

第6章 编写LotusScript

6.1 简介

由于Domino的Java编程能力会使人感到非常激动，LotusScript就像它的平凡但负责的姐妹不太显眼。LotusScript常在Notes客户端应用程序中而不是Web应用程序中扮演重要的角色，这是因为它的用户界面类（UI），尽管在Notes客户端工作良好但并不受Web浏览器支持。但是，LotusScript与Java的后台类库基本类似，有很长的使用历史并且易于为用户使用和接受。虽然Java代理有一些优点LotusScript并不拥有，如RMI（Remote Method Invocation，远端方法调用）。然而有时LotusScript还是一个较好的选择。许多Notes和Domino开发者在Java开发方面还是一个新手，而既熟悉LotusScript又熟悉Java的人就更少了。LotusScript另外一个优势就是它与Visual Basic非常类似，因此VB的开发者可以进入这个领域。

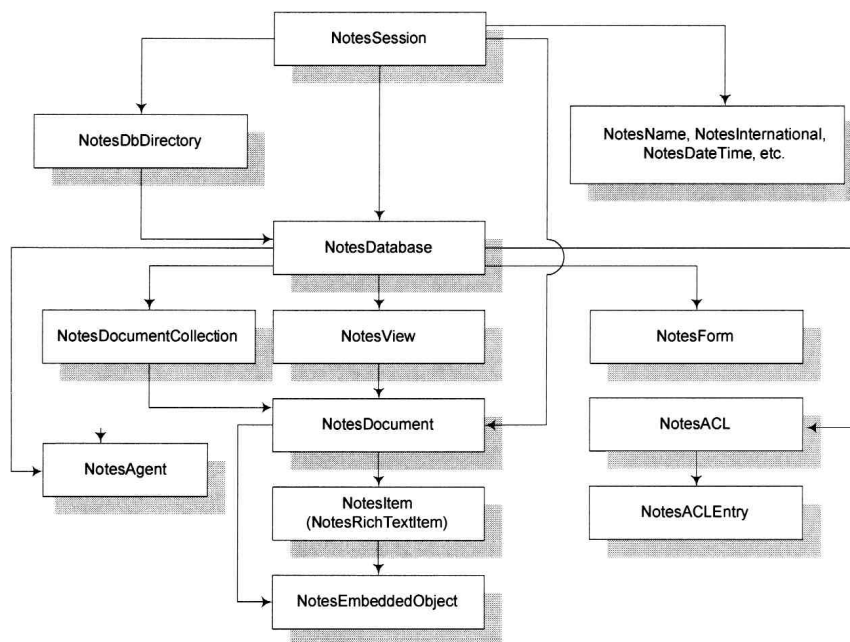


图6-1 基本的LotusScript后台类

本章的目的是说明在Domino的Web开发中使用LotusScript的主要方法。我们假设你至少已经基本熟悉LotusScript。如果需要逐步地介绍LotusScript，最好学习一下来自Domino的LotusScript指南或购买一本关于LotusScript的最新的教材。

LotusScript是Lotus公司为它的几个产品开发的一种通用的语言，而不仅仅是由Notes使用。

为了在Notes中使用对LotusScript的类进行了特别的设置。这些类的设计使某些对象看起来包含另外一些对象。例如，当你想编写一段脚本打开一个特别的 Notes文档时，必须首先得到一个Notes数据库(NotesDatabase)对象，从数据库对象中可以得到一个 Notes视图 (NotesView) 对象,然后从视图对象中得到你想要的 Notes文档对象 (NotesDocument)。

图6-1显示了基本的LotusScript后台类的层次关系，我并不会试着在此描写每一个类、方法、属性。对于其中的大部分来说其名称本来就是解释性的。一个 NotesDatabase对象就是一个Notes的数据库，一个NotesView对象就是一个Notes视图，等等。

注意NotesSession和NotesDocument之间的捷径。通常情况下，为了得到一个 Notes Document对象，你需要得到一个数据库，得到其中的一个 Notes视图或文档集，然后得到一个文档。然而，对于 Web代理来说还可以直接访问当前文档。这个文档是指 NotesSession的 DocumentContext属性。在整个本章的例子中你将经常见到 DocumentContext属性的使用。

6.2 关于LotusScript Web代理

从Web上运行一个LotusScript代理有三种方法：

- 使用它的URL，例如，`http://server/database/agent_name?OpenAgent`。
- 使用一个WebQueryOpen代理。
- 使用一个WebQuerySave代理。

当你从Web上运行一个LotusScript代理的时候，代理将执行由浏览器与 Domino Web服务器提供的信息，比如网络地址或 URL等。这些信息就是 CGI变量。根据选择用来运行代理的方法，这个代理还可以访问当前文档中的域的值。为了得到这个信息，你可以指向NotesSession的DocumentContext属性。

DocumentContext属性是一个特别的NotesDocument对象，它包含了关于当前环境的信息。这个文档内容类似于NotesUIDocument对象，你可以使用它在表单保存前从表单中得到值。另外，DocumentContext属性包含在CGI变量表单中的关于环境的信息。现在来看下面的脚本：

```
Sub Initialize
  Dim session As New NotesSession
  Dim doc As NotesDocument
  Set doc =session.DocumentContext

  ' CGI variables
  Print "User ID from browser= " & doc.REMOTE_USER(0) & "<br>"
  Print "Browser software= " & doc.HTTP_USER_AGENT(0) & "<br>"
  Print "Server software= " & doc.SERVER_SOFTWARE(0) & "<br>"

  ' Form field values
  Print "Subject=" & doc.Subject(0) & "<br>"
  Print "Category=" & doc.Category(0)
End Sub
```

这个代理打印从环境变量中收集的信息，包含 CGI变量和当前表单的域值。当你使用POST方法提交表单的时候，这些信息有效。输出结果可能类似于如下：

```
User ID from browser=Rose Kelleher
Browser software=Mozilla/4.04 [en] (Win95; I)
Server software=Lotus-Domino/5.0
Subject=This is a test
Category=Tests
```

注意代理输出时包含HTML换行符
。这是由于我们将直接输出到 Web浏览器中。在一个Domino代理中，Print语句直接输出到用户的浏览器中，因此你完全可以使用 HTML标签对你的输出进行格式化。

你还可以使用LotusScript的Print语句把浏览器重新指向不同的 Web页，比如说你可以输出一个包含在方括号中的URL，例如：

```
Print "[http://www.acme.com]"
```

Print语句使用户能够非常简单地获得正在运行的信息，例如你可以使用 Print语句：

- 当用户输入无效值时显示错误信息。
- 提示用户表单提交成功。
- 显示相关数据库查询结果。
- 根据用户的输出显示不同的响应。
- 显示到其他文档或其他 URL的<a href>链接。

6.3 激活一个代理的URL

你可以通过简单地将浏览器指向一个代理 URL运行一个代理。例如下面的 URL运行Show CGI Info代理

```
http://server/database/Show+CGI+Info?OpenAgent
```

当你使用这种方法运行代理的时候，代理可以访问 CGI环境变量，但是不能访问当前文档域值，因为现在实际上并没有当前文档。这类似于在 Notes中打开代理面板，选择代理，然后选择操作-运行，只是在此你将得到 CGI变量。例如，假设你在表单中创建一个热点执行如下公式：

```
@Command([ToolsRunMacro]; "Show CGI Info")
```

让我们假设你还没有打开数据库属性中的 " Use JavaScript when generating Pages " 选项（这个选项改变了按钮和热点的活动）。在这个环境下，Domino把公式转换为指向代理URL的HTML链接；例如：

```
<a href="http://server/db/Show1CGI1Info?OpenAgent">Click Here</a>
```

当你点击这个链接的时候，运行代理，但是并不提交表单，因此代理不能访问当前的Subject和Category域的当前值，因此输出类似于如下：

```
User ID from browser=Rose Kelleher
Browser software=Mozilla/4.04 [en] (Win95; I)
Server software=Lotus-Domino/5.0
Subject=
Category=
```

而且若非用户已经被验证（换句话说，已经输入了用户 ID和口令），则REMOTE_USER变量也为空。

6.4 在LotusScript中编写WebQueryOpen代理

WebQueryOpen代理运行于以下情况：

- 用户请求一个文档之后。
- 文档被转换为HTML以前。
- 在计算显示域被刷新以后。

通常情况下，WebQueryOpen代理不如WebQuerySave有用。用户学习WebQueryOpen代理的第一个念头是记录文档查询。不错，这确实是可行的，而且我也确实提供了一个例子，但是记录查询有更好的方法（甚至为此有一个特别的工具，Web Manager，可以从http://www.notes.net中下载，Web Manager工具通过使用Notes API的扩展管理功能执行）。

因为WebQueryOpen只在文档在浏览器显示之前执行，它不能在浏懒器上进行任何输出，因此将忽略所有Print语句。

有两种方法创建WebQueryOpen代理。最好是编辑表单的WebQueryOpen事件，使用你想运行的代理名代替<Your agent goes here>。例如，在文档打开之前运行如下WebQueryOpen公式运行Compute New Values代理：

```
@Command([ToolsRunMacro]; "Compute New Values")
```

创建WebQueryOpen事件的老方法（由R4.5留下来的）是创建一个名为 \$\$QueryOpen Agent的隐藏域，使用一个公式值返回代理名。

6.4.1 简单的LotusScript WebQueryOpen代理

Navigator3.04 ™	Navigator4.05 ™	Domino4.6.1 ™
Explorer3.0 ™	Explorer4.01 ™	Domino5.0 ™

这个LotusScript数据库例子（LSEexamples.nsf）包含一个Hit Tracker表单，它保持了打开并阅读表单的人名。这个表单包含如表 6-1的域。

表6-1 一个Hit Tracker表单

域 名	类 型	公 式
REMOTE_USER	计算并显示	REMOTE_USER
Hits	可编辑，数值型，当编辑时隐藏	N/A
HitLog	可编辑，允许多值，当编辑时隐藏	N/A
EditMode	计算并显示	@If(@IsDocBeingEdited; "1"; "0")

Hit Tracker表单的WebQueryOpen事件包含如下公式：

```
@Command([ToolsRunMacro]; "Log Hit")
```

这个公式在文档打开之前运行Log Hit代理：

```
Sub Initialize
  Dim session As New NotesSession
```

```

Dim doc As NotesDocument
Dim item As NotesItem
Dim hits As Integer
Dim user As String

Set doc = session.DocumentContext
If Not (doc.IsNewNote Or doc.EditMode(0) = "1") Then
    user = doc.REMOTE_USER(0)
    If (user = "") Then
        user = "Anonymous"
    End If
    hits = doc.Hits(0)
    doc.Hits = hits + 1
    Set item = doc.GetFirstItem("HitLog")
    Call item.AppendToTextList("Opened by " & user & " on " & Today)
    Call doc.Save(True, True)
End If
End Sub

```

每次用户读文档的时候，这个代理在 Hits域中依此增加并在 HitLog 域中记录用户的名字和当前日期。这个代理将在文档打开之前运行，因此总能够显示最新信息。图 6-2显示了在 Netscape Navigator中显示的Hit Tracker表单。

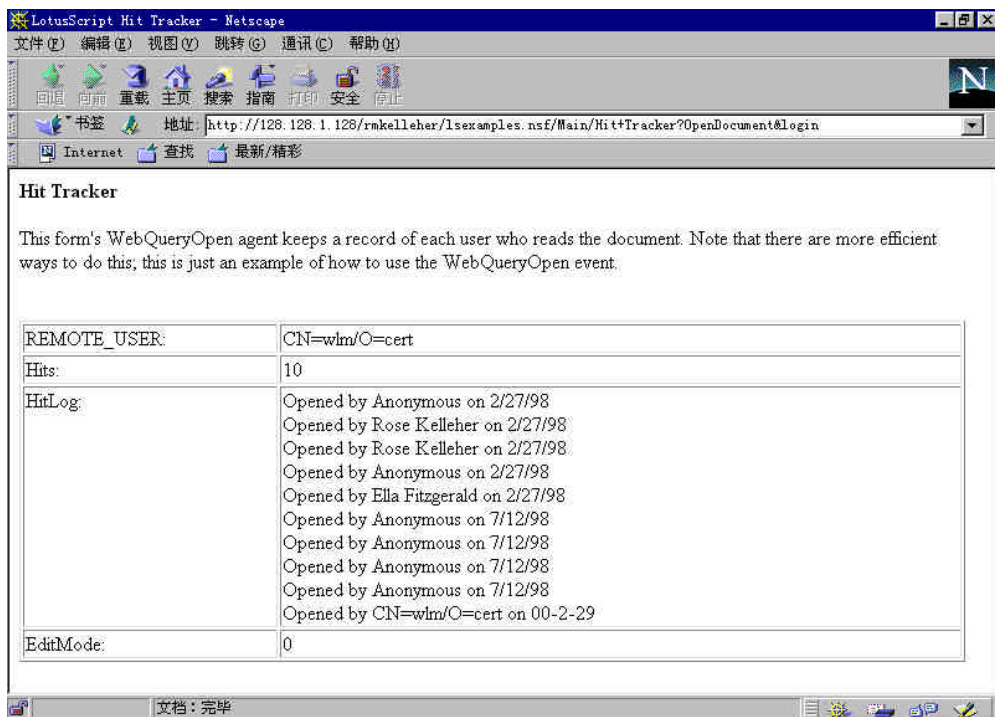


图6-2 一个Hit Tracker表单

提示 如果你在表单中包含一个名为 UserName 的计算显示域并返回值 @UserName , 下面的代码

```
user = doc.REMOTE_USER(0)
If (user = "") Then
    user = "Anonymous"
End If
```

可以简化, 上述代码可以被替代为

```
user = doc.UserName(0)
```

这个问题的例子在于当两个用户同时打开同一个文档的时候, 将产生一个存储冲突。更好的方法是创建一个 NotesLog, 然后你就可以调用 LogAction 方法记录每次打开的信息, NotesLog 可以把每一次输入写入文本文件的一行, 或者 Notes 注册数据库的一个文档。

6.4.2 查询数据库

Navigator3.04 ™ Navigator4.05 ™ Domino4.6.1 ™
Explorer3.0 ™ Explorer4.01 ™ Domino5.0 ™

LotusScript 例子数据库 (LSExamples.nsf) 包含一个 Birthday 表单, 它使用 WebQueryOpen 代理在文档打开之前进行一个对 MS Access 97 数据库的开放式数据库链接查询 (ODBC)。查询结果包括生日在今天的人名。把名字写入文档域中的代理在文档显示以前运行。

```
Option Public
Uselsx "*LSXODBC"

Sub Initialize

    On Error Goto ErrorHandler

    Dim session As New NotesSession
    Dim doc As NotesDocument
    Dim conn As New ODBCConnection
    Dim query As New ODBCQuery
    Dim result As New ODBCResultSet
    Dim sql As String
    Dim mm, dd, lastName, firstName, fullName As String
    Dim people As NotesItem

    ' Nothing to do if we can't connect
    If (conn.ConnectTo("Address Book")) Then

        ' Get the current document
        Set doc = session.DocumentContext
        doc.People = ""
        Set people = doc.GetFirstItem("People")

        ' Build the MS Access query
```

```

mm = Trim(Str(Month(Today)))
dd = Trim(Str(Day(Today)))
sql = "SELECT LastName, FirstName FROM Addresses " & _
      "WHERE Month(Birthdate)=" & mm & " AND Day(Birthdate)=" & dd
' Execute the query
Set query.Connection = conn
query.SQL = sql
Set result.Query = query
result.MaxRows = 10
Call result.Execute

' Cycle through the result set
Do Until result.IsEndOfData
    result.NextRow
    firstName = result.GetValue("FirstName")
    lastName = result.GetValue("LastName")
    fullName= firstName & " " & lastName
    Call people.AppendToTextList(fullName)
Loop
Call conn.Disconnect
End If
Exit Sub

ErrorHandler:
Call conn.Disconnect
Exit Sub
End Sub

```

ErrorHandler:

Call conn.Disconnect

Exit Sub

End Sub

图6-3显示了在2月27日显示的文档 (Chaz 和Elaine的生日)。

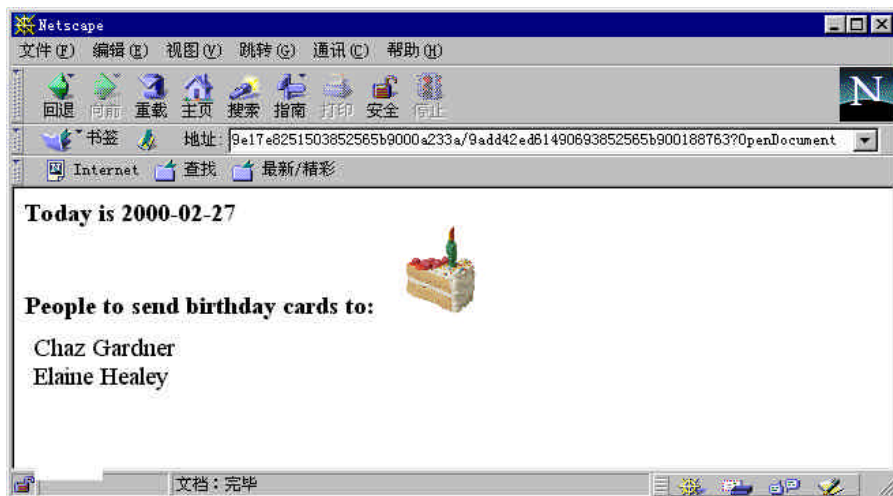


图6-3 由WebQueryOpen代理进行ODBC查询的结果

与Notes客户端应用程序不同，这个程序不要求在客户端安装任何特别的软件，对于 Notes 客户端程序来说，由文档的 QueryOpen事件触发的ODBC查询，事件脚本在客户端执行，这将要求终端用户链接到 MS Access数据库中。由于 WebQueryOpen代理在服务器端执行，只有服务器需要ODBC链接，地址簿数据库必须被注册为 Domino服务器的数据源，但是并不需要驻留在服务器中，只要Domino服务器链接到该Access数据库中即可。

6.5 在LotusScript中编写WebQuerySave代理

WebQuerySave代理在以下情况下运行：

- 当用户提交一个表单之后。
- 当计算域和有效的公式被刷新之后。
- 在文档保存在硬盘之前。

WebQuerySave代理对于复杂的不能通过公式完成的有效性逻辑验证非常有用。当保存的时候工作流把文档自动指向不同的用户 WebQuerySave代理也非常方便。使用 WebQuerySave代理的常用方法是捕捉用户在表单中的兴趣或爱好的信息，然后当提交表单的时候据此进行反应。

与 WebQueryOpen代理相同， WebQuerySave代理也通过 NotesSession对象的 DocumentContext属性访问当前文档。与 WebQueryOpen代理不同的是， WebQuerySave代理可以输出到浏览器或把浏览器重新指向 URL。

6.5.1 简单的LotusScript WebQuerySave代理

Navigator3.04 TM Navigator4.05 TM Domino4.6.1 TM

Explorer3.0 TM Explorer4.01 TM Domino5.0 TM

LotusScript例子数据库（LSExamples.nsf）包含一个Download表单提示你选择一个产品，如图6-4所示。



图6-4 在LotusScript Example数据库中等 Download表单

在提交表单的时候，WebQuerySave事件执行RedirectEBrowser代理，该代理将浏览器根据你的选择指向相应的URL：

```
Sub Initialize
    Dim session As New NotesSession
    Dim doc As NotesDocument

    Set doc = session.DocumentContext
    Select Case (doc.Product(0))
    Case "Lotus Domino"
        Print "[http://www.notes.net]"
    Case "Netscape Navigator"
        Print "[http://home.netscape.com]"
    Case "Sun JDK"
        Print "[http://java.sun.com]"
    End Select
End Sub
```

注意这个文档并未保存，因为 Download表单包含一个值为0的SaveOptions域。这个文档只是一个关于用户要求信息的临时的持有者。

6.5.2 在表中显示ODBC搜索结果

Navigator3.04 ™	Navigator4.05 ™	Domino4.6.1 ™
Explorer3.0 ™	Explorer4.01 ™	Domino5.0 ™

原来的代理是一个非常简单的例子，它收集临时文档的信息并根据信息决定输出的不同。在这个例子中，我们在同样的概念中再走远一点。

LotusScript例子数据库（LSEExamples.nsf）包含一个 Address Book Search表单，你可以使用它查询一个MS Access数据库。图 6-5是只Netscape Navigator中表单的显示结果。

Whom do you want to search for? - Netscape

文件(F) 编辑(E) 视图(V) 跳转(G) 通讯(C) 帮助(H)

后退 向前 重载 主页 搜索 指南 打印 安全 停止

书签 地址: /rmkelleher/LSEExamples.nsf/c957d8ca74e90f84852565b9001f8eb5?OpenForm

Internet 查找 最新/精彩

Whom do you want to search for?

Enter a full or partial name.

First Name:

Last Name:

文档: 完毕

图6-5 Address Book搜索表单

在这个例子中，用户在两个域中输入 AN，当表单被提交的时候，Name Lookup代理将搜索姓或名中包含 AN的内容。（注意：你的 SQL 语句的内容与你使用的数据库软件内容有关。MS Access 97 不区分大小写。对于其他数据库，需要使用 UPPER（）函数保证你的查询区分大小写）。

当提交表单的时候，NAME Lookup搜索MS Access数据库，找到包含特殊字符串的名称并把搜索结果输出到HTML表格中，以名字排序。

```
Option Public
Uselsx "*LSXODBC"

Sub Initialize
    On Error Goto ErrorHandler
    Dim session As New NotesSession
    Dim doc As NotesDocument
    Dim conn As New ODBCConnection
    Dim query As New ODBCQuery
    Dim result As New ODBCResultSet
    Dim sql As String
    Dim lastName, firstName As String

    ' Nothing to do if we can't connect
    If (conn.ConnectTo("Address Book")) Then

        ' Get the current document
        Set doc = session.DocumentContext
        lastName = doc.LastName(0)
        firstName = doc.FirstName(0)
```

最后几行代码根据用户的输入建立了 SELECT 代码。这样一来，用户不必熟悉 SQL 或搜索数据库的 MS Access 的属性。

```
' Build the MS Access query
' (clever algorithm supplied by Dovid Gross)
sql = "SELECT LastName, FirstName, HomePhone FROM Addresses"
cond1$ = "LastName LIKE '%" & lastName & "%'"
cond2$ = "FirstName LIKE '%" & firstName & "%'"
Select Case Cstr(lastName = "") & Cstr(firstName = "")
    Case "FalseFalse": where = " WHERE " & cond1$ & " OR " & cond2$
    Case "TrueTrue": where = ""
    Case "TrueFalse": where = " WHERE " & cond2$
    Case "FalseTrue": where = " WHERE " & cond1$
End Select
sql = sql & where & " ORDER BY LastName"

DoQuery:
    ' Print the column headers
    Print "<center>"
    Print "<h1>Query Results</h1>"
```

```

Print "<table border=1>"
Print "<tr>"
Print "<td bgcolor=""pink""><b>Last Name</b></td>"
Print "<td bgcolor=""pink""><b>First Name</b></td>"
Print "<td bgcolor=""pink""><b>Home Phone</b></td>e"
Print "</tr>"

' Execute the query
Set query.Connection = conn
query.SQL = sql
Set result.Query = query
result.MaxRows = 10 ' This is arbitrary; we could also
                    ' capture this value in the form
Call result.Execute

```

一旦搜索结果有效，代理在结果集中进行循环，在 HTML 列表中输出搜索的结果。

```

' Cycle through the result set
Do Until result.IsEndOfData
    result.NextRow
    Print "<tr>"
    Print "<td>" & result.GetValue("LastName") & "</td>"
    Print "<td>" & result.GetValue("FirstName") & "</td>"
    Print "<td>" & result.GetValue("HomePhone") & "</td>"
    Print "</tr>"
Loop
Call conn.Disconnect

' Terminate the table
Print "</table>"
Print "</center>"
End If
End If
Exit Sub

ErrorHandler:
    Call conn.Disconnect
    Exit Sub
End Sub

```

图6-6显示了一个搜索结果例子。注意列标题有粉红色背景，这是因为在代理的输出中包含BGCOLOR属性。

关于创建 WebQuerySave 代理连接到关系性数据库中应该提一句，当我在 32M 内存的 Windows 计算机上使用 Microsoft MS Access 驱动器测试这个例子的时候，它的速度惊人的缓慢。在 Terrance Crow 的文章 “Domino Does ODBC” 一文中，作者推荐使用高性能的 ODBC 驱动器，并且注意使用服务器的硬盘，处理器和内存。如果你的网络非常通畅的话，如果数据库驻留在其他计算机上则你会得到更好的效果。

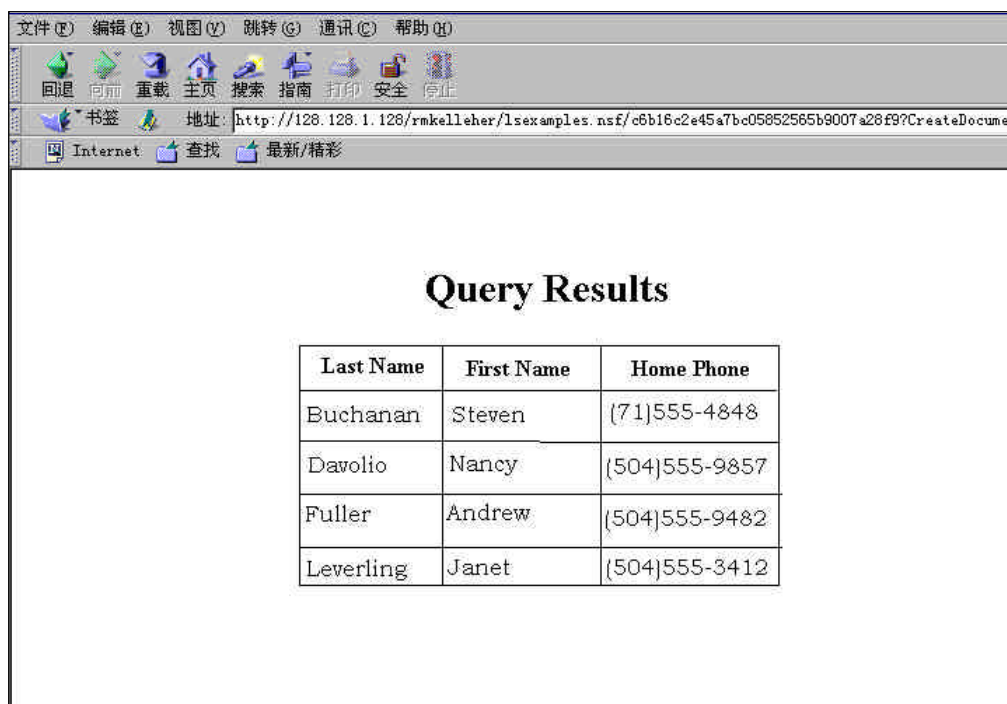


图6-6 从NameLookUp代理中的输出

6.6 在LotusScript代理中使用CGI变量

当你编写基于Web的LotusScript代理的时候，CGI变量非常有用。例如，HTTP_USER_AGENT返回用户浏览器的软件类型。可以使用这个信息使浏览器根据其软件的不同指向不同的URL（对于Netscape Navigator来说指向JavaScript,对于Internet Explorer指向VBScript）。还可以使用HTTP_COOKIE变量从浏览器中得到cookie的信息，QUERY_STRING提供了把自变量传递到代理中的方法。

这一节提供了一个QUERY_STRING和一个HTTP_COOKIE的例子。

6.6.1 QUERY_STRING代理

Navigator3.04 TM Navigator4.05 TM Domino4.6.1 TM
Explorer3.0 TM Explorer4.01 TM Domino5.0 TM

记得前面关于MS Access查询的例子吗？这个例子只是再向前走小小的一步，不是仅仅显示那些在搜索中被发现的人的名字和电话号码，这个例子还提供了HTML链接，点击链接你可以查询关于个体的细节查询，如图6-7所示。

在LotusScript例子数据库中的Name Lookup 2代理几乎与Name Lookup代理相同，只是它在OpenAgent URLs的输出中包含一个HTML链接：

```
Do Until result.IsEndOfData
```

```
result.NextRow
Print "<tr>"
addressID$ = result.GetValue("AddressID")
recordURL$ = "/" & dbFileName & "/GetDocByID?OpenAgent&ID=" & _
    addressID$
Print "<td><a href=" & recordURL$ & ">" & _
    result.GetValue("LastName") & "</a></td>"
Print "<td>" & result.GetValue("FirstName") & "</td>"
Print "<td>" & result.GetValue("HomePhone") & "</td>"
Print "</tr>"
Loop
```

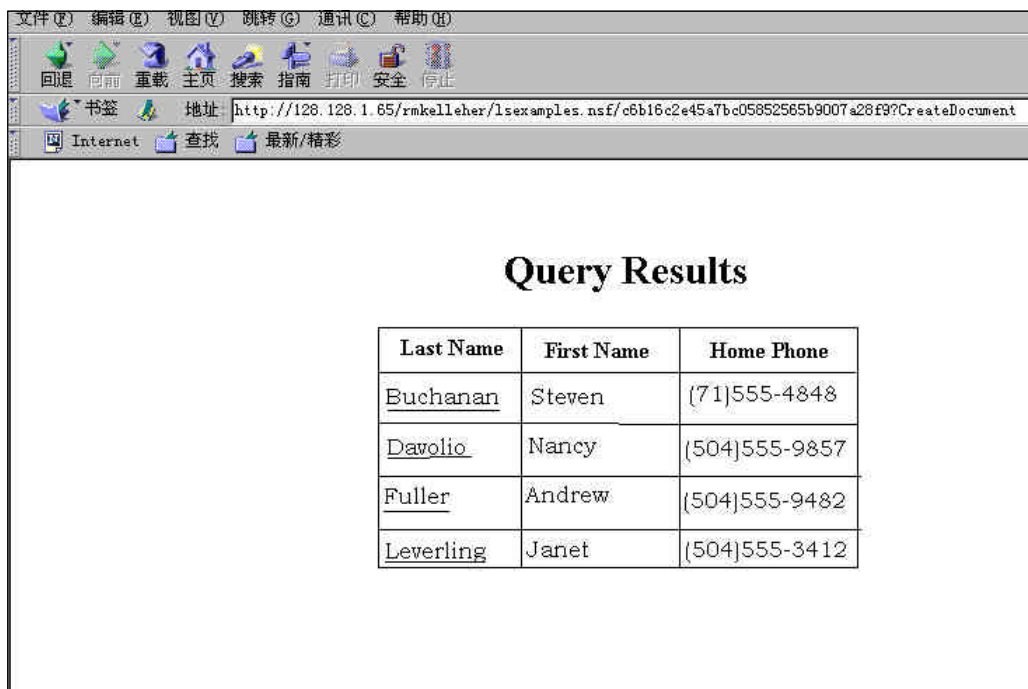


图6-7 包含链接的MS Access查询结果

HTML连接的结果调用一个URL例如：

http://server/RMKelleher/LSEExamples.nsf/GetDocByID?OpenAgent&ID=3

注意在URL结尾处的参数“ID=3”。这个参数被传递到GetDocByID代理以便代理了解到从数据库中得到的是哪一个记录。但是，代理怎样得到这个值呢？

很方便，QUERY_STRING CGI环境变量把所有东西保存在代理URL的问号的右边。这样，在上述例子中QUERY_STRING的值就是“OpenAgent+ID=3”。现在所有的代理所要做的是分析从ID中得到的字符串。

GetDocByID代理所要做做的就是这些。首先它从当前文档中得到QUERY_STRING的值，然后它使用GetParameter函数提取ID参数。一旦确定了要查询的ID，它进行另外一个ODBC查

询并输出相应记录的细节信息，如图 6-8所示。

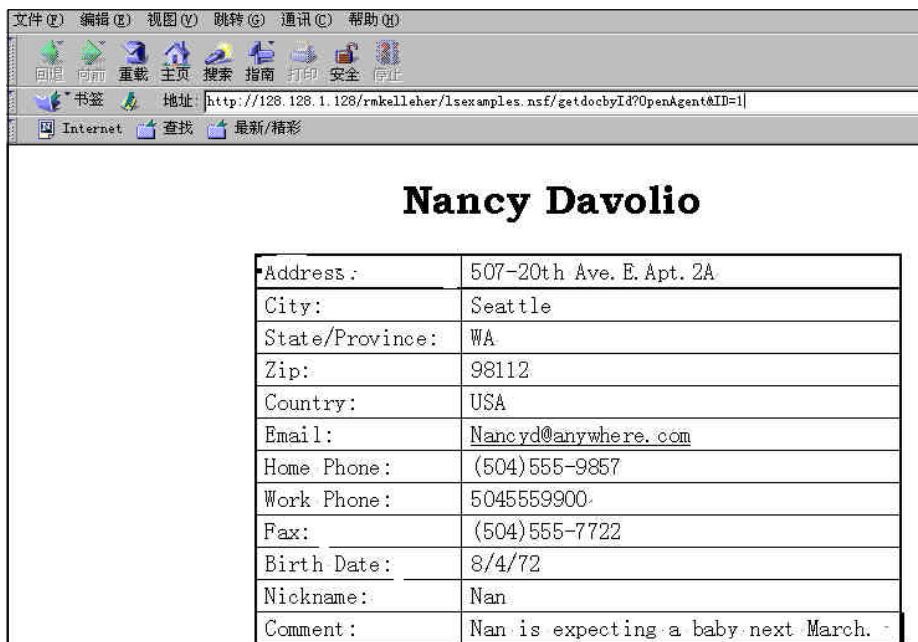


图6-8 一个使用ODBC的个人记录

```
'Get the ID of the requested record
Set doc = session.DocumentContext
queryString$ = doc.QUERY_STRING(0)
addressID$ = GetParameter("ID", queryString$)

' Build the MS Access query
sql = "SELECT * FROM Addresses WHERE AddressID=" & addressID$

' Execute the query
Set query.Connection = conn
query.SQL = sql
Set result.Query = query
result.MaxRows = 10
Call result.Execute

' Cycle through the result set
If (result.IsResultSetAvailable) Then
    result.NextRow
    Print "<center>"
    Print "<h1>" & r.GetValue("FirstName") & " " & _
    result.GetValue("LastName") & "</h1>"
    Print "<table border=1>"
    Print "<tr><td>Address:</td><td>" & result.GetValue("Address") & _
    "</td></tr>"
```

```

Print "<tr><td>City:</td><td>" & result.GetValue("City") & _
"</td></tr>"
Print "<tr><td>State/Province:</td><td>" & _
result.GetValue("StateOrProvince") & "</td></tr>"
Print "<tr><td>Zip:</td><td>" & result.GetValue("PostalCode") & _
"</td></tr>"
Print "<tr><td>Country:</td><td>" & result.GetValue("Country") & _
"</td></tr>"
mail$ = result.GetValue("EmailAddress")
mailUrl$ = "mailto:" & mail$
Print "<tr><td>Email:</td><td><a href=""" & mailUrl$ & """>" & _
mail$ & "</a></td></tr>"
Print "<tr><td>Home Phone:</td><td>" & _
result.GetValue("HomePhone") & "</td></tr>"
Print "<tr><td>Work Phone:</td><td>" & _
result.GetValue("WorkPhone") & "</td></tr>"
Print "<tr><td>Fax:</td><td>" & result.GetValue("FaxNumber") & _
"</td></tr>"
Print "<tr><td>Birth Date:</td><td>" & _
result.GetValue("Birthdate") & "</td></tr>"
Print "<tr><td>Nickname:</td><td>" & _
result.GetValue("Nickname") & "</td></tr>"
Print "<tr><td>Comment:</td><td>" & result.GetValue("Notes") & _
"</td></tr>"
Call conn.Disconnect
Print "</table>"
Print "</center>"
End If

```

如果你在Web中常使用LotusScript代理的话，就会了解QUERY_STRING环境变量的用处，尽管在这个例子中使用它只传递了一个变量，但在其他情况下完全可以传递多个变量，例如：

<http://server/database/agent?OpenAgent&color=red&size=10&qty=5>

在这个例子中，你可能觉得使用可以重用的 LotusScript 函数来进行参数的分析可能非常重要。在Web函数脚本库的GetParameter函数分析查询字符串并返回特定的参数的值：

```

Function GetParameter(Byval paramName As String, Byval queryString As
String) As String
    ' Given a QUERY_STRING, this function returns the value of the
    ' specified parameter. For example,
    ' GetParameter("SIZE", "color=red&size=10&qty=4") returns "10".
    Dim startPos As Integer, endPos As Integer, skipLen As Integer
    queryString = queryString + "&" + paramName + "=" + "&"
    paramName = Ucase("&" + paramName + "=")
    skipLen = Len(paramName)
    startPos = Instr(Ucase(queryString), paramName) + skipLen
    endPos = Instr(startPos, queryString, "&")
    GetParameter = Mid(queryString, startPos, endPos-startPos)
End Function

```


6.6.2 HTTP_COOKIE代理

Navigator3.04 TM Navigator4.05 TM Domino4.6.1 TM

Explorer3.0 o Explorer4.01 TM Domino5.0 TM

在Notes的简要表文档中保存关于用户的信息非常容易，特别当为数量较少的用户建立 Intranet的时候。但是当你的站点通信负担较重，Cooking可能是更好的选择。一个Cookie是一段浏览器保存在用户计算机上的数据，良好的 Web应用程序保证它们设置的 Cookie只耗费合理的时间，而不是储存大量的信息。

得到和设置 Cookie值的最方便的方法是使用 JavaScript。当然，这需要用户使用支持 JavaScript的浏览器。在服务器端，LotusScript是分析cookie字符串并根据它们的值产生适当的反应的好工具。

Cookie数据库(Cookie.nsf)使用一个在线零售商解释了 LotusScript的操纵Cookie数据的能力。如图6-9的零售商店是一个包含内嵌的视图的表单。视图使用列公式产生支持 JavaScript的加(+)和减(-)按钮，可以增加或减少存储在 cookie中的项目数量。



图6-9 Grocery Store 表单

当提交表单的时候，WebQuerySave代理保留cookie的信息，并将其存放在HTTP_COOKIE域中，分析它，并向用户显示当前的总数：

```
' Get cookies
cookie$ = Unencode(doc.HTTP_COOKIE(0))
doc.HTTP_COOKIE = cookie$
Dim eval1, eval2, eval3 As Variant
total@ = 0
```

```

' Split up the cookies
eval1 = Evaluate("@Explode(HTTP_COOKIE; ";");", doc)
Forall s In eval1

' Split up each cookie into label and value
eval2 = Evaluate("@Explode("'" & s & "'"; "=")", doc)
If Not (Ubound(eval2) = 1) Goto NextItem
If Not (Isnumeric(eval2(1))) Goto NextItem
qty% = eval2(1)

' Split cookie label into name and price
eval3 = Evaluate("@Explode("'" & eval2(0) & "'"; "@")", doc)
If Not (Ubound(eval3) = 1) Goto NextItem
If Not (Isnumeric(eval3(1))) Goto NextItem
item$ = eval3(0)
price@ = eval3(1)
itemTotal@ = price@ * qty%
total@ = total@ 1 itemTotal@

' Print cookie data
Print "<tr>"
Print "<td align=left>" & item$ & "</td>"
Print "<td align=right>" & qty% & "</td>"
Print "<td align=right>" & Format(price@, "Currency") & "</td>"
Print "<td align=right>" & Format(itemTotal@, "Currency") & "</td>"
Print "</tr>"

```

图6-10显示了结果表单确认界面，注意对 Evaluate语句的重复使用。Evaluate语句允许你使用Notes公式进行计算而不必使用复杂的 LotusScript。在这个例子中，我们使用 @Explode函数分割字符串，因为，LotusScript不提供任何字符处理函数。

在早期的版本中，Evaluate语句只能使用普通的字符串作为公式。而现在已经取消了这个障碍，使用公式与 LotusScript配合非常简单，例如：

```
eval2 = Evaluate("@Explode("'" & s & "'"; "=")", doc)
```

Unencode函数在代理的开始使用，它属于 Web函数脚本库，这个函数用来处理 cookie和 LotusScript的URL。它把加号转换为空格，把十六进制代码转换为字符。（例如，把 "%2C" 转换为 ","）

```

Function Unencode(theString As String) As String
    Dim newString As String
    Dim char As String
    Dim hexStr As String
    Dim n, i, jumpTo As Integer
    Dim charNum As Long

    newString = ""
    parsingHex = False

```

```

n = Len(theString)
jumpTo = 1
For i = 1 To n
    If (jumpTo > i) Then
        Goto EndOfLoop
    End If
    char = Mid(theString, i, 1)
    If (char = "+") Then
        char = " "
        jumpTo = i + 1
    ElseIf (char = "%") Then
        ' Convert Hex to Decimal
        hexStr = Mid(theString, i, 2)
        charNum = CInt("&H" & hexStr)
        char = Chr(charNum)
        jumpTo = i + 3
    Else
        jumpTo = i + 1
    End If
    newString = newString & char
EndOfLoop:
Next
Unicode = newString
End Function

```

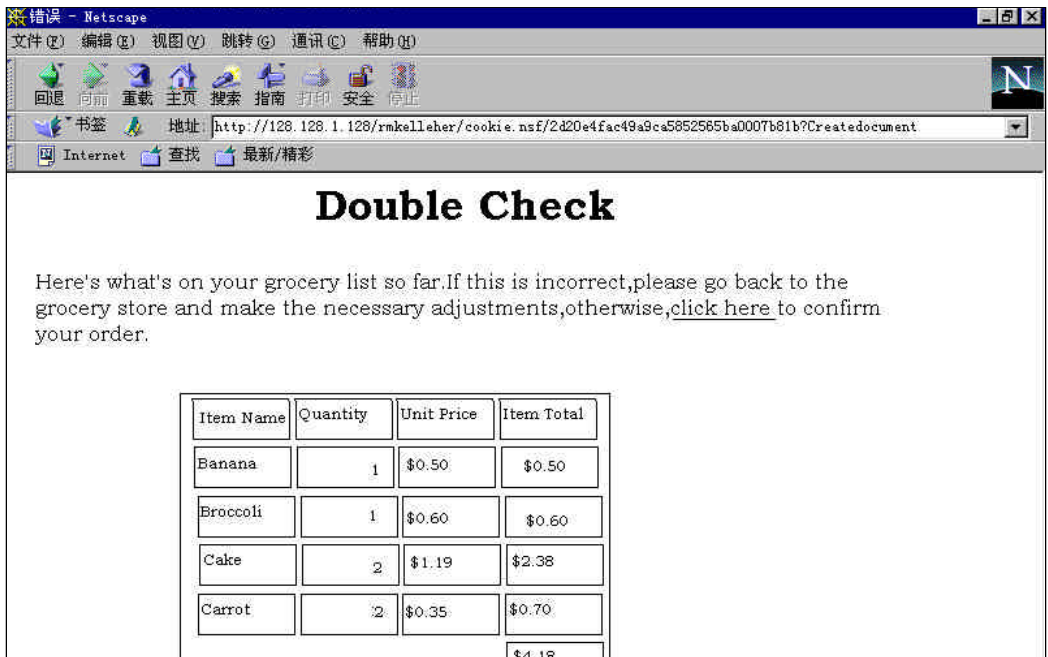


图6-10 订购确定屏幕显示

通常JavaScript函数处理和设置 cookie在Cookie.nsf的数据库的cookie函数子表单有效。子表单是保存JavaScript函数以便将来重新使用的好地方，当你希望重新使用 cookie的表单，你只需要简单地将子表单插进去，然后，Cookie就可以使用了。

6.7 在基于Web的工作流中使用LotusScript

Java可以完成许多LotusScript不能完成的功能：applet、CORBA、awt库都是一些很好的例子。当你使用LotusScript不能完成任务的时候，使用Java，本书关于Java一章集中讲述了那些只有Java才能完成的任务。但是对于更多的工作来说 LotusScript同样有效，基于Notes的工作流就是其中的一例。

工作流是什么呢？许多Notes开发者乐于暗示其为科学的、复杂的名词。但是对于大多数Notes工作流应用程序来说事实上是相当简单的。工作流的应用程序的工作就是帮助把一个应用任务向前推进，例如书写和编辑一个文档，或者批准一个请求，通过对组织中的一系列人员指定路线完成这项工作。

许多Notes客户应用程序要求所有的参与者都在服务器上拥有一个Notes邮件帐号。有些工作流应用程序使用存储的表单，这意味着诸如文档，表单等都被送到用户的Notes邮件数据库中。另外一种工作流应用程序使用Notes的文档连接属性在Notes便笺中发送一个到文档的链接。用户可以使用Web浏览器打开Notes邮件数据库并点击文档链接，该链接将转换为HTML链接。

现在要求用户使用一个单独的邮件包显得有点麻烦，尤其现在Domino服务器可以轻松地使用SMTP收发Internet邮件。对于基于Web的工作流应用程序，向一个用户发送一个指向文档的链接的最好的方法是简单地把指向文档的URL附加在邮件中。大部分邮件软件把URL显示为一个链接，当点击这个链接的时候自动下载目标页面到用户的Web浏览器中。当使用这种方法的时候保持URL较短以免隐藏问题。

提示 你可以使用一个较短的视图名而不是32个字符的视图ID，使用较短的关键值（比如Notes ID或独特的关键词）代替文档的独特的ID，来缩短文档的URL。非常重要是在DNS环境下Notes ID可能无效。另外一些缩短URL的有趣的方法是：

- 如果数据库名太长或在一个子目录中，使用16进制数代替ID。
- 不使用视图名。
- 使用目录映射，例如，把/workflow/mail/mail.nsf/映射为/mailkey/*。
- 丢弃?openDocument。

该技巧由DOVID GROSS提供

基于Web的工作流应用程序

Navigator3.04 TM	Navigator4.05 TM	Domino4.6.1 TM
Explorer3.0 TM	Explorer4.01 TM	Domino5.0 TM

Bug Report数据库（Bugs.nsf）是一个简单的工作流应用程序的例子，使如下方案自动运行：

- 1) 一个匿名的用户向 Acme汇报bug。
- 2) 一个 Acme管理员将该 bug指定给某个软件工程师。
- 3) 软件工程师针对该 bug提出解决办法报告。
- 4) 工程管理人员或者赞成该报告并把它发送到 Acme的Web站点的公共区，或者否定它并添加注释后将它返回软件工程师。

图6-11显示了一个当 Bug Report被提出但还未指定软件工程师的例子。当管理员打开这个文档的时候，他只能编辑文档中与他的特定任务有关的那部分内容：把该 bug指定给某个工程师。

The screenshot shows a Netscape browser window titled 'Bug Reports - Netscape'. The address bar shows 'http://128.128.1.128/rmkelleher/bugs.nsf/Entry?OpenForm'. The main content area displays a 'Bug Report' form with the following fields:

- From:** Anonymous
- Subject:** Install program formats hard disk
- Date:** 2000-03-02
- Description:** Enter a description. Be as specific as possible. The install program for this software formats your hard disk before it crashes.
- Workaround:** Enter a workaround, if available.
- Assignment:** Who should work on this document? (If you enter a name here, the status will change to) Not Assigned Yet

图6-11 新的错误提交报告

一旦管理员将 bug指定给某个工程师并提交表单的时候，SaveEntry代理（一个WebQuery Agent代理）开始执行，这个代理：

- 提升文档状态。
- 根据其状态限制对文档的访问。
- 当一个bug报告被赋值或重新赋值的时候通过 Email提示工程师。
- 使用时间标签，记录每次文档存储的状态和储存人。

下面的代码代表了应用程序的内容，注意当邮件便笺产生的时候，代理把接受者的 Notes用户名填写在 SendTo域中。如果用户有 Internet邮箱，在共用地址簿中的用户的个人的文档 Notes的用户名（例如 Don Johnson/Acme）可以映射到 Internet的用户名（例如 djohnson@acme.com）。当代理调用文档的 Send方法，邮件路由器检查用户名的映射并在必要的时候列

出SMTP MTA帮助。这对于开发者来说情况变得更为简单，我们不需要关心这些细节性问题，只需要表达希望把邮件送到 Acme的Don Johnson，然后，它就会自己运行了。

```
' Update status
  Select Case doc.Status(0)

' If entry has just been Assigned, send assignee an e-mail
  Case "New":
    If ( doc.AssignedTo(0) = "Not Assigned Yet" ) Then
      doc.CurrentDocEditors = "[Assigner]"
    Else
      doc.Status = "Assigned"
      doc.CurrentDocEditors = "[Editor]"
      assignmenttype = "assigned"
      Gosub Notify
    End If

' If entry has been edited and editor is finished with it, change
' status so doc can be reviewed and approved
  Case "Assigned":
    If ( doc.DoneEditing(0) = "Yes" ) Then
      doc.Status = "Edited"
      doc.CurrentDocEditors = "[Approver]"
    End If

' If approver approves and saves entry, change status to Approved;
' otherwise, reassign entry and send memo
  Case "Edited":
    If ( doc.Approved(0) = "Yes" ) Then
      doc.Status = "Approved"
      doc.CurrentDocEditors = "[Manager]"
      doc.CurrentDocReaders = ""
    Else
      doc.Status = "Assigned"
      doc.CurrentDocEditors = doc.AssignedTo(0)
      assignmenttype = "re-assigned"
      Gosub Notify
    End If
  Case Else:
    doc.Status = "Unknown"
  End Select

' Print different output based on current status
  If ( doc.Status(0) = "New" ) Then
    Print "<H1>Thank you for your entry.</H1>" & _
      "Due to the volume of entries we receive each day, " & _
      "we cannot respond to every entry. " & _
      "Check back within the next few days to see if your " & _
```

```

    "entry has a response." & _
    "<BR><BR><A HREF="/" & doc.DbName(0) & "/">Back</A>"
Else
    Print "<H1>Changes noted.</H1>" & _
    "<BR><BR><A HREF="/" & doc.DbName(0) & "/">Back</A>"
End If

' Update workflow history
lastaction$ = "Saved on " & Today & " by " & _
doc.CurrentUserCN(0) & " with Status of " & doc.Status(0)
Call history.AppendToTextList( lastaction$ )

Exit Sub

ErrorHandler:
Print "<h1>Error</h1>" & Str(Err) & ": " & Error$
Exit Sub

Notify:
Set memo = New NotesDocument( db )
memo.Form = "Memo"
memo.SendTo = doc.AssignedTo(0)
memo.Subject = "Work assignment"
Set msg = New NotesRichTextItem( memo, "Body" )
' servername$ = doc.SERVER_NAME(0)
servername$ = doc.HTTP_HOST(0)
dbname$ = doc.DbName(0)
Call msg.AppendText( "The following item has been " & _
assignmenttype & " to you:" )
Call msg.AddNewline( 2 )
key$ = "NT" & Right("00000000" & doc.NoteID, 8)
docurl$ = "http://" & servername$ & "/" & dbname$ & _
"/" & "ByNoteID/" & key$ & "?OpenDocument&Login"
Call msg.AppendText( docurl$ )
Call memo.Send( False )
Return
End Sub

```

提示 在Notify程序中，注意以下一行

```
servername$=doc.SERVER_NAME(0)
```

已经被注释掉而改为

```
servername$=doc.HTTP_HOST(0)
```

参数HTTP_HOST优于SERVER_NAME。在HTTP/1.1中，HTTP_HOST返回实际的主机名，而SERVER_NAME仅仅返回从NAB服务器文档中返回的主机名。只有在服务器使用了多个IP地址和主机名的时候，或者服务器设置错误的时候上面的两个参数才有差异。

图6-12显示了一个经过 Internet 发送的邮件消息被 Netscape Messenger 收到的图例。注意在这个邮件消息的 URL 以 ?OpenDocument&Login 结尾。除了完成的文档以外，Web 用户在没有注册之前不能在 Bug Report 数据库中访问任何文档。&Login 参数触发注册提示，允许被指定者注册并立即开始在浏览器中编辑一个文档。



图6-12 在Domino工作流程应用程序中的 Internet mail

6.8 在LotusScript代理中使用小应用程序

与大部分人想像的情况不同，在 R4.5 版本中就可以创建 Java 小应用程序与 Domino 服务器上的数据库互相动态地进行影响。技巧就是使用 URLConnection，如果你开发的应用程序将可能被安装在早期的 Domino 服务器中，这个技巧非常方便。

使用 Java URLConnection，可以通过 GET 和 POST 方法把数据传送到 CGI 程序中。GET 和 POST 是把数据传送到 CGI 程序的两种不同的方法。

GET 方法是从一个 HTML 表单中提交数据的原始方法，但是通常已经被效果更好的 POST 方法代替。GET 方法把数据作为 URL 的一部分进行传输，例如，URL

`http://altavista.digital.com/cgi-bin/query?what=Web&q= " Lotus+Domino "`

把下列数据传输到查询过程中

参 数	值
what	Web
q	Lotus Domino

Web 表单使用 POST 方法，例如，当你提交 Web 文档的时候 Domino 表单使用 POST 方法传输信息到服务器上。POST 方法优于 GET 方法是多方面的，不能使用 GET 方法传输大量的数据，

GET方法把用户的表单中的内容保留在他们的历史文件中。然而，当你创建一个假托为从表单中产生的HTML链接的时候GET方法表现仍然良好，例如上面的初始化 Lotus Domino搜索的AltaVista例子。

与CGI程序一样，Domino代理可以访问带有参数的命令行或邮寄的输入表单中的数据，并根据数据的不同产生不同的响应。命令行的参数可以通过访问CGI环境变量QUERY_STRING得到（具体细节可以翻阅名为A QUERY_STRING代理的一节）。另外一个方法是打开不包含命令行参数的代理的URL链接，并POST数据。

当然，如果你使用的是R5版本的服务器，使用CORBA和Notes Java类更为简单直接。但是如果你负责咨询工作，有时不得不和一些并未在每台计算机上都安装了最新的软件的系统打交道。

Job Finder数据库

Navigator3.04 o Navigator4.05 TM Domino4.6.1 TM

Explorer3.0 o Explorer4.01 TM Domino5.0 TM

Job Finder例子数据库（JobFinder.nsf）包含POST和GET方法的例子。在使用这些例子以前编辑applet代码（JobFinder.java）并将之拷贝在Domino HTML目录下。现在，当你使用浏览器打开数据库并连接到Search Job Posting中的时候，如图6-13显示了该applet。

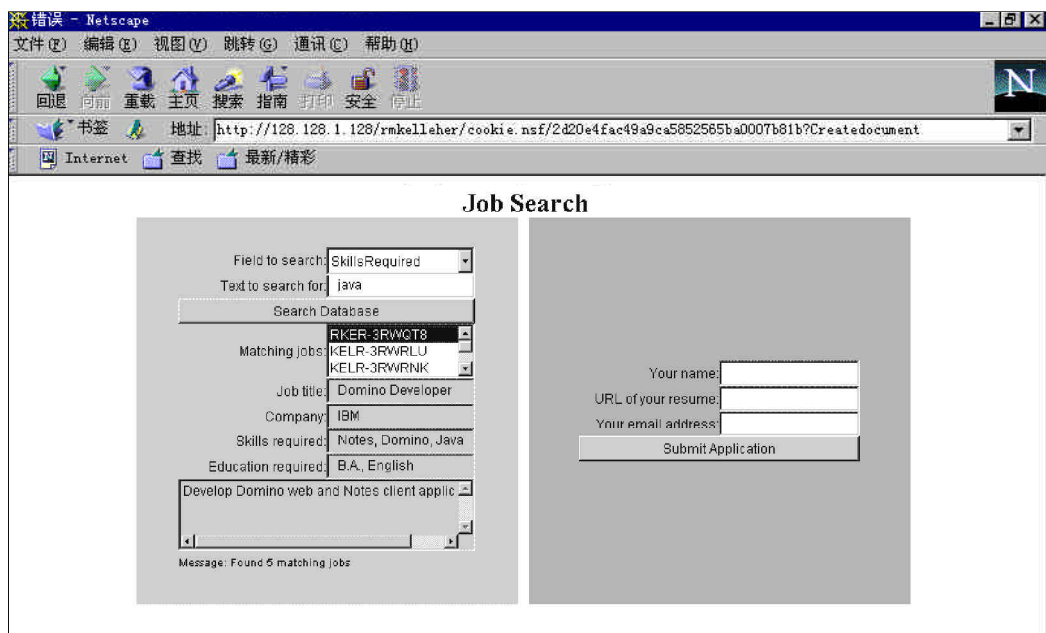


图6-13 JobFinder小应用程序使用URLConnections与LotusScript代理连接

为了寻找一个工作，选择一个域并输入搜索文本。在图6-13中用户已经开始搜索 Skills Required域中包含“java”的工作。用户可以双击某个查找到的工作ID以便显示详细信息。这部分小应用程序使用GET方法通过Find Job代理查询数据库。

为了申请一份工作，选择一个 ID 并在小应用程序的右边填写一个申请表单，并选择提交按钮，这部分小应用程序使用 POST 方法通过小应用程序代理向数据库中添加一个申请。

当编写一个小应用程序并把它连接到 URL 中时，注意，为了安全方面的原因，许多浏览器只允许小应用程序连接到自己的主机上。验证小应用程序可能要求例外的权限。例如连接到其他主机上的权利，这种权利用户可以拒绝。对于我们来说，这不是一个问题，因为 JobFinder 小应用程序只连接到自己所驻留的 Domino 服务器上。

当编写连接到 URL 的 Java 应用程序的时候，需要了解是否受防火墙的保护。如果确实是这样的话，当运行应用程序的时候可以在命令行设置代理。例如，下面的命令通知 JAVA.EXE 在访问 Internet 之前必须通过防火墙：

```
java -DproxySet=true -DproxyHost=MyFirewall ClassName
```

当你编写一个小应用程序，不必担心设置代理的问题，小应用程序由浏览器控制，它（假设你能够正确设置的话）了解什么是防火墙。

1. 工作过程

下面是这个例子的工作过程，Java 小应用程序向用户显示了一个查询表单，根据用户的输入，它通过 Find Job 代理提交申请到 Java 小应用程序。Find Job 代理解释 Java 小应用程序的请求，执行查询，并把输出送回浏览器，在此，仍然输出到一个小应用程序。这个小应用程序得到回复，包含每个发现匹配的工作的简短描述和每个匹配文档的 ID，这个小应用程序以一个漂亮的格式提供一个工作列表给用户，此时用户可以了解每个工作的细节或使用列表来填写申请并提交表单。当这个表单被提交的时候，小应用程序发送另外一个申请，这次发送到 Domino 的 Apply 代理，与 Find Job 代理相同。Apply 代理解释小应用程序的请求，得到细节并将结果返回浏览器（还是一个小应用程序，但是作为过程的一部分它还要在 Domino 服务器中创建一个工作申请文档。小应用程序解释结果，包括确认提交的表单，并向用户显示这个确认信息。

在小应用程序中的不同方法运行每个我曾提到的小应用程序功能：

- search() 方法把原来的搜索申请传递到 Find Job 代理。
- submit() 方法把工作申请信息传递到 Apply 代理。
- parse() 方法解释和管理由每个代理返回的信息。

2. search() 方法

当你在 Job Finder applet 中点击 Search Database 按钮，触发小应用程序的 search() 方法，这个方法连接到一个包含要搜索什么信息的 URL。在这个例子中图 6-13 的 URL 可能类似于

```
http://server/db/Find+Job?OpenAgent&Field=SkillsRequired&Value=java
```

search() 方法如下，注意使用了 Buffered Reader 对象以便一次读入一行信息。BufferedReader 类是在 JDK 1.1 中首次引入：

```
void search() {  
    String inputLine, theURL, theField, theValue;  
    URL url;
```

```
URLConnection conn;
BufferedReader br;
jobs.clear();
jobIDList.removeAll();
jobTitleField.setText("");
companyField.setText("");
skillsField.setText("");
edField.setText("");
descArea.setText("");
int started = 0;

try {
    theField = fieldChoice.getSelectedItemAt();
    theValue = valueField.getText().trim().replace
);
    theURL = "http://" + host + "/" + db + "/";
    theURL += "FindJob?OpenAgent&Field=" + theField;
    theURL += "&Value=" + theValue;
    showMessage(theURL);
    url = new URL(theURL);
    conn = url.openConnection();
    conn.setDoInput(true);
    conn.setUseCaches(false);
    conn.connect();
    showMessage("Reading data");
    br = new BufferedReader(new
        InputStreamReader(conn.getInputStream()));
    while ((inputLine = br.readLine()) != null) {
        System.out.println(inputLine);
        if (started == 1)
            parse(inputLine);
        if (inputLine.equals("Start"))
            started = 1;
    }
    br.close();
} catch (Exception e) {
    showStatus(e.getMessage());
    e.printStackTrace();
}
```

Search()方法输出到System.out中，为了查看这个输出，打开浏览器的Java控制台，图6-14显示了在Java控制台的search()方法的输出，如果你的浏览器指向另外一个搜索URL，显示同样的输出，如果查看页面资源，将发现各行已经被新的行字符分割。

当你编写一个连接到LotusScript代理的Java代理的时候，只要当applet从输入流中获得每行都打印到System.out中，浏览器的Java控制台可以辅助调试applet和代理。



图6-14 Java控制台可以用来调试在一起工作的 applet和LotusScript代理

3. Submit()方法

当你选择 submit application 按钮的时候，触发 applet 的 submit() 方法，这个方法打开到 Apply 代理的 URL 的链接，向链接中写入数据并获得响应。Submit() 方法如下所示，注意这个方法在把它写入输出流之前对数据进行编译：

```
void submit() {
    String jobID, name, resumeURL, email,
        theURL, content, inputLine;
    URL url;
    URLConnection conn;
    DataOutputStream output;
    BufferedReader br;
    int started = 0;
    try {
        jobID = jobIDList.getSelectedItem();
        if (jobID == null) {
            showMessage("No job is selected");
            return;
        }
        name = nameField.getText();
        resumeURL = urlField.getText();
        email = emailField.getText();
        theURL = "http://" + host + "/" + db + "/Apply?OpenAgent";
        showMessage(theURL);
        url = new URL(theURL);
```

```

// Send POST data
conn = url.openConnection();
conn.setDoInput(true);
conn.setDoOutput(true);
conn.setUseCaches(false);
conn.setRequestProperty("Content-Type", "application/
x-www-form-urlencoded");
content =
    "JobID=" + URLEncoder.encode(jobID) +
    "&Name=" + URLEncoder.encode(name) +
    "&URL=" + URLEncoder.encode(resumeURL) +
    "&Email=" + URLEncoder.encode(email);
output = new DataOutputStream(conn.getOutputStream());
output.writeBytes(content);
output.flush();
output.close();

// Read response data
br = new BufferedReader(new
InputStreamReader (conn.getInputStream()));
while ((inputLine=br.readLine()) != null) {
    System.out.println(inputLine);
    if (started == 1)
        parse(inputLine);
    if (inputLine.equals("Start"))
        started = 1;
}
br.close();
} catch (Exception e) {
    showStatus(e.getMessage());
    e.printStackTrace();
}
}

```

它还可以以相同的方式通过连接到 Domino CreateDocument URL而不是LotusScript 代理URL来提交一个文档。在第9章中包含一个这样的例子，如果你使用代理创建一个文档，则这个代理由于要履行原来需要 WebQuerySave代理完成的工作而负担双倍的责任。

4. Parse()方法

当applet从URLConnection中读入数据的时候，它使用 parse()方法确定什么信息是重要的，以便根据这些信息进行反应。如果行中包含关于工作位置的信息，它将创建或更新一个 Job对象。如果行中包含错误消息或提示消息，它就向用户显示消息。Parse()方法显示如下，这个方法依赖String类的startsWith()方法决定它需要处理的是哪种类型的信息。

```

void parse(String inputLine) {
    if (inputLine.startsWith("Error:") || inputLine.startsWith("Message:"))

```

```

        showMessage(inputLine);
    } else if (inputLine.startsWith("JobID=")) {
        job = new Job();
        job.JobID = inputLine.substring(6);
        jobIDList.addItem(job.JobID);
    } else if (inputLine.startsWith("JobTitle=")) {
        job.JobTitle = inputLine.substring(9);
    } else if (inputLine.startsWith("Company=")) {
        job.Company = inputLine.substring(8);
    } else if (inputLine.startsWith("SkillsRequired=")) {
        job.SkillsRequired = inputLine.substring(15);
    } else if (inputLine.startsWith("EducationRequired=")) {
        job.EducationRequired = inputLine.substring(18);
    } else if (inputLine.startsWith("Description=")) {
        job.Description = inputLine.substring(12);
        jobs.put(job.JobID, job);
    }
}

```

5. Find Job代理

Find Job代理通过使用Replace和GetParameter函数与文档的QUERY_STRING域得到每个域的值和参数值。QUERY_STRING域代表包含参数的QUERY_STRING环境变量的值。这个代理通过Replace和GetParameter函数分析查询字符串得到每个参数的值。这些函数由名为Web Function的脚本库定义。然后这个代理搜索数据库并显示关于匹配文档的信息。代理的输出除了被显示在浏览器中，还会被小应用程序读入并进行分析。代理的代码显示如下，代理使用applet希望的格式打印所有的信息。例如，一行的开始是 " Error " 表明是一个错误信息。

```

Sub Initialize
    ' This agent is run via a Domino URL. It returns information
    ' about job postings meeting the specified search criteria
    ' (field contains value).

    On Error Goto ErrorHandler

    Dim session As New NotesSession
    Dim doc As NotesDocument
    Dim dateTime As NotesDateTime
    Dim queryString As String
    Dim fieldName As String
    Dim fieldValue As String
    Dim db As NotesDatabase
    Dim dc As NotesDocumentCollection
    Dim searchDoc As NotesDocument

```



```
' Signal that the HTML header and initial tags have ended and the
' meaningful data is coming next.
Print "Start"
Set session = New NotesSession()
Set doc = session.DocumentContext
Set dateTime = New NotesDateTime("01/01/1998")

' Get URL parameters from the QUERY_STRING environment variable.
queryString = doc.QUERY_STRING(0)
fieldName = ReplaceIt(GetParameter("Field", queryString), "~", " ")
fieldValue = ReplaceIt(GetParameter("Value", queryString), "~", " ")

' Search the database.
searchFormula = "@Contains(@Uppercase(" & fieldName &
");
@Uppercase("'" & fieldValue & "'"))"
Set db = session.CurrentDatabase
Set dc = db.Search(searchFormula, dateTime, 0)

' If no matches are found, let the applet know. The "Message:" and
' "Error:" prefixes flag text as important info that needs to be
' displayed to the user.
If (dc.Count = 0) Then
    Print "Message: No matching documents found"
    Exit Sub
End If

' If matching documents are found, print the info using
' a "protocol" that the applet can understand.
Print "Message: Found " & dc.Count & " matching jobs"
Set searchDoc = dc.GetFirstDocument()
While Not (searchDoc Is Nothing)
    Print ""
    Print "JobID=" & searchDoc.JobID(0)
    Print "JobTitle=" & searchDoc.JobTitle(0)
    Print "Company=" & searchDoc.Company(0)
    Print "SkillsRequired=" & searchDoc.SkillsRequired(0)
    Print "EducationRequired=" & searchDoc.EducationRequired(0)
    Print "Description=" & searchDoc.Description(0)
    Set searchDoc = dc.GetNextDocument(searchDoc)
Wend
Exit Sub

ErrorHandler:
Print "Error: " & Str(Err) " - " & Error$
Exit Sub

End Sub
```

6. Apply代理

Apply代理从REQUEST_CONTENT环境变量中获得数据并使用它填写一个新的表单，并将其保存在数据库中。这个代理的代码如下所示，代理使用了 Message:字符串通知用户表单提交是否成功。

```
Sub Initialize
    On Error Goto ErrorHandler
    Print "Start"
    Dim session As New NotesSession
    Dim doc As NotesDocument
    Dim db As NotesDatabase
    Dim content As String, theID As String,
        theName As String, theURL As String, theEmail As String

    ' Collect POSTed data
    Set doc = session.DocumentContext
    content = doc.REQUEST_CONTENT(0)
    theID = Unencode(GetParameter("JobID", content))
    theName = Unencode(GetParameter("Name", content))
    theURL = Unencode(GetParameter("URL", content))
    theEmail = Unencode(GetParameter("Email", content))

    ' Save doc as a job application
    Set db = session.CurrentDatabase
    doc.Form = "Job Application"
    doc.JobID = theID
    doc.Name = theName
    doc.ResumeURL = theURL
    doc.Email = theEmail
    Call doc.Save(True, True)
    Print "Message: Application submitted successfully"
    Exit Sub

ErrorHandler:
    Print "Error:" & Str(Err) & " - " & Error$
    Exit Sub
End Sub
```

记住REQUEST_CONTENT数据被 applet使用 URLEncoder类进行了编辑。Encoder函数使用加号代替了空格，使用百分号（%）与十六进制数代替特殊字符。Uncode函数则将该过程逆方向进行。这个函数在 Web Functions脚本库中定义。

7. Job Application表单

当提交工作申请的时候，它包含申请人姓名，履历 URL（我们假设申请者拥有一个 Web 主页），邮件地址。图6-15显示了一个在Web浏览器中打开的工作申请。

工作申请表使用 HTML通用文本把邮件地址转换为一个 mail to:链接。而履历URL不需要进行同样的处理，Domino自动显示http:URL作为HTML链接。



图6-15 提交工作申请包含到用户履历和 email地址的连接

参考信息

为了得到更多关于使用 Java连接到 URL的信息，阅读 JavaSoft的“Reading from and Writing to a URLConnection”，访问地址：

<http://java.sun.com/docs/books/tutorial/networking/urls/readingWriting.html>

本章小结

LotusScript在开发 Domino Web应用程序中仍然是一个有用的工具。在 Web中运行 LotusScript代理的三种方法是打开代理的 URL，WebQueryOpen事件和 WebQuerySave事件。一个基于Web的LotusScript代理可以把HTML文本输出到浏览器中或将其指向其他 URL。基于Web的LotusScript代理可以用来查询关系型数据库，分析 cookie信息，并在工作流应用程序中生成email。在R4.5版本中LotusScript代理的URL就可以被Java applet用来从Notes数据库中读出或写入数据。