCK
生成错误信息的 HTML 块的名称。

PID# 生成错误信息的进程的进程 ID 号码。这个 ID 是必需的，因为多个 Net.Data 进程都可以写入这个记录文件。

TID# 生成错误信息的线程的线程 ID 号码。这个 ID 是必需的，因为来自同一个 Net.Data 进程的多个线程都可以写入这个记录文件。

| *error_message*
| 错误信息的文本

附录A. Net.Data for AIX

AIX 版本的细节已经包含在与 Net.Data 一起提供的“自述文件”中。自述文件中包含以下信息:

- 需求
- 安装
- 配置
- 解除安装

为语言环境装入共享程序库

在 AIX 平台上创建语言环境时，需要装入共享程序库。在 AIX 中，当提供一个由 Net.Data 调用的例程时，或返回语言环境例程(例如 dtw_initialize() 和 dtw_execute())的地址时，需要提供语言环境。

Net.Data 使用 dtw_fp 结构从 AIX 的语言环境中检索指向该语言环境接口例程的指针，其格式如下:

```
typedef struct dtw_fp {
    int (* dtw_initialize_fp)();   /* dtw_initialize 函数指针         */
    int (* dtw_execute_fp)();      /* dtw_execute 函数指针            */
    int (* dtw_getNextRow_fp)();   /* dtw_getNextRow 函数指针         */
    int (* dtw_cleanup_fp)();      /* dtw_cleanup 函数指针            */
} dtw_fp_t;
```

当装入共享库时，这个结构由 Net.Data 作为 dtw_getFp() 例程中的一个参数被传递到语言环境。

作为仅有的参数来传递 dtw_fp 结构。此结构包含一个用于每个所支持接口的字段，并且由语言环境来设置这些字段。如果语言环境提供指定的接口，则它将该字段设置成指向指定接口的函数指针。 如果语言环境没有提供指定的接口，则它将该字段设置成 NULL。程序模板中的 dtw_getFp() 例程显示了此例程的一个正确实现。

为了在装入共享库时 Net.Data 能够获得指向这个例程的指针，dtw_getFp 例程必须是在共享库的调出文件中指定的第一个入口点。以下是 dtwsampshr.o 库的调出文件的一个例子，这个库支持所有可用的语言环境接口例程。

```
#!dtwsampshr.o
dtw_getFp
dtw_initialize
dtw_execute
dtw_getNextRow
dtw_cleanup
```

改进 REXX 环境的性能

如果在 AIX 系统中有许多个对 REXX 语言环境的调用，则可以考虑将 RXQUEUE_OWNER_PID 环境变量设置为 0。而调用此 REXX 语言环境的宏可以很方便地调用许多进程、调用系统资源。

您可以用以下三种方式来设置环境变量:

- 在宏文件中使用 DTW_SETENV 内部函数:

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

- 在 AIX 系统环境文件中:

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

  此方法将影响整个机器上 REXX 的功能。

- 在 HTTP Web 服务器环境文件中; 例如, 对于 Domino Go Webserver 插入以下语句:

  ```
  InheritEnv RXQUEUE_OWNER_PID = 0
  ```

  此方法将影响 Web 服务器上 REXX 的功能。

附录B. Net.Data 向导

Net.Data 向导是为了给您提供一种创建定制 Net.Data 应用程序的便捷方式而设计的。只要简单地选择一个向导，回答几个问题，Net.Data 就会为您创建一个定制应用程序。

Net.Data 为您提供了以下向导，您可以在学习如何创建宏和使用 Net.Data 特征的时候使用这些向导:

下访 此向导能够取得您现有的数据库表并创建一个支持 Web 功能的下访应用程序，这个下访应用程序能使您访问不同详细程度的数据。还可以选择将下访向导连接到数据库，以便收集数据库信息。您最多可以定制 5 个 Web 报表。生成的宏文件使用存储在您数据库中的主要关键字和外部关键字信息来自动链接您的 Web 报表。

论坛 此向导创建一个支持 Web 功能的论坛，可用于公告、察看和回答信息。您可以定制论坛类别和讨论主题。生成的论坛应用程序包含一个在论坛公告中搜索关键字的功能。您还可以选择在宏文件中启用 Cookie 功能，这样就能使论坛『记住』它的用户。

存储过程
此向导将连接至您的数据库，并获取一张所有登记在数据库中的存储过程的列表。选择一个存储过程，向导就会为您生成一个调用存储过程的 Net.Data 宏文件。然后，您可以修改生成的宏文件，或者将它集成到现有的 Net.Data 应用程序中。

联络 此向导生成一个支持 Web 功能的通讯录，可用于存储姓名、地址、电话号码和其它重要的联络信息。它带有一个搜索功能，可以让您快速访问您的联络人。生成的联络应用程序中可以包括一个简短报告或一个详细报告。另外，您也可以包括一个定制报告。

请查看 Net.Data 的 Web 站点: `http://www.software.ibm.com/data/net.data`，以获取 Net.Data 向导软件包的最新版本。

开始之前

要运行此向导并生成 Net.Data 宏文件，必须先安装以下软件:

- Java Development Kit (JDK) 或 Java Runtime Environment (JRE) 1.1.x
- Net.Data 版本 2 或更高版本
- IBM Universal Database (UDB) 5.0 或更高版本
- REXX (下访向导所必需)

运行向导

Net.Data 向导是从命令行启动的，它们包含在文件 `NetDataSmartGuides.jar` 中。

要使用 Java Development Kit (JDK) 来启动向导:

1. 将以下行添加到 CLASSPATH 环境变量中。

   [Path]`NetDataSmartGuides.jar`

其中 *[Path]* 是到 NetDataSmartGuides.jar 文件的可选路径。

2. 如果您想要使用下访和存储过程向导的数据库连接功能，请在 CLASSPATH 环境变量中添加 UDB JDBC 驱动器。

   对于 Windows NT 和 OS/2 操作系统，在您的 CLASSPATH 环境变量中添加以下这一行:

   ```
   [Path]NetDataSmartGuides.jar;[UDBInstallationPath]\java\db2java.zip
   ```

   对于 UNIX 操作系统，请向您的 CLASSPATH 环境变量中添加以下行:

   ```
   [Path]NetDataSmartGuides.jar:[UDBInstallationPath]\java\db2java.zip
   ```

   其中 *[Path]* 是到 NetDataSmartGuides.jar 文件的可选路径， *[UDBInstallationPath]* 是到 UDB 安装的路径，例如 C:\SQLLIB。

3. 启动向导。
   - 对于 UNIX 操作系统，键入以下命令:

     ```
     java TaskGuide LaunchPad.class
     ```

   - 对于 Windows NT 和 OS/2 操作系统，运行以下文件:

     ```
     SmartGuides.cmd
     ```

   将打开一个具有可用向导列表的启动板，如第136页的图 26中所示。

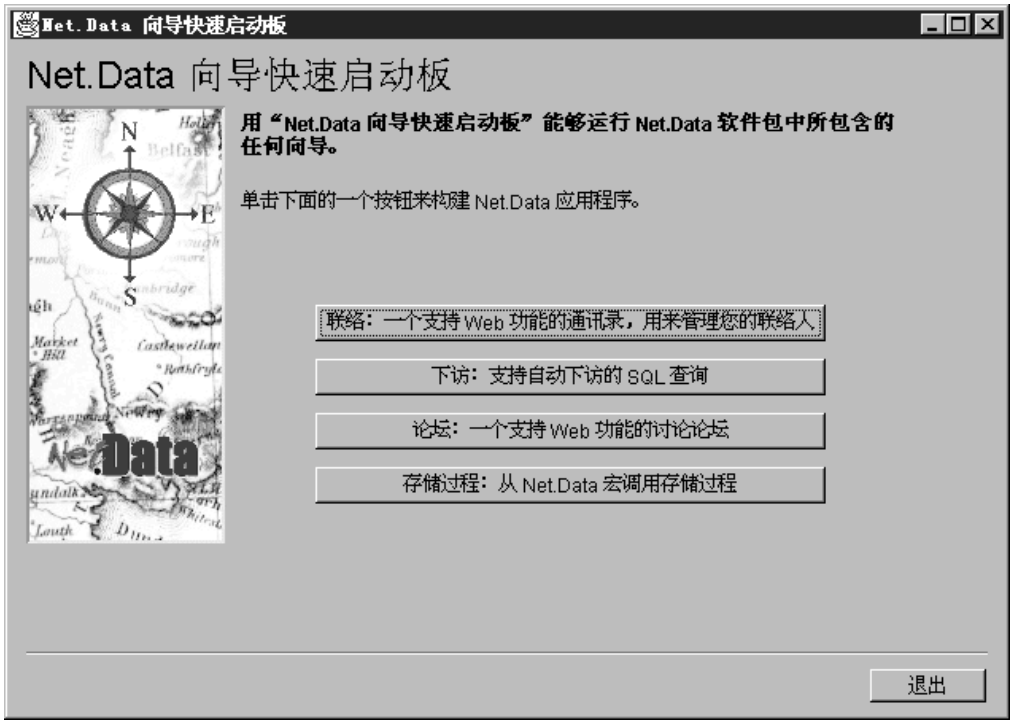


图 26. 向导启动板

4. 单击您想要运行的向导的名称。

要使用 Java Runtime Environment (JRE) 来启动向导:

1. 键入以下命令来运行向导:

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

其中 *[Path]* 是到 NetDataSmartGuides.jar 文件的可选路径。

将打开一个具有可用向导列表的启动板，如第136页的图 26中所示。

2. 如果您想要使用下访和存储过程向导的数据库连接功能，请键入以下命令:

   对于 Windows NT 和 OS/2 操作系统:

   ```
   jre -cp [Path]NetDataSmartGuides.jar;[UDBInstallationPath]
     \java\db2java.zip TaskGuide LaunchPad.class
   ```

   对于 UNIX 操作系统:

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

   其中 *[Path]* 是到 NetDataSmartGuides.jar 文件的可选路径， *[UDBInstallationPath]* 是到 UDB 安装的路径，例如 C:\SQLLIB。

3. 单击您想要运行的向导的名称。

附录C. 使用 NetObjects Fusion NOF 插件和 Net.Data 小服务程序

Net.Data 为 Net.Data 小服务程序提供了一个 NetObjects Fusion 插件。Net.Data 小服务程序在第93页的『Net.Data 小服务程序』中有所描述。

您可以使用 NetObjects Fusion (NOF) 来更好地集成现有的 Net.Data 宏文件和 Web 站点管理，它还提供了一个便于使用的图形用户界面。

有关 NetObjects Fusion 插件

NetDataServlet.NFX 插件与 Net.Data 小服务程序一起使用。这个 NOF 插件支持对现有 Net.Data 宏文件或单个 Net.Data 函数的调用。该插件生成 HTML，将 Net.Data 作为小服务程序或服务器方包含文件 (SSI) 来调用。当 Web 服务器调用 Net.Data 时，Net.Data 宏文件或函数将运行。使用 Net.Data 小服务程序插件将宏和函数小服务程序插入 NOF 管理的 Web 站点，如第139页的图 27所描述。该插件提供了系统基本宏文件或函数参数，以及为自动构建宏文件而选择的缺省项。要使用这个插件，必须安装并配置 NOF 和插件文件。

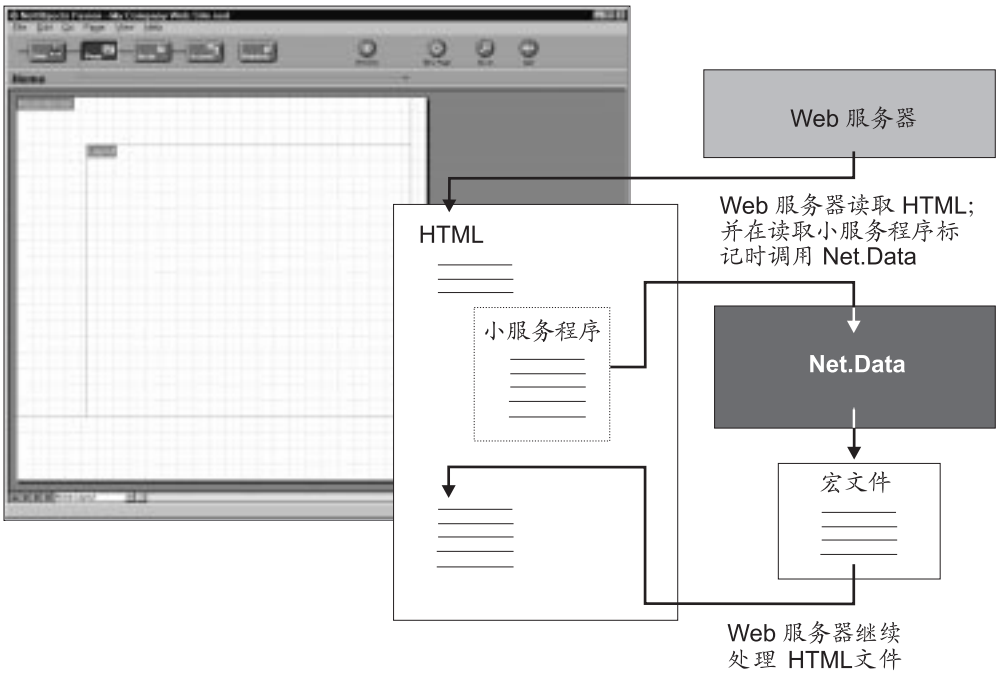


图 27. Net.Data 小服务程序插件

安装 NetObjects Fusion 插件

软件与硬件需求:

NOF 插件需要 NetObjects Fusion 版本 2.0 或更高版本。

要安装: 将 <inst_dir>\fx\NetDataServlet.nfx 和 <inst_dir>\fx\NetDataServlet.gif 复制到您的 <NetObjects Fusion>\components\ 目录。

为 NetObjects Fusion 设置 Net.Data 插件

您可以使用 NOF 来更改正在使用的小服务程序的特性。

1. 打开 NetObjects Fusion。
2. 从 NetObjects Fusion (NOF) 的 **Tools**选用区选择 **NetObjects Components**按钮:
    插件按钮显示在 **Tools**选用区的底部。
3. 从这六个 **Tools**选用区按钮中选择 **NetObjects Components**按钮: 
4. 在 NOF 画面上，标记您想要放置所选插件的区域。这是显示小服务程序结果的地方。将打开显示插件列表的 “Installed Components” 窗口，您可以从中进行选择。如果小服务程序插件不在列表中，可以使用 “path” 已经 “file name” 字段来指定插件的文件名 NetDataServlet.NFX，以便与宏小服务程序以及函数小服务程序一起使用
5. 从列表中选择小服务程序插件并单击 **OK**按钮。

   该插件将变为 NOF 画面上的一个对象。

修改插件的特性

您可以使用 Net.Data 小服务程序插件来修改宏和函数小服务程序。

要使用 NOF 来修改 Net.Data 小服务程序:

1. 在 NOF 画面上，标记您想要放置所选插件的区域。这是显示小服务程序结果的地方。将打开显示插件列表的 “Installed Components” 窗口，您可以从中进行选择。如果小服务程序插件不在列表中，可以使用 “path” 以及 “file name” 字段来指定插件的文件名 NetDataServlet.NFX，以便与宏小服务程序以及函数小服务程序一起使用。
2. 从列表中选择 Net.Data 小服务程序插件并单击 **OK**按钮。

   该插件将变为 NOF 画面上的一个对象。
3. 选择并定制 Net.Data 小服务程序插件的特性:
   a. 在 NOF 画面上选择 Net.Data 小服务程序插件。将打开 NOF **Properties**选用区，显示插件的特性，如第141页的图 28中所示。

图 28. Net.Data 小服务程序 的 Properties 选用区

Net.Data 小服务程序的特性:

您可以定制以下特性:

Servlet name
选择您想要调用的小服务程序的名称: Function 或 Macro。根据您所选择的小服务程序名称，将显示不同的特性。

Servlet type
选择想要的小服务程序类型: SSI、HREF 或 FORM 提交按钮。根据您所想要的小服务程序类型，将显示不同的特性。

Submit label
如果选择 FORM 提交按钮类型，则请指定提交标签的文本。否则，将不显示此特性。

Server URL
如果选择 HREF 小服务程序类型，则需要向支持小服务程序功能的 Web 服务器指定服务器 URL。如果选择 SSI，将不显示此特性。

Macro name
如果选择宏小服务程序的名称，则指定要执行的现有 Net.Data 宏文件的名称。否则，将不显示此特性。

HTML block name
如果选择宏小服务程序的名称，则指定要运行的 Net.Data 宏文件中的 HTML 块的名称。否则，将不显示此特性。

Function type
如果选择函数小服务程序的名称，则选择要执行的函数类型: Function 或 SQL。否则，将不显示此特性。

Language env
如果选择函数小服务程序的名称，则指定要使用的 Net.Data 语言环境。否则，将不显示此特性。

Statement
如果选择函数小服务程序的名称，则指定要执行的语句。否则，将不显示此特性。

Database
如果选择函数小服务程序名，则指定要使用的数据库的名称。否则，将不显示此特性。

# of other parameters
指定要传递给 Net.Data 的其它参数个数(最大: 25)。对于每个参数，输入参数的名称及其值(可选)。

Before HREF text
如果选择 HREF 小服务程序类型，则在要显示在 <A HREF> HTML 标签前的文本前面指定要显示的文本。否则，将不显示此特性(可选)。

Inside HREF text
如果选择 HREF 小服务程序类型，则在 <A HREF> HTML 标签内指定要显示的文本。否则，将不显示此特性(可选)。

After HREF text
如果选择 HREF 小服务程序类型，则在要显示在 <A HREF> HTML 标签后的文本后面指定要显示的文本。否则，将不显示此特性(可选)。

SQL Reminder!
如果选择 HREF 小服务程序类型并指定了一个 SQL 函数类型，将显示带有提示的消息，告诉您 HREF SQL 语句应对任何空格字符 ( ) 使用加号字符 (+)。在页面发行之后，文本既不能更改，也不能显示。否则，将不显示此特性。

b. 在定义页面的特性以后，您就可以单击 **Publish**按钮来用 Net.Data 小服务程序 NOF 插件构建并发行自己的 Web 页面。

注: 如果选择了 SSI 小服务程序类型，Web 页面文件的扩展名就应当是 .shtml。您可以在 NOF **Properties**选用区中将此设置为页面的缺省值: 选择 **Page** 笔记本标签，单击 **Custom names**按钮并在 **Extension Type**字段中输入 .shtml。

使用 NOF 插件发布小服务程序

在设置页面的特性以后，您就可以单击 **Publish**按钮来用插件构建并发行自己的 Web 页面。

附录D. Net.Data 示例宏

这个示例宏应用程序显示了一张雇员列表，应用程序用户可以通过在列表中选择雇员的姓名来获取某个雇员的额外信息。此宏使用 SQL 语言环境来查询 EMPLOYEE 表，从中获取雇员姓名和某个特定雇员的有关信息。

```
%{************************ 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 for     *
*               additional information about the employee identified 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  %}
```

附录E. 注意事项

本信息是为在美国提供的产品和服务而开发的。IBM 在其它国家也许没有提供本文档中所讨论的产品、服务或功能部件。 关于您所在区域目前可用的产品及服务的信息，请向当地的 IBM 代表咨询。任何对 IBM 产品、程序或服务的引用并不说明或暗示只能使用 IBM 的产品、程序或服务。凡是同等功能的产品、程序或服务，只要不侵犯 IBM 的知识产权，都可以用来替代 IBM 产品、程序或服务。当然，评估和验证非 IBM 产品、程序或服务均由用户自行负责。

本文档的议题可能涉及 IBM 的某些专利或正在申请中的专利的应用。提供此文档并不给予您使用这些专利的任何许可。您可以将许可证查询以书面形式发送给:

IBM Director of Licensing
IBM Corporation
500 Columbus Avenue
Thornwood, NY 10594
U.S.A.

关于双字节 (DBCS) 许可证查询的信息，请与您所在国家的 IBM 知识产权部分联系，或通过写信将查询邮寄至:

IBM World Trade Asia Corporation
Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106, Japan

以下段落不适用于英国与其它当地法律不允许这种供应方式的国家: 国际商用机器公司『按现在的样子』出版此书，不做任何明确或暗示的担保，包括但不局限于可销售性或适用于特殊目的暗示担保。一些地区在某些事务中不允许放弃明确或暗示的担保，因此本条款可能不适合您。

本信息中可能有技术方面不够准确的地方或印刷错误。此处的信息将定期更改；这些信息将包含在本书新的版本中。IBM 可以在任何时间对本书中说明的产品或程序进行改进，而不必通知您。

本书中任何对非 IBM Web 站点的引用都只是为了提供方便，并不以任何形式作为对那些 Web 站点的保证。那些 Web 站点的资料并不是本 IBM 产品资料的一部分，使用那些 Web 站点的风险将由您自己承担。

已经获得这个程序许可证的用户，如果希望得到有关的更多信息，以允许: (i) 在独立创建的程序和其它程序(包括本程序)之间交换信息，以及 (ii) 相互使用已经交换的信息，请联络:

IBM Corporation
_W92/H3_
555 Bailey Avenue
P.O. Box 49023
San Jose, CA 95161-9023
_U.S.A.

这些信息可以通过遵循相应的条款和条件来获取，在某些情况下，需支付一定的费用。

这些信息中描述的特许程序及其所有可用的特许资料，按 IBM 客户协议、IBM 国际程序许可证协议或任何等价的协议中的条款，由 IBM 提供。

涉及非 IBM 产品的信息可从这些产品的供应商、其出版宣布或其它公众可用源得到。IBM 未测试这些产品，因此不能确认性能、兼容性或其它关于非 IBM 产品承诺等的准确度。有关非 IBM 产品功能方面的问题可向它们的供应商提出。

所有关于 IBM 未来方向或意向的声明都可能更改或撤消，而不作任何通知，并且仅代表目标和结果。

此信息仅供用于计划。在描述的产品可用之前，此处的信息可能会更改。

此信息中包含了每日商业操作中所使用的数据和报表的例子。为了尽可能完整地进行说明，例子中包含了个人、公司、商标以及产品的名称。所有这些名称都是虚构的，如果与实际商业企业使用的名称和地址相似，则纯属巧合。

商标

以下术语是 IBM 公司在美国或其他国家的注册商标:

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

以下术语是其它公司的商标:

Java 和所有基于 Java 的商标与标志都是 SUN 公司在美国和其它国家的商标。

UNIX 是在美国和其它国家的注册商标，许可权属于 X/Open 有限公司专有。

Lotus 和 Domino Go Webserver 是 Lotus 公司在美国和其它国家的商标。

Microsoft、Windows、Windows NT 和 Windows 标志是 Microsoft 公司在美国和其它国家的商标或注册商标。

其它公司、产品和服务名称，以两个星号(**)表示，可能是其它公司的商标或服务标志。

词汇表

API. Application programming interface 的缩写，应用程序设计接口。

applet (小应用程序). 包含在 HTML 页中的一段 Java 程序。在支持 Java 的浏览器(例如 Netscape) 中可以运行小应用程序，它是在装入 HTML 页时装入的 。

application programming interface (API，应用程序设计接口). 由操作系统或可单独订购的特许应用程序提供的一个功能接口，它允许以高级语言编写的应用程序可以使用特定于操作系统或特许程序的数据。 Net.Data 支持以下这些专门的 Web 服务器 API，这些 API 用于 CGI 进程中的改进性能: ICAPI、GWAPI、 ISAPI 和 NSAPI。

BLOB. Binary large object 的缩写，二进制大型对象。

cache (高速缓存). 一部分包含最近访问过的数据的内存或磁盘空间，是为了加快对相同数据的后继访问而设计的。高速缓存经常是用来对网络中可以访问的、频繁使用的数据保留一个本地副本。

caching (高速缓存). 将频繁使用的结果(来自对 Web 服务器的请求)存储在本地以备快速检索的过程，直到刷新信息时为止。

Cache Manager (高速缓存管理器). 为一台机器管理高速缓存的程序。它可以管理多个高速缓存。

CGI. Common Gateway Interface 的缩写，公共网关接口。

cliette. Net.Data 现场连接中一个长时间运行的进程，为来自 Web 服务器的请求提供服务。连接管理器负责调度 cliette 进程，使其为这些请求提供服务。

CLOB. Character large object 的缩写，字符大型对象。

Common Gateway Interface (公共网关接口). Web 服务器将控制传递给一个应用程序以及接收回数据的一种标准方法。

Connection Manager (连接管理器). Net.Data 中的一个可执行文件 `dtwcm`，用于支持 “现场连接”。

cookie. 一个信息包，由 HTTP 服务器发送给 Web 浏览器，然后在浏览器每次访问该服务器时发回。Cookies 中可以包含服务器所选择的任意信息，用于维持否则将没有状态的 HTTP 事务之间的状态。*计算的自由联机字典*

database (数据库). 表格的一个集合，或表格空间和索引空间的一个集合。

database management system (DBMS，数据库管理系统). 用于控制创建、组织和修改一个数据库，并控制对其中存储的数据进行访问的一个软件系统。

data type (数据类型). 列和字面量的属性。

DBMS. Database management system 的缩写，数据库管理系统。

firewall (防火墙). 一台装有软件的计算机，用于防止外部未授权计算机侵入内部网络。

flat file interface (平面文件接口). 一系列 Net.Data 内部函数，可让您在明文文件中读写数据。

HTML. Hypertext markup language 的缩写，超文本标记语言。

HTTP. Hypertext transfer protocol 的缩写，超文本传送协议。

hypertext markup language (超文本标记语言). 一种用于编写 Web 文档的标记语言。

hypertext transfer protocol (超文本传送协议). 一种在 Web 服务器和浏览器之间使用的通信协议。

ICAPI. Internet Connection API 的缩写。

ICS. Internet Connection Server 的缩写。

ICSS. Internet Connection Secure Server 的缩写。

Internet. 国际公用 TCP/IP 计算机网络。

Internet Connection Server. IBM 公司的公开 Web 服务器。

Internet Connection Secure Server. IBM 公司的安全 Web 服务器。

Intranet. 在公司防火墙内部的 TCP/IP 网络。

ISAPI. Microsoft 公司的 Internet Server API。

Java. 一种独立于操作系统的面向对象的程序设计语言，特别适用于 Internet 应用程序。

language environment (语言环境). 一个模块，提供从 Net.Data 宏到外部数据源(例如 DB2 或诸如 Perl 等程序设计语言)的访问。有一些语言环境是与 Net.Data 一起提供的，例如 REXX、 Perl 和 Oracle。您还可以创建自己的语言环境。

Live Connection (现场连接). 一个 Net.Data 组件，由连接管理器和多个 cliette 组成。现场连接用于管理数据库和 Java 虚拟机连接的再使用。

LOB. Large object 的缩写，大型对象。

middleware (中件). 一种介于应用程序与网络之间的软件。它管理客户应用程序和服务器之间通过网络进行的交互。

NSAPI. Netscape API 的缩写。

null (空值). 表示信息异常的一个特殊值。

path (路径). 用于查找文件的搜索路径。

Perl. 一种解释性编程语言。

port (端口). 一个 16 位数，用于在 TCP/IP 和高级协议或应用程序之间进行通信。

TCP/IP. Transmission Control Protocol / Internet Protocol 的缩写，传输控制协议/网际协议。

Transmission Control Protocol / Internet Protocol (传输控制协议/网际协议). 一组通信协议，同时支持局域网和广域网中的点对点连接功能。

URL. Uniform resource locator 的缩写，统一资源定位器。

uniform resource locator (统一资源定位器). 一个用于命名 HTTP 服务器和(可选的)目录及文件名的地址，例如: `http://www.software.ibm.com/data/net.data/index.html`。

Web server (Web 服务器). (1) 一台运行 HTTP 中件的计算机 (2) 服务器软件，例如 Internet Connection。

索引

本索引按汉语拼音，数字，英文字母和特殊字符顺序排列。

[A]

[B]

[C]

[D]

[F]

[G]

[H]

[J]

[K]

[L]

[N]

[P]

[Q]

[R]

[S]

[T]

[W]

[X]

[Y]

[Z]

A

B

C

D

E

F

G

H

I

J

L

M

N

O

R

U

W

Printed in China