

第2章 使用表单

2.1 关于表单

表单是Notes 应用的基础,它们决定了数据在输入输出过程中的显示方式;在一定程度上,也决定了在数据库中信息是如何存储的。如果没有设计漂亮的表单,就不会有精美的 Domino 应用。

如果你熟悉HTML的表单,开始时你可能会被 Lotus Notes 的表单弄糊涂。和HTML表单一样,Notes的表单被用来从用户那儿获取信息(通常是通过可编辑的域)。和HTML表单不一样的是,Notes的表单也被用来显示已经获取的信息。这些信息存储在称为文档的容器之中。当你打开一个Notes表单时,填写完信息,选择保存。此时,在数据库中保存一个新文档。这个文档中包含了一个Form域,它的值确认用来创建文档的表单名,当你打开文档时,这个表单被用来显示信息。文档是数据的一部分,表单是数据库设计的一部分。这和一个数据库中的记录(称为数据)与表(称为基础组织)二者之间的关系相类似。

在HTML表单和Notes表单中,另一个不同的地方是访问的模式。在默认情况下,当你打开一个Notes文档时,它处于只读模式;读者所看到的都是在一个字处理器中创建的文档,不包含任何文本域或下拉列表。如果你将此文档改成编辑模式,则它有着明显的不同:可编辑文本域,下拉列表,以及其他元素都变得可见了,好像是在一个HTML的表单之中。然而,你所见到的并非是一个表单,确切地说,你所看到的文档,是通过一个表单显示出来的。

表单和Web

当你想通过Web来创建一个新文档时,你通过打开表单 Open Form URL 命令打开一个适当的表单。例如,下面的URL在Jobe数据库中打开了Job Application的表单。

<http://server/Jobs.nsf/Job+Application?OpenForm>

Domino 将Notes表单转化成一个HTML表单以响应你的请求。当然,Notes客户端应用还可以使用表单的没有HTML等价的另外一些特征,这些特征并不能在Domino的Web应用中工作。

你可以通过提交表单来保存新文档。提交表单时会触发表单事件(通过一个 Create Document URL对象)。继而,Domino使用在Create Document URL中指定的表单将信息从HTML表单传输过来并保存在数据中。

你不必考虑一个Notes表单是否适合Web。在你的Notes表单或文档中使用的大部分格式都能被Domino 服务器翻译成HTML语言,例如:左对齐、右对齐、居中;粗体、斜体、下划线和背景色;表;等等。而这些,将开发者和用户从费时又单调乏味的HTML标识符的输入中释放出来。

一旦你实践着创建Notes表单和文档，并在某个Web浏览器中测试它们时，你开始意识到，从Notes 客户端上丰富华丽的设计转换至普通而又略显老旧的 HTML语言的过程中，你失去了很多，而这些都是Domino服务器所产生的。关于这一点，你有两种选择：

(1) 放弃你的应用并责备那些高高在上 Iris公司的伙计们，他们为何不按你的思想来进行设计。

(2) 继续往下读完本书，并根据所学按照你的思想来创建应用。当熟悉这些非常有用的工具后，比如，公式语言，HTML规则，Cascading Style Sheets和JavaScript 等，你将会意识到通过这些Web应用你正在不断获取经验。此外，即使加上这些努力所花时间，你仍将比使用其他的工具来开发应用程序花费更少的时间。

2.2 在表单中使用公式

当LotusScript语言被引入到Notes的R4版本中时，许多的Notes开发者开始认为Notes的公式语言已经过时了，确实在某些方面，公式语言不再合适，比如循环，或者长而复杂的连续文件。但是，在许多场合，公式语言比 LotusScript语言更可取。例如，当你使用字符串和数组，计算域值，填充例表框等。

许多开发者并没有意识到短短的几行 Notes公式语言能够完成多少工作。例如，有两个字符串，ListA 和ListB，你想创建第三个列表，它包含 ListA的所有值，但并不包含 ListB中的值；换句话说，就是ListA 减去ListB。在LotusScript中（只有少量的函数可以处理字符和数组，非常苦恼），必须循环列表，一次比较一个值：

```
Redim ListC(0) As String
Forall s In ListA
    If Not (Contains(ListB, s)) Then
        Redim Preserve ListC(Ubound(ListC) + 1)
        ListC(Ubound(ListC)) = s
    End If
End Forall

Function Contains(sArray() As String, s As String) As Variant
    Forall value In sArray
        If (value = s) Then
            Contains = True
            Exit Function
        End If
    End Forall
    Contains = False
End Function
```

使用Notes公式语言，在一行中即可完成相同的事：

```
ListC := @Trim(@Replace(ListA; ListB; ""));
```

这一部分提供了在表单中使用公式的两个普通的例子，本书还包含有很多类似的例子。看起来似乎想说明Java小程序、小服务程序、LotusScript代理、或者JavaScript技术，但我仍

然以使用公式作为结束。如果你对 Domino是个新手，并想了解怎样更好地学习 Domino开发，我强烈地建议你从Notes公式语言开始。

2.2.1 数组和多值域

一旦你熟悉了Notes公式语言的字符和数组处理函数，就会在 Notes表单中使用多值域并利用一些非常有用的技巧。多值域是值为数组的一个域。可以配置一个多值域分行显示域值，或者通过空格、逗号或分号来分隔。在 Notes公式语言中，数组的值被冒号分隔，例如：

```
List1 := "aaa" : "bbb" : "ccc";  
List2 := 55 : 89 : 243 : 7;
```

在为本书编写例子时，我常常发现，多值的文本域结合 Notes的公式语言非常有用。不幸的是，Domino有一些限制，妨碍了它的使用，其中：

- 在一个文本域中，最多只能存储 15kbyte的数据。
- @DbColumn和@DbLookup的返回值最多只有 64kbyte。

非常遗憾，如果想处理大批量的数据，这些限制严重地影响了效果。希望这些缺点在以后的版本中会改进。想要了解详细的限制情况表（表单命名的最长的长度、在一个表单中域的最多数目，等等），请参看Notes的帮助数据库的“关于Notes的限制”主题。

2.2.2 公式的类型

在表单中有许多不同的使用公式的方法，包括：

- 域公式。
- 段落隐藏公式。
- 计算域文本。
- 特殊域。
- HTML属性。
- 表单操作公式。
- 热点。

1. 域公式

在表单中使用最多的公式是为计算域赋值，你可以创建下列类型的域公式：

- Value:为计算域赋值。
- Default value:为可编辑域提供缺省值。
- Input translation:转换已存在的值。
- Input validation:校验域值。
- HTML属性：为域指定HTML属性（比如“size=35”）。

2. 隐藏公式

另一类被大量使用的公式是隐藏公式。表单中每一类元素都有一个隐藏属性选项卡。这些属性确定元素（或者更确切地说，元素所在的段落）什么时候可见什么时候不可见，可使

用公式并由其返回值为 True（非零）或 False（零）来确定。

指示 使用隐藏公式将对整个段落起作用，不能够在一个段落内选取一小片文字或一个图表去应用隐藏公式。在某些情况下，可以使用表格来绕过这些限制，因为在表格的同一行内，每一个单元格会被自动划分为一个段落。

3. 计算域文本

计算文本元素是一种返回值到页面上显示的公式，但更确切地说，它是一个热点，而不是一个域。这种区别比较重要。因为域仅仅只能被对数据库具有设计者以上权限的人创建，但任何用户都可以创建热点。使用计算文本，用户可以添加公式到他们的个人文档和其他文档中去；开发者也可以添加计算文本到表单和其他条目中去，而在其中域是不可见的（例如，数据库的帮助文档）。

4. 特殊域

为特殊域赋值时公式是非常有用的。表 2-1 显示了在 Web 表单中非常有用的特殊域。

表2-1 在Web表单中有用的域

域	描 述
\$\$HTML Head	包含在<HEAD>和</HEAD>标记之间的文本。在 R4.6 以后版本中可用 HTML 的 Head 属性
\$\$Navigator Body	在表单中嵌入的导航器名，在 R4.6 以后版本中可用“嵌入导航器”
\$\$Navigator Body-n	的功能
\$\$QueryOpenAgent	文档打开之前运行的代理名。代理产生的输出被忽略，而不是传到浏览器；在 R4.6 或更新的版本中，则是在表单的 Web Query Open 事件中编辑缺省的公式，使用代理名来替换<Your agent goes here>
\$\$Query Save Agent	文档递交时运行的代理名。代理在计算域刷新之后，文档被保存之前运行。在 R4.6 或更新版本中，则是在表单的 Web Query Save 事件中编辑缺省的公式，使用代理名来替换<Your agent goes here>
\$Readers	使得文档仅仅只能在该域中或者其他类型的“读者”域中指定的用户阅读。可以包括用户名，群组 and 角色（将角色名包括在方括号内），例如：“Joe Smith”：“Marketing”：“[Reviewer]”
\$\$Return	表单被递交时响应浏览者的文本；例如：“<h1>Thank you</h1>”；你也可以根据表单的递交情况通过返回一个包含在方括号内的 URL 传递给浏览器一个 URL 资源，例如：[http://www.erols.com/kelleher]
Save Options	决定一个文档是否使用该表单保存的值：取默认值 0 时不保存。当递交按钮被设计成激活某些操作而不是创建文档时，这一点非常有用。例如，搜索或激活一个代理
Send To	文档被邮寄时邮件接收人列表。相关的域包括 CopyTo, BlindCopyTo 和 DeliveryPriority
\$\$View Body	表单中想要嵌入的视图名，在 R4.6 以后版本中可用“嵌入视图”功能
\$\$View List	在数据库中显示视图列表，在 R4.6 以后版本中可用嵌入文件夹功能

5. HTML 属性

在设计表单时可以使用公式来指定表单的 HTML 的 Head 和 Body 属性。

HTML的Head属性是你想在<HEAD>和</HEAD>标记之间显示的文本。此区域通常被用来设置小程序和定义Java Script函数，等等。

HTML的Body属性是你想包含在<BODY>标记中的属性，包括：

Alink=color	活动链接的颜色
Background=url	背景图像的URL
Bgcolor=color	背景色
Link=color	未访问链接的颜色
Text=color	默认的文本颜色
Vlink=color	已访问链接的颜色

当body对象被打开或关闭时，可以在其中包含触发 JavaScript代码的命令：

```
onLoad5"doSomething();"
onUnload5"cleanup();"
```

为了详细说明HTML属性，可以使用公式，例如：

```
"onLoad5\"doSomething();\""
```

6. 表单操作公式和热点

在页面的顶端，可以创建表单操作，它在 Web的表单中显示为按钮的形式。当表单被翻译成HTML时，按钮被翻译成链接到URLs上的图像。

除了可以自己规定图形和文本外，热点和按钮相类似。

2.2.3 例子：一个组合框域

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

本例使用公式创建一个在 HTML中并不存在的元素：组合框。组合框是一个域，它提供一个值的列表，让你可以从中选择；它还提供一个文本域，如果你所要的值不在列表中时可以自行输入。在 Notes中，你可以轻松地创建一个组合框。只要创建一个关键字域，然后选中属性“允许不在列表中的值”即可。组合框的界面如图 2-1 所示。

在Web中复制这个功能并非如此直接，图 2-2 给出了Form Examples.nsf 数据库中Download表单的一部分，当递交一个 Download文档时，你必须从提供的列表选择一个分类，或者在列表下方的文本域中输入一个新分类。如果你使用新分类递交一个文档，此新分类被添加到列表中，以后的用户便可以选择它，如果你错误地选择了一个已有的分类又输入了一个新分类，Category域的值默认为使用你选择的列表中的分类。



图2-1 在Notes客户端创建组合框域很简单，但是在Web应用程序中怎样创建组合框域呢？

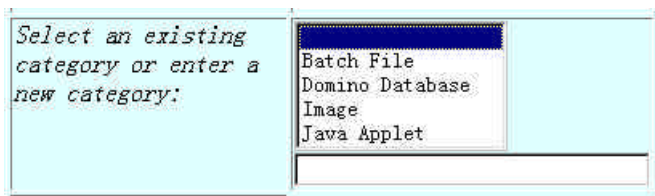


图2-2 这两个域联合起来构成组合框效果

本例用到三个域：ExistingCategory, NewCategory 和Category。这些域将在下面的三个部分描述。

1. Existing Category域

第一个域Existing Category是一个关键字域，它提供已有的分类列表以便你可以从中选择。这个域使用下列公式：

```
" " : @DbColumn("Notes":"NoCache"; "" ; "Downloads by Category"; 1)
```

@DbColumn返回DownLoads by Category 视图的第一列的所有值，并按照分类排序；也就是说，返回已有的分类列表。列表顶端的空格（" "）是允许该域的空值。

ExistingCategory 域的Input TransLation公式在递交表时将空格转换成空字符串：

```
@Trim(ExistingCategory)
```

Existing Category域被设置成仅当文档是新文档或该文档处于编辑状态时才可见。

2. New Category域

New Category域是一个普通的可编辑文本域，此域有一个 Input Validation 公式，用来检查和确认用户已指明一个分类，不论是在 ExistingCategory域中选择的值还是在 NewCategory 域中输入的值：

```
@If(ExistingCategory = "" : " " & NewCategory";  
    @Failure("Please select an existing category or enter a new category.");  
    @Success)
```

如果在递交该表单之前你忘了输入或选择一个分类，将会显示错误信息，并且新文档或已更新的文档不会被保存。

NewCategory域被设置成仅当文档是新文档或该文档正处于编辑之中时可见。

3. Category域

Category域是一个计算文本域，它使用下列公式：

```
@Trim(ExistingCategory : NewCategory)
```

此公式的作用是：

- 创建一个列表，其值包含 Existing Category和NewCategory两个域的值。
- 从列表中删除空值。

当使用此公式时，ExistingCategory或NewCategory域的值应为空，所以Category域的值总是其中的一个。

该域被设置成仅当文档以只读方式打开时才可见。

2.2.4 例子：\$\$Return域的公式

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

在数据库FormExamples.nsf的Download表单中，使用了一个特殊的域公式，决定用户递交表单后如何结束。在表单底部有一个叫 \$\$Return的计算域，它使用了如下的公式：

```
db := @ReplaceSubstring(@Subset(@DbName; -1); "\\\"; "/");
"[/\" db 1 "]"
```

此公式的返回值是Form Examples数据库中包含在方括号内的 URL资源。\$\$Return是一个特殊的域，它的值决定了用户递交了表单后将会看到什么。如果返回值是包含在方括号中间的URL，用户的浏览器将直接指向此 URL，否则，Domino仅仅向浏览器传输值，例如，你可以创建一个 \$\$Return的域，使用公式返回HTML标签，例如：

```
"<h1>Thank you!</h1>" 1
"<font size=\"12\">Thanks for your feedback. We'll get back to you
ASAP.</font>"
```

在Download表单中的 \$\$Return域公式中，路径和文件名通过计算来得到，因此，如果你将数据库移动或改名， \$\$Return域的值仍将指向正确的 URL。例如，如果你将此数据库移至 Domino数据目录下的 Misc子目录中，并更名为 MyDatabase.nsf，结果将会是 [/Misc/MyDatabase.nsf]。注意，http://prefix 和服务名被忽略了。通常，如果允许的话，忽略服务器名将是一个好主意，这样可使你能够将应用程序移动到其他的服务器，只需一个简单的斜杠 “/”（代表当前服务器）就行了。

2.3 在表单中使用Web元素

在Domino中提供了许多不同类型的设计元素，他们都非常有用。但是，当配合着使用这些元素时，常常变得更为有效。配合使用 Domino的设计元素的最常用的方法是在一个表单内嵌入多种元素。嵌入的元素包括视图、导航器、大纲和文件上载控件。

2.3.1 嵌入Web元素

在表单中和其他设计元素，例如导航器，一起配合使用的最简单的方法是直接嵌入元素。例如，你可以创建一个小的水平的导航器，并嵌入到每一个表单中，以帮助用户使用数据库。若使用这种方法，通过一个OpenForm URL打开表单来显示此导航器（或其他设计元素）。

2.3.2 创建特殊用途的模板表单

你可以创建一些特殊的表单，不但可以使用 OpenForm的URL，也可以使其他URLs，例如OpenView或OpenNavigator 来打开这个表单。

1. \$\$ ViewTemplate

你可以让Domino总是使用一个表单来显示一个特殊的视图。为此，给此视图命名为：

```
$$ViewTemplate forviewname
```

在此，viewname是视图的名字或别名。例如，一个用来显示 All by Data 视图的表单可以命名为\$\$ViewTemplate for All by Date。

当你创建这种包含有 \$\$ViewTemplate的类型的表单时，你必须在表单中嵌入视图本身（或者，你也可以包含一个名为 \$\$ViewBody的域来替换嵌入的视图）。

2. \$\$ViewTemplateDefault

假设你的Domino数据库包含20个不同的视图，而你想让他们具有一致的风格，你可以创建一个\$\$ViewTemplate表单，并做19份拷贝，每一个对应一个视图，但这会非常费时。相反，你可以创建一个简单的表单，用来显示数据库内的任何视图，仅仅只须给表单命名为\$\$ViewTemplateDefault。

当你创建这种\$\$ViewTemplate表单时，嵌入一个视图元素（或 \$\$ViewBody域），而不必指定一个特殊的视图。

3. \$\$NavigatorTemplate

你可以让Domino总是使用一个表单来显示一个特殊的导航器。为此，给表单命名为：

```
$$NavigatorTemplate fornavname
```

在此，navname是导航器的名称。例如，用来显示 Main Menu导航器的表单可以命名为\$\$ViewTemplate for Main Menu。

当你创建这种\$\$NavigatorTemplate表单时，必须在表单中嵌入导航器本身（或者，也可以创建一个名为\$\$Navigator Body的域来替代嵌入的导航器）。

4. \$\$NavigatorTemplateDefault

你可以创建一个简单的表单，用来在数据库中显示任何导航器，只需给表单命名为\$\$NavigatorTemplateDefault。当你创建这类\$\$NavigatorTemplate表单时，嵌入一个导航器元素（或\$\$NavigatorBody域）。而不必指定一个特殊的导航器。

5. 联合表单

你可以使用联合的表单名或别名来创建多重目的的表单。例如，如果你给表单命名为：

```
$$ViewTemplateDefault | $$NavigatorTemplateDefault
```

缺省地，使用同一个表单来显示数据库中的任何视图或导航器。

你也可以在同一个简单的表单中联合嵌入多重元素。一个好的例子是 \$\$ViewTemplate Default表单，它既包括一个嵌入的视图元素（一个普通的视图或一个空 \$\$ViewBody域），也包括一个嵌入的导航器（一个特殊的导航器）。

2.3.3 例子：在\$\$ViewTemplateDefault表单中嵌入导航器

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

Explorer3.0™ Explorer4.01™ Domino5.0™

使用导航器的一个比较有效的方法是在 \$\$ViewTemplateDefault表单中嵌入，这样，当视图显示时，导航器也随之显示。

图2-3显示了FormExamples.nsf数据库中的Downloads by Category 视图。当你打开此视图时，它的内容自动嵌入到表单中，此表单包含有一个嵌入的导航器。此表单被命名为 \$\$ViewTemplatedefault，使用了一个一行两列的表格，将嵌入的导航器排列到页面的左边，而视图的内容放到了右边。

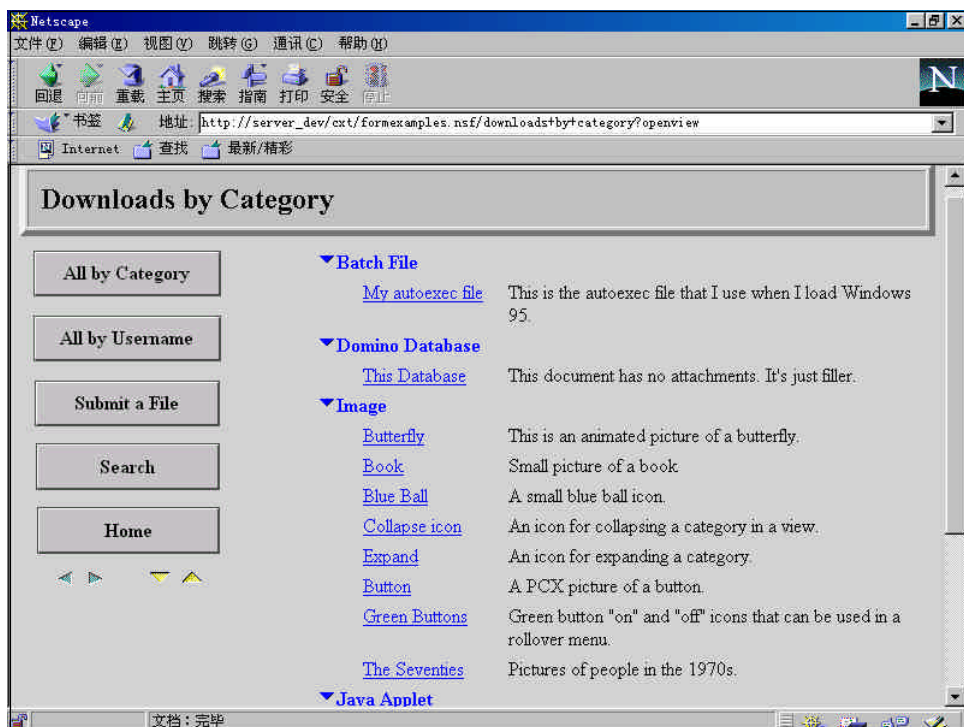


图2-3 每次打开视图时页面左边的导航器显示

这样做有什么好处？试着单击“ All by Username ”按钮，现在，在页面的右边打开了一个不同的视图，但嵌入的导航器仍然没有改变。如果使用视图的操作按钮来代替一个导航器，必须将每个按钮创建两次（每个视图一次），或者从一个视图拷贝并粘贴到另外一个。在一个简单的导航器内包含所有的视图导航器按钮，给自己节约了时间。不仅如此，这种简单的导航器比页面顶端的一系列操作按钮看起来要好得多。

2.3.4 例子：动态嵌入式导航器

Navigator3.04™

Navigator4.05™

Domino4.6.1™

Explorer3.0™

Explorer4.01™

Domino5.0™

本例在前面的例子的基础上更进一步。为了得到本例，在设计模式打开 Form Examples.nsf 数据库，将 \$\$ViewTemplatedefault 表单改名成其他的名字，然后，会发现命名为 Rename Me 的表单，将它改为 \$\$ViewTemplateDefault。这个表单和 \$\$ViewTemplateDefault 是一样的，除

了它使用一个公式决定哪一个导航器在页面的左边显示。

`@If (@UserName="Anonymous";"Anonymous";"Main")`

如果当前用户名是 Anonymous (也就是说, 当前用户并没有登录), 在页面的左边显示 Anonymous 导航器。除了没有 “ Submit a File ” 按钮以外, Anonymous 导航器和 Main 导航器是相似的。如果用户名并非 Anonymous (意思是他或她已经登录进来了), 显示 Main 导航器。

为了测试本例, 不登录打开 Downloads by Category 视图:

`http://sv/db/RMKelleher/FormExamples.nsf/DownLoads+by+Category?OpenView`

你可以看到 Amomymous 导航器; 然后, 通过附加登录的信息到视图的 URL 来登录:

`http://sv/db/RMKelleher/FormExamples.nsf/DownLoads+by+Category?OpenView&Login`

现在, 你可以看到 Main 导航器了。简单, 是吗? 你可以使用相同的公式来动态地显示视图和其他的嵌入元素。下面是几种可能的应用:

- 根据用户的访问权限显示不同的视图。
- 配置一个视图显示一个 Java 小程序, 其他的显示为 HTML, 使用 `@GetProfileField` 函数决定用户喜欢哪一种视图。
- 根据用户的角色显示不同的导航器。

2.4 在表单和表单对象中使用 HTML 属性

你可以在表单中为表单和表单中的每一个元素指定 HTML 属性。在 Domino 中这样做而不是编辑一个静态的 HTML 文件的好处是你可以使用 Notes 的公式语言来决定 HTML 属性。如果你想使表单的 HTML 根据用户或域值或其他元素的不同而改变的话该属性非常有用。表 2-2 显示了一些特殊的 HTML 属性的例子。

表2-2 HTML属性的实际例子

对 象	属 性	描 述
Text field	"Size=40"	设置文本域的宽度为 40 列
Keyword field(select)	"OnChange=\"doIt();\""	当域值改变时激活 JavaScript 的 doIt() 函数
Keyword field(radio)	"OnClick=\"doIt();\""	当 MOUSE 单击时激活 JavaScript 的 doIt() 函数
Keyword field(checkbox)	"Onclick=\"doIt();\""	当 MOUSE 单击时激活 JavaScript 的 doIt() 函数
Rich text field	"rows=3 cols=40"	设置文本区域为 3 行 40 列
Form	"onLoad=\"doIt();\""	当打开表单 (body) 时激活 JavaScript 函数 doIt()

除了 HTML Body 属性之外, 一个 Domino 表单也有一个特殊区域输入 HTML Head 属性, 使用这一部份来输入你想显示在 <HEAD> 和 </HEAD> 标记之间的 HTML。典型地, 这一部分被用来定义文档标题, 帧结构集和高级数据。

在 R5 以前的版本中, 创建帧结构集的惟一方法是使用 <HEAD> 部分来输入公式并返回 <FRAMESET> 的 HTML 标记; 仅管新方法确实更适合, 老的方法仍然可用。典型地, 你可以创建一个命名为 \$\$HTMLHead 的域, 并使用下列公式:

```
"<frameset cols=\"25%,75%\" border=0>" +
"<frame name=\"Frame1\" src=\"url1.htm\">" +
"<frame name=\"Frame2\" src=\"url2.htm\">" +
"<noframes>" +
"Click <a href=\"url3.htm\">here</a> to continue." +
"</noframes>" +
"</frameset>"
```

这段公式创建了两个帧：左边小的一个显示 url1.htm，右边大一点的显示 url2.htm。

提到这些，仅仅让你知道，如果从其他开发者那继承了一个老的 Domino数据库并且看到了像上面一样的公式时知道会发生什么。

在新的R5中，域也有特殊的 HTML，包括 Cascading Style Sheet (CSS) 信息，比如类和类型。

2.4.1 <META> 标记的例子

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

Explorer3.0™ Explorer4.01™ Domino5.0™

一个Web页面的<HEAD>部分的通常用法是在此设置<META>标记。<META>标记的不同属性提供关于文档的附加信息，例如：下面的<META>标记使得搜索引擎不索引本页：

```
<meta name="robots" content="noindex">
```

也可以使用<META>标记告知浏览器在若干秒以后应该刷新页面。当刷新页面时，你可以让浏览器打开一个不同的 URL，或者每隔几秒继续刷新当前页面。例如：假设你想将站点改为别的位置，则可以替换旧主页使它弹出提示信息：“We’ve moved to http://www.acme.com, click here or just wait for 10 seconds.” 10秒钟以后，可以将用户的浏览器改变到你的新点，此时，使用了如下的标记：

```
<meta http-equiv="refresh" content="10;url=http://www.acme.com">
```

如果指定的 URL 是相同的 URL，文档重复地刷新。数据库 FormExamples.nsf 中的 \$\$ViewTemplateDefault 表单说明了这个特点，此表单包含一个计算域 \$HTMLHead，它使用下列公式。

```
"<meta http-equiv=\"refresh\" content=\"60;url=PATH_INFO + \">"
```

上述公式使用 PATH_INFO 变量获得当前的 URL，因此，如果你打开 Downloads by Category 视图（总是在 \$\$View TemplateDefault 表单内显示），结果值如下：

```
<meta http-equiv="refresh" content="5;url=/FormExamples.nsf/Downloads+by+Category?OpenView">
```

在此，使用 \$\$ViewTemplate Default 表单打开的任何视图（也就是说，数据库中的任何视图）每60秒钟自动刷新一次。

不要疯狂地使用此特性，每次刷新页面，用户的显示就被中断，这非常烦人。此外，每次“client pull”都会要求服务器响应。对小部分用户来说，每隔几秒钟自动刷新一次视图非常好，但当你有成百上千的并发用户时，还使用此技术，将会是一种灾难。

2.4.2 例子：RTF文本域的属性

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

Explorer3.0™ Explorer4.01™ Domino5.0™

你可以通过在域的 HTML属性中指定行数和列数来控制 RTF文本域的大小。数据库 FormExamples.nsf中的Simple表单使用下列公式来为Body域指定属性：

```
"rows=10 cols=40 wrap=virtual"
```

下面是文本区域的HTML值：

```
<TEXTAREA NAME="Body" rows=10 cols=40 wrap=virtual></TEXTAREA>
```

这使得浏览器内的文本域有10行高，40列宽。Wrap属性(Netscape的特点)设置文本区域的单词范围是虚拟的（显示范围但并不传到服务器）而不是物理上的（显示并传输范围）或关闭它。虚拟是一个好方法，因为，对于用户来说，它使得数据条目更容易，而不需要添加一系列的硬分行，在窗口的大小不同的时候这会扰乱当时情形。

2.4.3 例子：cookie

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

Explorer3.0™ Explorer4.01™ Domino5.0™

本书包含几个cookie的例子，大部分用的是JavaScript。本例使用了一个<META>标记，设置cookie和一个CGI域，并返回值。

cookie是当客户端要求一个特殊的URL时，HTTP服务器传输给客户端的一些信息。然后，当客户端在同一域内请求其他的URL时，客户端为此域传输给服务器所有的cookies.cookie数据被保留在客户端机器上的浏览中。

当用户打开一个Domino表单时，你可以使用<META>标记的HTTP-EQUIV属性来设置一个cookie。为此，创建一个HTML的Head公式，它的值为如下格式：

```
"<meta http-equiv=\"Set-Cookie\" content=\"cookievalue\">"
```

可选地，在cookie内可以包括路径和终止日期。当用户打开表单时，服务器开始设置cookie。

为了得到cookie数据，创建一个名为HTTP_COOKIE的多值文本域（HTTP_COOKIE是一个CGI变量）。

在数据库 FormExamples.nsf的Simple表单中，设置了三个 cookies :CUSTOMER, CATEGORY和INTERESTS。在本例中，每一个cookie的值是\$\$HTML Head域内的代码，但是你也可以使用公式来计算值。通常，你可以在表单中使用 HTML Head 属性，而不是使用\$\$HTML Head域（在R4.6以前版本，\$\$HTML Head域是你惟一的选择），但是，当你在浏览器中打开表单时，使用域允许看到下列公式的结果：

```
"<meta http-equiv=\"Set-Cookie\" content=\"CUSTOMER=SMITH; path=// expires=Friday, 31-Dec-99 23:59:59 GMT;\">" + @Newline +
"<meta http-equiv=\"Set-Cookie\" content=\"CATEGORY=BOOKS; path=//"
```

```
expires=Friday, 31-Dec-99 23:59:59 GMT;">" + @Newline +
"<meta http-equiv=\"Set-Cookie\" content=\"INTERESTS=SPORTS;
path=/; expires=Friday, 31-Dec-99 23:59:59 GMT;\">"
```

第一次打开表单时, Domino从浏览器收到一些已有的cookie数据(已访问过的),并将新的cookie数据传输到浏览器。关于此点,假设服务器以前没有传输 cookie到你的浏览器,HTTP-COOKIE域的值是空的。当你单击“Reload”按钮时,浏览器将你的已更新过的 cookie数据传输到Domino服务器,并在HTTP-COOKIE域内修改。

一旦得到了用户的HTTP-COOKIE数据,可以使用公式来返回分开的值。例如下列公式,从HTTP-COOKIE域中取得了Customer,Category 和Interests的值。

```
Customer      cookies := @Implode(HTTP_COOKIE;";");
              tmp := @Right(cookies;"CUSTOMER=");
              @If(@Contains(tmp; ";"); @Left(tmp; ";"); tmp)

Category      cookies := @Implode(HTTP_COOKIE;";");
              tmp := @Right(cookies;"CATEGORY=");
              @If(@Contains(tmp; ";"); @Left(tmp; ";"); tmp)

Interests     cookies := @Implode(HTTP_COOKIE;";");
              tmp := @Right(cookies; "INTERESTS=");
              @If(@Contains(tmp; ";"); @Left(tmp; ";"); tmp)
```

这样,如果域 HTTP_COOKIE的值是CUSTOMER=SMITH;CATEGORY=BOOKS;INTERESTS=SPORTS,则各个值分别为SMITH; BOOKS和SPORTS。

cookie在存储用户喜爱的信息方面非常有用。例如,你决定用户的个人主页不用帧,可以使用一个称为FRAMES的cookie使用户自动使用适当的页面。

对于某些信息不要依赖于 cookie。用户的cookie文件可能太多,或者用户拒绝让你设置cookie(老版本的浏览则根本不支持 cookie)。例如, Netscape Navigator在接受cookie之前会弹出一个菜单显示警告信息(如图 2-4)。用户可以单击“Cancel”拒绝cookie。

当使用cookies来设计一个站点时,记住用户可能使用不同的机器来访问你的站点,如果他们遇到“自动化的”页面改变他们过去访问的东西,他们便会非常困惑。

关于cookies的更多信息,请访问:

http://www.netscape.com/newsref/std/cookie_spec.html

关于<META>标记的更多信息,请访问:

<http://cyberati.com/wguide/meta.cfm>

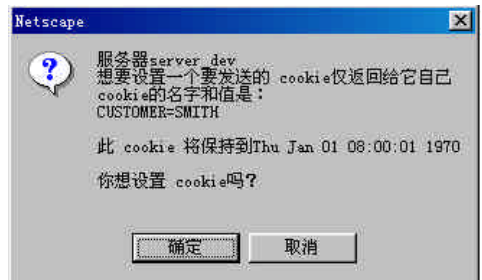


图2-4 你不能强迫用户接受你的cookie

2.5 使用表单操作按钮

可以创建表单操作按钮,用于 Notes客户端, Web浏览器,或者两者都用。对于 Notes客户端的应用,你可以配置表单操作,并在菜单条中显示,但在 Web浏览器中的应用并不支持

这一点。

对Domino的操作按钮有两种类型的使用方式。有一种缺省按钮的方式，当你在数据库属性中设置了“Web access:Use Javascript when generating pages”，即是这种结果。这一部分描述了缺省的方式。关于操作按钮的这些属性的效果的有关信息，参看第5章：编写JavaScript。

缺省地Domino将操作按钮转换为简单的URL链接。某些类型的公式比较容易转换为URL。例如，我确信你可以猜出下列公式将转换成什么样的URL：

```
@UrlOpen("http://www.notes.net")
```

但有些公式，例如@Dialogbox，因为它们没有相类似的对应URL而不被支持，如果你创建了一个表单操作按钮，但它不能被转换为一个URL链接，Domino将帮你隐藏它。

仅管Domino并不支持直接使用LotusScript的按钮，但它可以显示一个按钮，此按钮使用@command([ToolsRun Macro])公式来运行Lotus Script或Java代理。当然，这些代理只能使用Web支持的特性。对于在Domino下运行的代理，LotusScript的前端类是不可用的。例如，你不能创建一个运行Domino代理的按钮，它使用NotesUIDocument.Cut()和UIDocument.paste()。表2-3显示了在Domino应用中非常实用的表单操作按钮。

表2-3 实用的表单操作按钮

操 作	例 子
创建新文档	@Command([compose]; " OrderFom) 创建一个新的 Order Form文档
编辑当前文档	@Command([Edit Document]) 将当前文档设为编辑模式
删除当前文档	@Command ([Edit clear]) 删除当前文档
使用浏览器打开URL	@UrlOpen(" http://www.notes.net ") 打开Notes.net站点
打开一个视图	@Command([Open View];"User Profiles") 打开User Profiles 视图
在视图中打开一个特殊的文档	@Command ([Open View];"User Profiles";@Username); @Command ([Open Document]); 在UserProfiles 视图中打开当前用户的配置文档
运行代理	@Command ([ToolsRunMacro]; "Cleanup") 运行Cleanup代理
运行CGL程序	@UrlOpen ("/cgi-bin/My perl.pl") 运行MyPerl Script程序
打开导航器	@command ([Open Navigator]; "Main Menu") 打开Main Menu导航器
提供在线帮助	db:=@Subset (@Dbname; -1);@Urlopen("/" +db+ "\$help?Openhelp") 为当前数据库打开“使用数据库”文档

2.6 使用热点

热点是表单（或者文档，页面等）内包含有文本和 /或段落的一块区域，当你单击它时，便激活某个文件。Domino支持下列类型的热点：

- 链接热点。
- 操作热点。

链接热点和操作热点与表单操作按钮一样，有同样的限制条件。Domino不支持文本 popups，公式 popups，或者热点按钮。Popups没有类似的HTML对象，你可以在Domino的表单中创建热点中热点按钮，但（缺省地）Domino假定它是一个提交按钮。

一个有用的热点应用是创建菜单，图 2-5显示了在“Bug报告应用”（Bugs.nsf）中的Workflow菜单。此菜单条是使用操作热点来创建的，Domino将之转换为<A HREF>链接。

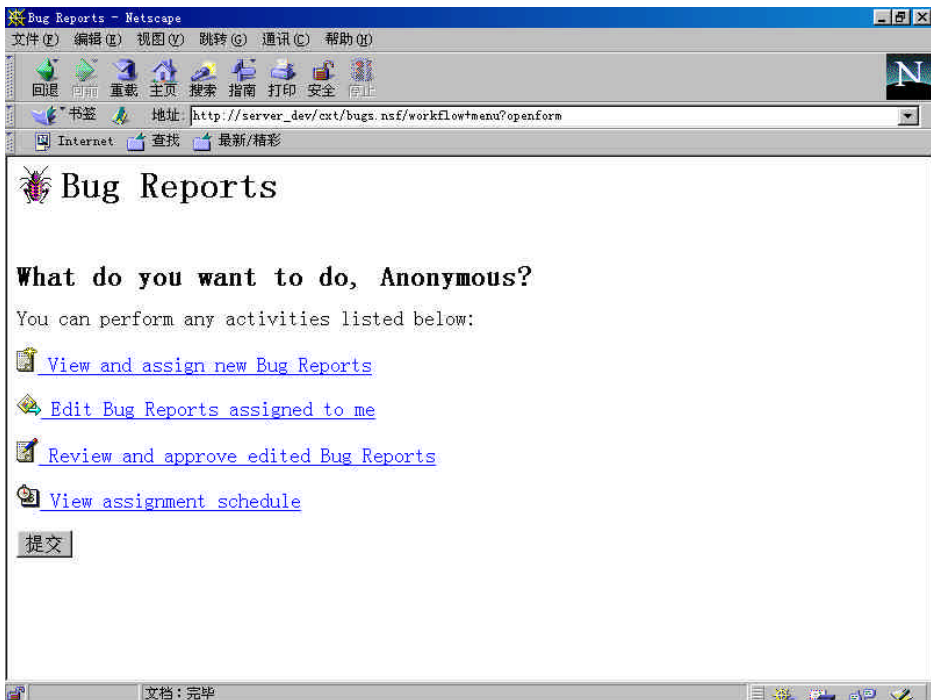


图2-5 通过操作热点和公式条件下的隐藏可以创建用户定制的按钮

第一个按钮操作使用 @Command([Open View])函数打开一个Notes视图。你可以使用下列方法得到相同的效果：

- 创建一个视图连接热点。
- 使用@UrlOpen函数和视图URL创建一个操作热点。
- 在表单中输入<A HREF>链接，然后将之转化为HTML通用文本。

1. 链接热点

一个链接热点连接到一个文档，视图或数据库，链接热点像粘贴到表单中，或保存在 RTF

文本域内链接一样，但看起来不一样。当你希望用户点击文本或段落而不是“链接按钮”时，你可以创建一个链接热点。

2. 操作热点

和表单操作按钮一样，当你点击操作热点时运行公式，和表单操作按钮又不一样，操作热点使用你提供的文本或段落来显示。当操作位于表单内特殊的区域，或者想阻止用户滚动屏幕时，常使用操作热点来代表操作按钮。

2.7 在表单内使用HTML

你不必为使表单或文档是“Web_enabled”而添加任何HTML。然而有时，在表单内包含HTML能达到特殊的效果。一个非常简单的例子是一个加重列表，假设你有一个名为ProjectSkill的多值域，用户使用下拉列表框中选定的值来填充它。之后，当用户使用只读模式打开该文档的时候，选定的值被显示时，Notes会让你选择某个老式的定界符，例如：

Project Management ;Systems Analysis;Network Engineering

使用HTML通用文本，你可以把它转换为一个加重列表。创建一个特别的计算域，令它在编辑状态时隐藏，使用下面的值把这些值转换为HTML：

```
"[<li>" + ProjectSkills + "</li>"]"
```

这个公式在ProjectSkill域的开始和结尾添加加重标签，如果你将这个域设置为每行显示一个值，你得到的结果如下：

```
[<li>Project Management</li>]  
[<li>Systems Analysis</li>]  
[<li>Network Engineering</li>]
```

在Web浏览器中转换为加重列表：

Project Management
Systems Analysis
Network Engineering

方括号和尖括号（如[<and>]）在Domino中代表直接传递到浏览器的HTML代码，作为结果，在浏览器中以加重表单的方式列出技能项目。如果你希望你的表单在Notes中和在浏览器中同样吸引人，请注意保证这个域对Notes客户端用户隐藏。

为了在域中包含HTML文本，你可以在表单中直接输入HTML文本，在HTML文本中你可以包含CGI脚本，JavaScript代码，以及其他一些可以包含在HTML文件中的东西。

原来，把HTML文本标志为HTML通用文本是把它包含在方括号中，幸运的是，现在有了更好的方法，选中文本并将其Text菜单设置为HTML通用文本选项（但是不要忘记尖括号，就像在第3章中所述，当你在视图中使用HTML通用文本你将需要它）。

2.7.1 例子：单个类视图

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

Domino R4.x 一个非常烦人的缺点是它不提供任何方法任意过滤视图。例如, 假设你有一个在数据库中由分类排序的包含所有文档的视图, 你希望当用户选择一个类别的时候, 你可以提供一个只包含所选类别文档的视图。然而, Domino 根本不提供完成这个任务的方法。当然, 你可以进行一个全文搜索:

```
FIELD Category contains "Java Applet"
```

但是它不能保持原有视图的顺序。

幸运的是, 在 R5 中, 当你在表单或页面中嵌入一个视图的时候, 你可以把它设置为一个单个类视图, 并制定返回类别的公式。当显示视图时, Domino 执行公式并只显示属于指定类的文档。

在 R4 版本下, 本节内容详细说明了 Notes 公式语句中列表的使用。主要的缺点是在公式中使用 @DbColumn 和 @DbLookup 函数的时候, 可能仅仅返回 64K 字节的数据, 另外, 它还需要一点额外的工作。这个例子包含三个主要组件:

- 在数据库的 “About” 页面中可计算的 URL 链接。
- SingleCategory 表单。
- DownloadLookup 视图。

如果你使用的是 R5, 阅读本节将学到更多在 Notes 公式中列表的使用方法。但是在使用 R5 组件创建单个类的视图的时候, 工作非常简单, 而且不必考虑 64K 的限制。

1. 可计算的 URL 链接

查看在图 2-6 中显示的 FormExamples.nsf 的介绍性页面。这事实上是数据库的 About 页面, 它被设置为在打开数据库时启动。这个页面包含连接到 URL 的分类的列表。

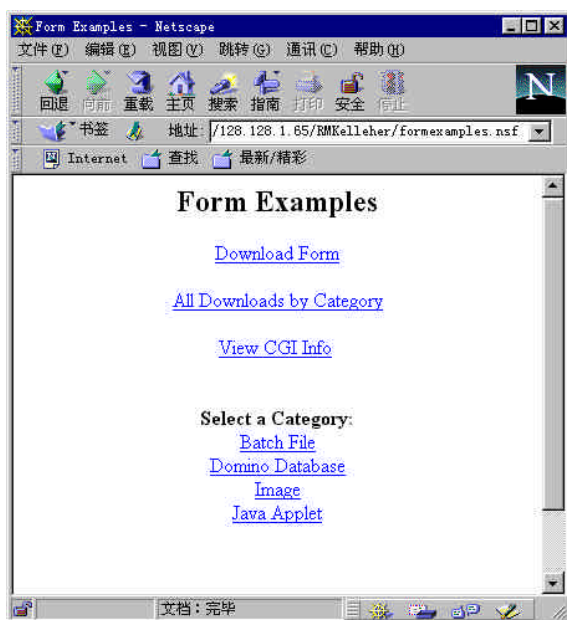


图2-6 数据库的包含计算域文本的页面为数据库中每个类创建一个 URL 链接

用About 文档中的HTML通用文本 的固定代码编写 URL链接非常简单，但是，当你添加或者删除分类的时候，这个链接将失效。因此，我们想要创建一个链接的动态列表，以便总是能够反映当前的类别。完成这个功能需要可计算的文本公式：

```
db := @ReplaceSubstring(@Subset(@DbName; -1); "\\\"; "/");
category1 := @DbColumn("Notes":"NoCache"; ""; "Downloads by
Category"; 1);
category2 := @ReplaceSubstring(category1; " "; "+");
url := "/" + db + "/SingleCategory?OpenForm&Category=" + category2;
list := "[<a href=\"+url + \"\"> +category1 + \"</a><br>]";
@Implode(list)
```

这个公式为每个在 Downloads by Category 视图第一栏中列出的类别返回一个 URL链接。在此使用了标记 HTML通用文本的老式方法：每个链接被包含在方括号中。这样一来，如果数据库包含下面4个分类：

- Batch File。
- Domino Database。
- Image。
- Java Applet。

结果如下图所示：

```
[<a href="/FormExamples.nsf/SingleCategory?OpenForm&Category=Batch1
File">Batch File</a><br>]
[<a href="/FormExamples.nsf/SingleCategory?OpenForm&Category=
DominoDatabase">Domino Database</a><br>]
[<a href="/FormExamples.nsf/SingleCategory?OpenForm&Category=Image">
Image</a><br>]
[<a href="/FormExamples.nsf/SingleCategory?OpenForm&Category=Java
Applet">Java Applet</a><br>]
```

注意，在URL中，类别名的空格被加号（+）代替。因为你不能在URL中包含空格，但是你激活URL调用的链接文本包含空格，因此需要用到两个变量， category1(包含空格)和 category2(包含加号)。

如果你才开始使用Notes公式语句，这些公式可能非常混乱。但是至少在此提出了它们有用的一方面，它可以很好地完成列表工作，让我们再来看一下 category1变量：

```
category1 := @DbColumn("Notes":"NoCache"; ""; "Downloads by Category"; 1);
```

这个变量保存了从 the Downloads by Category 视图中第一列得到的类别的列表，在 Notes 公式表示下，结果列表类似如下显示：

```
"Batch File " : "Domino Database" : "Image" : "Java applet"
```

可以使用加号把每个值连接为同样的字符串，例如，“pre- ” +category+ “-post ” 执行结果为：

```
"pre-Batch File-post" : "pre-Domino Database-post" :
"pre-Image-post" : "pre-Java Applet-post"
```

使用这种方法，我们可以把开始的括号与 <A HEAF> 标签插入到每个类别的开始，并在类别的后面添加 ,</BR> 标签和后半括号。

如果将两个列表相加，则其列表值会分别相加。例如，假设我们又有 category1 变量为下列值：

```
"Batch File" : "Domino Database" : "Image" : "Java Applet"
```

现在假设我们的变量 category2 是下面的值：

```
"AAA" : "BBB" : "CCC" : "DDD"
```

这样一来，公式 category1+category2 的结果为：

```
"Batch FileAAA" : "Domino DatabaseBBB" : "ImageCCC" : "Java AppletDDD"
```

而 category1+category2+ " ZZZ " 的结果为：

```
"Batch FileAAAZZZ" : "Domino DatabaseBBBZZZ" : "ImageCCCZZZ" :  
"Java AppletDDDZZZ"
```

在这种方法下，两个列表（category1 和 category2）把单个的，静态的字符串联合起来创建 URL 链接列表。

2. SingleCategory 表单

在 FormExamples.nsf 数据库中的 SingleCategory 表单使用大量的公式产生一个 HTML 表，创建一个视图的外观。这个表单的主要成分是名为 Links 的计算域，它包含适当文档的链接，下面是它的计算公式：

```
list := @DbLookup("Notes";"NoCache"; "" ; "(DownloadLookup)"; Category; 2);  
@If(@IsError(list); "<h3>No documents found</h3>"; list)
```

这个公式返回 DownloadLookup 视图的第二列，但只是关于那些关键值（也就是第一列的值）与特定的类别相同的文档。Category 域的值作为自变量传递到 @DbLookup 函数中。例如，假设 Category 域的值是 Image，则这个函数返回第一列为 Image 的文档的第二列的值。

Category 域从加载表单时使用的 URL 中得到它的值，在表单的顶部，一个名为 QUERY_STRING 捕捉 URL 的问号（？）后面的部分，这个 Category 域从 QUERY_STRING 中以如下公式获得值：

```
@ProperCase(@ReplaceSubstring(@Right(@UpperCase(QUERY_STRING);  
"CATEGORY="); "+" ; " " ))
```

这样一来，如果从如下 URL 中加载表单：<http://server/database/FormExamples.nsf/SingleCategory?OpenForm&Category=Java+Applet>

Category 域的值将是 Java Applet。这个值然后将被用来从 Java Applet 类的所有文档得到视图的列的数据。

另外一个包含在 SingleCategory 表单中的优点是添加新的 <category> 的链接。例如，如果你使用这个表单显示属于 Java Applet 类的文档的列表，这个链接将读 Upload a New Java Applet 并生成目标 URL：

```
http://server/database/FormExamples.nsf/SingleCategory?OpenForm&Category=Java+Applet
```

当单击这个链接的时候，一个 Category域已经设置为Java Applet的空白表单显示出来，这将通过在 Category域中使用默认公式的 QUERY_STRING值来实现。尽管从列表框中选择类别并不会花费用户很多时间，但能为用户减少麻烦总是愉快的。

在我们开始讨论 DownLoadLookup视图之前，需要指出一个重要细节，链接的域应该插入在<TABLE>和</TABLE>标签之间，这是因为 DownLoadLookup视图的每一行都使用 HTML通用文本创建一个 HTML表格的行。

3. DownLoadLookup视图

最后不等于最不重要， DownLoadLookup视图是这个例子的最重要的部分。它是插在 SingleCategory表单的表格列的数据源，产生如图 2-7结果。

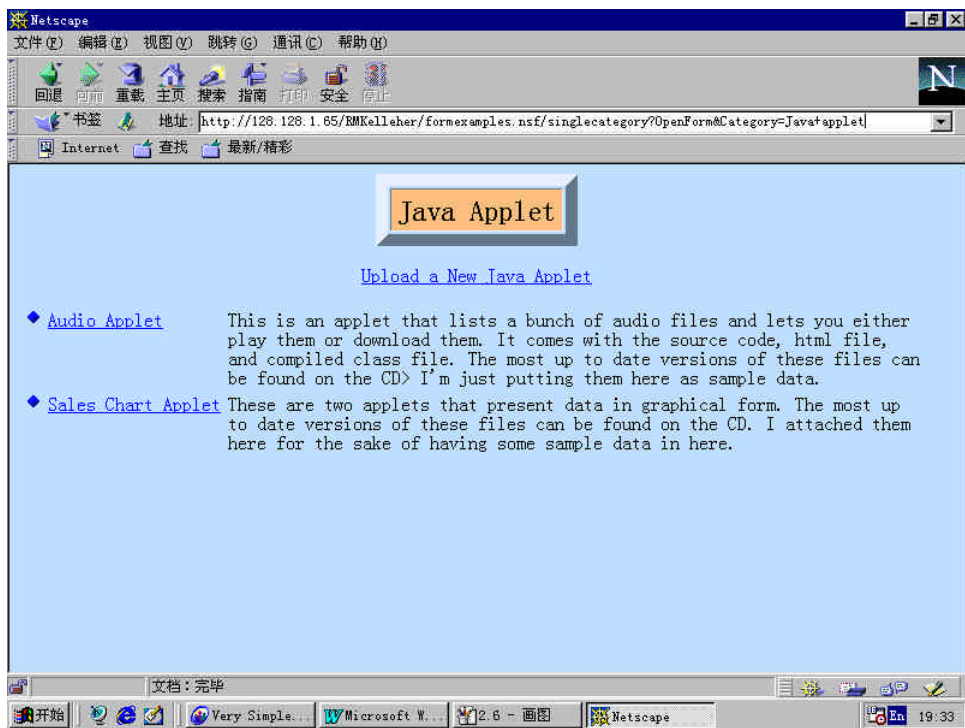


图2-7 这个SingleCategory 表单包含Notes公式和HTML突出显示
文本显示到特定的类的文档的链接

DownLoadLookup视图有两栏，第一栏，包含每个文档类，被隐藏而且以升序排列。第二行使用下面的公式为视图的每个文档产生一个表格行：

```
db := @ReplaceSubstring(@Subset(@DbName; -1); "\\\"; "/");
url := "/" + db + "/Downloads+by+Category/" +
      @Text(@DocumentUniqueID) + "?OpenDocument";
"<tr>" +
"<td valign=\"top\"><img src=\"/icons/vwicn100.gif\" border=0></td>" +
"<td valign=\"top\"><a href=\"" + url + "\">" + Title + "</a></td>" +
"<td valign=\"top\">"+Description+"</td>" +
"</tr>"
```


这个公式在一个单个视图栏中创建多个 HTML 栏，每个<TD>.....</TD>代表一个表格行的单元格。在每个行中，第一个单元格包含一个 标签，该标签显示一个小的着重按钮。第二个单元格包含一个到文档的 URL 的链接，使用 Title 域作为链接文本，第三个单元格包含描述。所有单元格被包含在定义 HTML 表格一行的<TR>和</TR>标签之间。

2.7.2 例子：显示附属图形文件

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

这个例子详细说明了怎样使用 HTML 通用文本显示附属在文档中的图形。

图 2-8 显示了一个 FormExamples.nsf 数据库中的 Download 表单在 Internet Explorer 中的效果。在按下作为文件上载控制的一部分的浏览器按钮的时候，你可以选择一个本地文件上载到数据库。

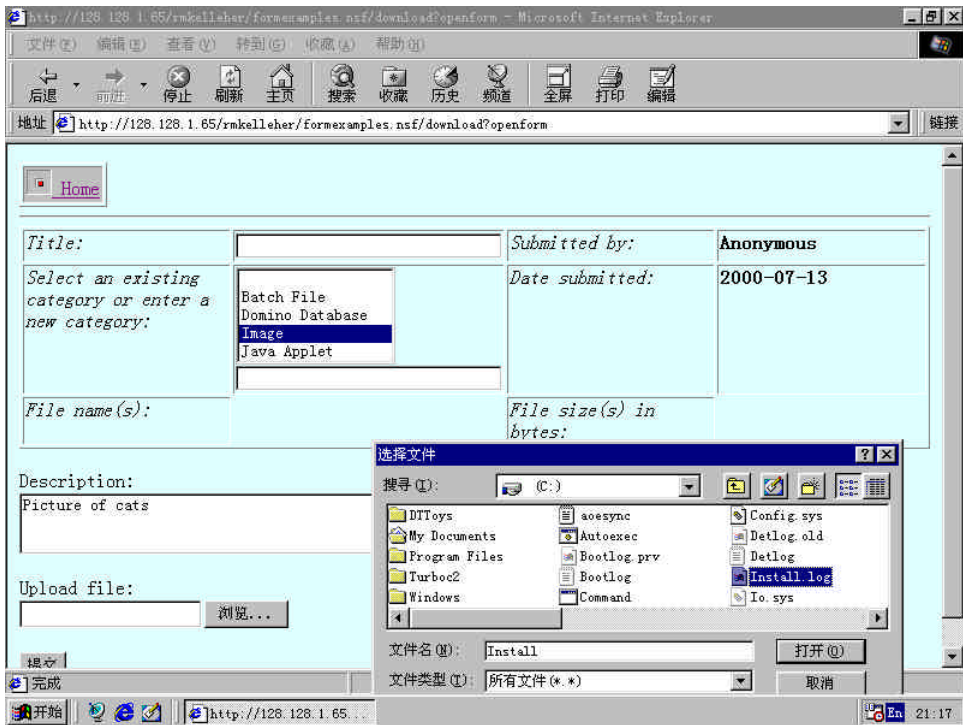


图 2-8 Download 表单包括一个文件上载控制以便你可以在文档中附加文件

使用 Domino 表单嵌入的控制上载的文件被自动附加在响应的 Notes 文档中。默认情况下，Domino 把文件附件显示为一个图标，该图标连接到附加的 URL 中。附加的 URL 采用如下形式：

http://server/database/view/document/\$FILE/filename

根据文件类型的不同，点击一个指向文件的链接将有不同的效果。例如，点击一个指向

典型的二进制文件的链接将引发浏览器显示一个提示询问你是想用某个应用程序打开该文件还是想把它保存在本地的硬盘上。点击一个指向 HTML 文件的链接，将引起浏览器显示 HTML 文件内容，这与把 HTML 文件保存在服务器的文件系统中效果是不一样的。

加载表单可能在 Notes 文档中使用不同类型的附件。这些文件的大部分不需要特别的处理方式，但是对于 Web 兼容的图形文件，比如 JPG 形式或 GIF 形式，用户如果能在加载它们之前就能够看到它们的话是不是更好呢？

最简单的处理办法是创建一个计算域，这个域使用 @AttachmentNames 函数产生一系列的 标签，每个附属文件一个。（@AttachmentNames 函数返回所有附属文件的列表，类似地，@AttachmentLengths 函数返回所有附属文件的长度）。这种方法的问题在于并非每个文件都是图形文件，事实上，并非每个附属图形都是 Web 兼容的形式，某些图形可能是 TIF，PCX 或者 BMP 形式。我们需要做的是只对那些 JPG 或 GIF 形式的文件产生 标签而忽略其他文件格式。

这个问题的解决办法并不像它看起来那样简单，你不能使用简单的 @If(条件；操作；否则) 公式产生一个列表，因为它总是返回单个值而不是值的列表，比如如下公式：

```
list := "aaa.pcx" : "bbb.gif" : "ccc.jpg";
@if(@Right(list; ".") = "gif"; "yes"; "no");
```

如果认为这个公式将产生三个值的列表，比如（“no”：“yes”：“no”），那就错了。这个公式只返回单个值：“yes”。它只判断全体中是否有符合条件者，而“no”值只有在没有文件名以.gif结尾时才返回。

现在看一下在 Download 表单底部的 IMG 域，这个计算域产生了一系列的 链接，而且只针对那些以.gif或.jpg结尾的附件。图 2-9 显示了包含 4 个文件附件的文档：两个 JPEG 图像(.jpg)，一个 GIF 图像(.gif) 和一个 Windows Bitmap(.bmp)。注意两个 JPEG 图像和一个 GIF 图像已经被显示出来，而 BMP 文件则没有。

下面是 IMG 域的公式：

```
tmp := @UpperCase(FileName) + "." + @Right(FileName; ".");
gifs := @Trim(@Left(tmp; ".GIF") + @Right(tmp; "GIF"));
jpgs := @Trim(@Left(tmp; ".JPG") + @Right(tmp; "JPG"));
files := @Trim(gifs : jpgs);
db := @ReplaceSubstring(@Subset(@DbName; -1); "\\\"; "/");
url := "/" + db + "/Downloads+by+Category/" + @Text(@DocumentUniqueID) +
"/$FILE/" + files;
@if(files=""; "" ; "[<img src=\"" + url + "\" border=0>]");
```

让我们使用一个简单的脚本一次一步运行这个公式，假设文档包含 4 个附件，因此，FileName 域的值是：

```
"aaa.gif" : "bbb.jpg" : "ccc.bmp" : "ddd.gif"
```

在公式的第一行每个文件名的末尾再加一个扩展名，得到如下结果：

```
"AAA.GIF.GIF" : "BBB.JPG.JPG" : "CCC.BMP.BMP" : "DDD.GIF.GIF"
```

第二行公式调用 @Left(tmp; ".GIF") 返回下面列表：

```
"AAA" : "" : "" : "DDD"
```

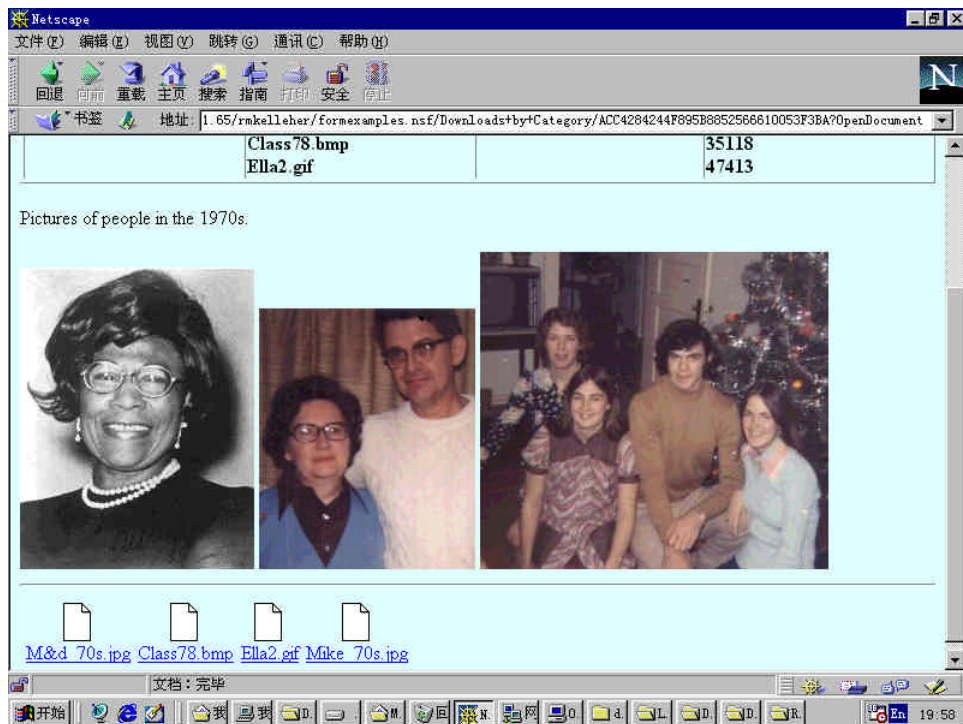


图2-9 在这个表单中的计算域使用 标签显示
附属的GIF和JPG文件而忽略其他文件

当有两个字符串使用 @Left函数不能发现任何 “.GIF” 的时候，返回两个空值。

在同一行通过使用同样方法 @Right函数产生的列表为两个非空值添加扩展名。在第二和第三列中由于字符串中根本不存在 “.GIF”，故返回空值，因此调用 @Right(tmp; “GIF”)返回如下列表：

```
".GIF" : "" : "" : ".GIF:"
```

@Trim函数从两个列表中移去空字符串，因此第一个列表现在是：

```
"AAA" : "DDD"
```

第二个列表现在是：

```
".GIF" : ".GIF"
```

这两个列表现在被成对结合，然后创建一个 GIF文件列表。换句话说：

```
"AAA" : "DDD" + ".GIF" : "GIF" = "AAA.GIF" : "DDD.GIF"
```

接下来，Notes公式的列表处理能力就得得心应手了。我们现在只得到了 .GIF文件格式的文件，使用同样的方式，可以得到 .JPG文件格式的文件。这两个列表组成了我们想使用 连接显示的图像文件，其他文件，如 ccc.bmp就被忽略掉了。

提示 在“Download”表单中，我们的主要目的是保证上载的文件可以附加在文档

中并显示在 Web 中以便将来的下载。但是有时你可能对用户附加在文档中的文件（比如一个图形）有其他用途，当文档被显示的时候，你不想显示附件的图标。为了隐藏上载的附件，在表单中创建一个名为 \$V2AttachmentOptions 的域并将其值设为 0。本指示由 DOVID GROSS 提供。

2.7.3 例子：把文本域显示为文本区域

Navigator3.04™

Navigator4.05™

Domino4.6.1™

Explorer3.0™

Explorer4.01™

Domino5.0™

默认情况下，当 Domino 把一个表单转换为 HTML 形式的时候，它把 RTF 文本域转换为 <TEXTAREA> 标签，把文本域转换为 <INPUT TYPE = "TEXT"> 标签。通常情况下这种转换工作良好，文本域的值较短，可作为姓名或类别，RTF 域的值较长，可作为信件的主题。但是，偶尔也有例外，例如，假设你有一个包含一段文字或者更多的 Description 域，而且你想把这个域包含在视图中，则你就不能使用 RTF 域，因为 RTF 域的内容不能显示在视图中。那么，开发者应该怎么做呢？

就工作环境来说，你可以使用 HTML 通用文本创建一个文本域和一个 <TEXTAREA> 界面如图 2-10。首先，你应该了解 Domino 不管你使用的文本域是自动产生的还是人工创建的，只要这个域的 NAME 属性与设计表单时创建的域的属性相同即可。

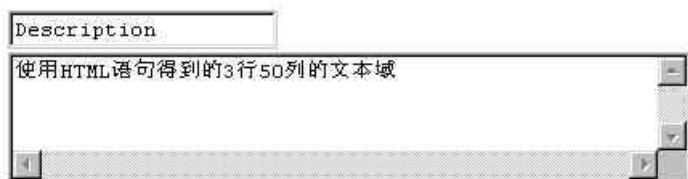


图2-10 这个文本区域用来作为 Notes 的文本域的界面

现在，回顾一下在 FormExamples.nsf 数据库中的 Download 表单，这个表单包含名为 Description 的文本域，这个域被设置为只有当文档是只读模式的时候才可见。当你填写表单的时候，这个域隐藏。

在 Description 域下面是如下 HTML 通用文本：

```
[<textarea name="Description" rows=3 cols=50> [Computed Value] </textarea>]
```

在此 [Computed Value] 代表了一个包含非常简单的公式的计算文本热点：

Description

HTML 通用文本创建了一个 3 行高 50 行宽的文本区域，允许用户有足够的空间填写描述。注意文本区域的 name 属性被设置为 Description，这样，当你在文本区域中输入一个值并提交表单的时候，你输入的值被保存在文档的 Description 域中。

最后一个难题是为什么要使用计算文本，如果文本没有被包含在 <TEXTAREA> 和 </TEXTAREA> 标签之间，每次编辑或退出文档的时候将丢失原来的 Description。计算文本保证当你编辑一个存在的 Download 文档的时候，原来的 Description 总是出现在文本区域中，如

果不编辑文本区域并提交表单的时候，Description域保持不变，当你对文本区域进行了编辑的时候，该域的内容被更新。

2.8 在表单中使用CGI变量

CGI变量是能把服务器端信息传递到其他 CGI程序或脚本中的环境变量。典型情况下，当用户提交表单的时候，CGI变量被设置，处理表单的CGI程序从环境中获得这些变量。CGI程序可能是一个Perl脚本，C程序，Domino CreateDocument URL, 一个LotusScript代理或其他能够处理一个表单的程序。

在Domino中捕获CGI变量有下面两种方式：

- 在Domino表单中创建一个特别的域。
- 在LotusScript或Java中创建一个DocumentContext对象并访问它的属性。DocumentContext对象在第6章（编写LotusScript）和第8章（编写Java代理）中有详细讲解。

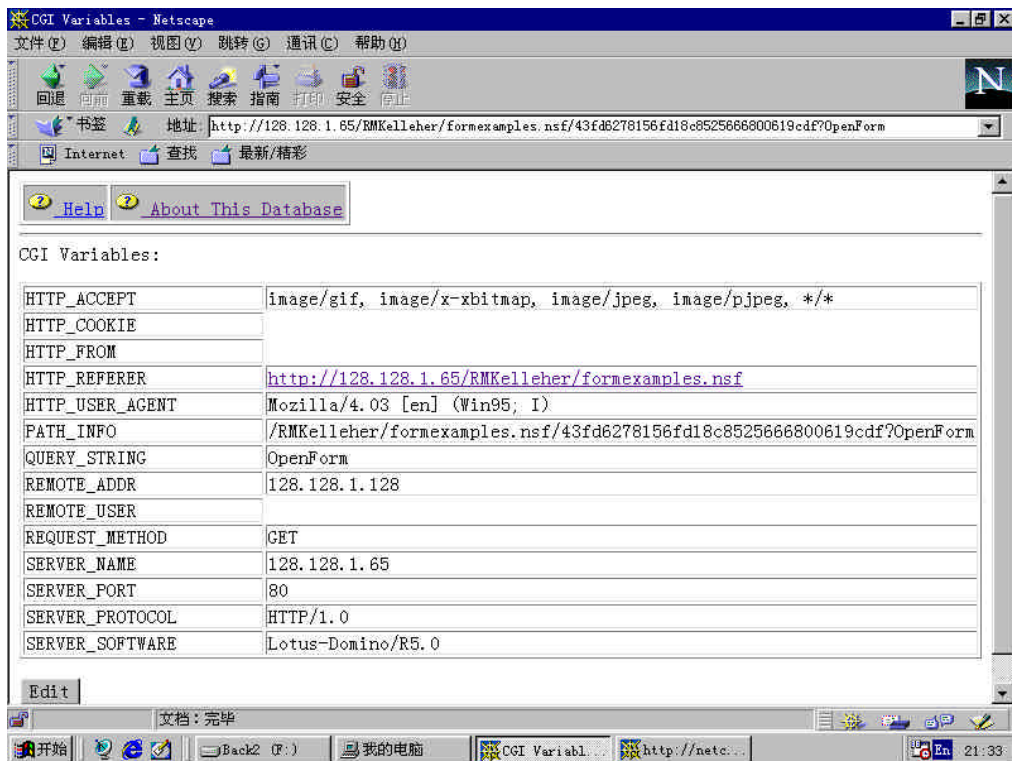


图2-11 这个表单使用计算并显示域显示 CGI变量

为了在域中捕获CGI变量，创建一个以CGI变量命名的文本域（例如，HTTP_COOKIE）。一般地，在编辑状态下将该域设置为隐藏，这样用户就不能改变域中的值。当用户打开表单的时候，HTTP_COOKIE环境变量的实际值变成了域中的默认值。图 2-11显示了捕获下面CGI变量的表单。

HTTP_ACCEPT	列出客户端可以接受的多媒体类型
HTTP_COOKIE	任何属于当前服务器的用户的 COOKIE数据
HTTP_REFERER	能被用户激活并到达此处的 URL
HTTP_USER_AGENT	浏览器类型
PATH_INFO	当前URL，不包括协议和服务器信息
QUERY_STRING	在URL问号后面附加的查询信息
REMOTE_ADDR	用户的IP地址
REMOTE_USER	用户登录名
REQUEST_METHOD	发出请求的方法（常常是一个表单中的 URL链接或邮件）
SERVER_NAME	HTTP服务器的IP地址
SERVER_PORT	Domino服务器端口（默认为 80）
SERVER_PROTOCOL	发出请求的协议版本和名称
SERVER_SOFTWARE	响应请求的服务器软件版本和名称

CGI域根据它们是可编辑域、计算域、显示时计算域而有所不同。在表单打开的时候可编辑CGI域捕获信息。当你想在表单打开的时候得到 CGI变量的信息时使用可编辑 CGI域。

当使用可编辑的CGI域的时候，服务器使用 CGI变量作为域的默认值。当表单被提交的时候，这些值被保存在文档中，就像用户手工键入的一样。如果你想在以后读取或编辑文档信息的时候，这些值将不反应任何实际 CGI变量的更新。例如，假设使用一个包含可编辑的 HTTP_COOKIE域的表单，在提交一个表单之后（如果没有改变默认值），用户在其他地方修改了 HTTP_COOKIE的值。随后，用户又打开刚提交过的表单，编辑它（不改变 HTTP_COOKIE的值），第二次保存它，这个HTTP_COOKIE域将保持第一次保存时的值。

CGI计算域在每次文档被保存的时候更新。于是，当 Sally创建一个文档的时候，REMOTE_ADDR域将保持 Sally的IP地址。以后，当Joe保存这个文档的时候，REMOTE_ADDR域将被更新为Joe的IP地址。当你想得到文档最后一次被保存的环境信息的时候使用CGI计算域。CGI计算域的公式应该为域的名称。

CGI显示时计算域在每次文档被打开的时候其值更新，但是它不会被保存在数据库中，如果Sally建立一个文档，REMOTE_ADDR域将显示Sally的IP地址但是不会保存它。以后，当Joe打开文档的时候，Joe的IP地址被显示。当你想显示更新过的 CGI环境变量但并不想保存下来以备以后使用的话，使用CGI显示时计算域。CGI显示时计算域的公式应该是域的名称。

例子：在URL中指定一个默认的值

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

还有一个未讨论过的在 FormExamples.nsf数据库中下载 Download表单时的一个特性：当在URL中包含一个Category参数的时候，可以在打开表单的时候指定 Category域的默认的值。回忆一下在 combobox例子中，我们使用两个域捕捉输入类别：ExistingCategory和

NewCategory。你可以从 ExistingCategory 列表框中选择一个存在的类或在 NewCategory 中输入一个新的类，然后 Category 域从 ExistingCategory 列表框域或 NewCategory 域得到类别的值。

如果激活 OpenForm URL 加载表单的时候不使用参数，则 ExistingCategory 列表框域的值为空白 ("")。如果你在 URL 末尾添加一个 Category 参数并且参数的值是存在的类别的时候，ExistingCategory 列表框域的默认值将通过参数提供。例如，如果调用如下 URL：

```
http://myserver/FormExamples.nsf/Download?OpenForm&Category=Java+Applet
```

则 ExistingCategory 列表框域的默认值为 Java Applet，一个数据库中已经存在的类。如果类的参数值不是一个已经存在的类，则 ExistingCategory 列表框域值默认为空白 ("")。例如，如果你激活下面这个 URL：

```
http://myserver/FormExamples.nsf/Download?OpenForm&Category=Nonexistent
ExistingCategory 列表框域值默认为空白 ("" )。
```

为了完成上述功能，ExistingCategory 列表框域使用一个默认值公式以便当使用 URL 加载表单的时候得到 Category 参数的值。如果这个值在关键字列表中，就使用它，否则，忽略它。

```
cat := @ProperCase(@ReplaceSubstring(@Right(@UpperCase(QUERY_STRING);
"CATEGORY="); "+" ; " "));
list := @DbColumn("Notes":"NoCache"; "" ; "Downloads by Category"; 1);
@if(@IsMember(cat; list); cat; " ")
```

与 ExistingCategory 列表框域相似，NewCategory 默认值也依靠用来下载表单的 URL。如果激活包含 Category 参数的 OpenForm 的 URL，而且参数值是一个不存在的类别，NewCategory 以该参数为默认值，下面是 NewCategory 域的默认值公式：

```
cat := @ProperCase(@ReplaceSubstring(@Right(@UpperCase(QUERY_STRING);
"CATEGORY="); "+" ; " "));
@if(cat != "" & ExistingCategory = " " ; cat; " ")
```

这个公式使用户创建一个属于特殊类别的文档变得更为方便。为了了解我们的方法，试着选择在 FormExamples.nsf 页面上的列表框上的一个类，如果选择 Image，你可看到属于 Image 类别的页面列表和一个到 Upload a New Image URL 的链接。

```
http://server/db/FormExamples.nsf/Download?OpenForm&Category=Image
```

如果选择 Java Applet，看到了属于 Java Applet 类别的页面列表和一个到 Upload a New Java Applet URL 的链接。

```
http://server/db/FormExamples.nsf/Download?OpenForm&Category=Java+Applet
```

当单击这些链接的时候，Download 表单打开，其中已经添入了默认类别。好了，简直像一个复杂的外科手术，但是总是有一点帮助的。

2.9 表单和搜索

你没有必要为了搜索创建一个特殊的表单。当用户输入一个搜索 URL 的时候，Domino 提

提供了一个默认搜索表单，然而，你可能为了某种风格的统一或者提供额外的功能而自己定制这种表单。

2.9.1 定制视图搜索表单

当你选择一个连接到视图的搜索链接的时候，Domino在当前数据库中寻找一个名为 \$\$Search 的表单。如果存在，Domino显示这个表单，否则，它显示默认的视图搜索表单。

搜索表单的 ACTION 属性依赖于搜索的视图。例如，由 Domino 产生的视图的搜索表单可能包含如下 <FORM> 标签：

```
<FORM METHOD=POST
ACTION="http://server/123e543a123e543a123e543a123e543a?SearchView">
```

然而，当 Domino 显示一个规则表单的时候，对表单的操作使用 CreateDocument URL，下面是两种避开这个问题的主要方法：

- 方法1：使用你自己的 <FORM> 标签覆盖由 Domino 产生的 <FORM> 标签。
- 方法2：使用 \$\$Return 域或者 WebQuerySave 代理改变当用户按下 Submit 按钮时 Domino 的反应。

第一个方法有点旧了，但是确实有效（不能保证以后仍然有效）。只需要把下面的 HTML 通用文本包含在你的 \$\$Search 表单的顶部：

```
[</FORM><FORM METHOD=POST ACTION=""?SearchView">]
```

这将终止由 Domino 产生的 <FORM> 模块并开始一个新的 <FORM> 模块。这个操作的 URL 包含在当前目录中，因此如果用户显示 \$\$Search 表单使用下面的 URL：

```
http://server/123e543a123e543a123e543a123e543a/$searchForm?SearchView
```

则这个操作 URL 转换为：

```
http://server/123e543a123e543a123e543a123e543a?SearchView
```

使用第二种方法，你使用一个 \$\$Return 域或者 WebQuerySave 代理分析用户的输入并把浏览器指向 SearchView URL。例如，Domino 的“Search Site”模板的 \$\$Search 表单使用下面的 \$\$Return 公式：

```
DBName:=@Subset(@DbName;-1);
"[/1DBName1"?SearchSite&Query="1Query1"&SearchOrder="1@Text(Sort)1"&
SearchMax="1@Text(MaxResults)1"&SearchWV="1@If(ExactMatch="";"TRUE";"
FALSE")1"&SearchThesaurus="1@If(UseThesaurus="";"FALSE";"TRUE")1"]]"
```

这个公式返回如下值：

```
[/db?SearchSite&Query=cats&SearchOrder=1&SearchMax=10SearchWV=
TRUE&SearchThesaurus=FALSE]
```

因为这种方法使用了 HTTP 的 GET 方法而不是 POST 方法，它导致了很长的可读性很差的 URL。但是，另一方面，这种方法使你使用 Notes 公式语句计算查询自变量的值。

为什么要使用生成查询自变量的公式呢？

坦率的说，大部分用户并不能很熟练地使用 Domino查询语法组成如下查询：

```
FIELD Title CONTAINS "domain" OR FIELD Categories CONTAINS
```

但是这些查询可能非常有用而且有效率。当你只对某个特定的域感兴趣的话为什么要对每个文档进行全文搜索呢？你可以通过添加选项定制 \$\$Search表单使用户搜索特定的域变的简单，从而可以使用 \$\$Return公式（或WebQuerySave代理）根据用户的输入计算一个 Domino搜索URL。

无论使用方法1还是方法2，你可能都想在你的 \$\$Search表单中包含表2-4中部分或所有域。

表2-4 \$\$Search 表单的域

域 名	数 据 类 型	描 述
Query	文本域	搜索的单词或条件
SearchMax	文本域或关键字域	搜索结果的最大文档值
SearchOrder	关键字域	对搜索结果排序：
		无序/1；老文档在前/2；新文档在前/3
SearchWV	关键字域	是否包含单词的变体：
		包含/TRUE；不包含/FALSE
SearchThesaurus	关键字域	是否包含同义词：
		包含/TRUE；不包含/FALSE
SaveOption	文本域	值为0表示不能保存由本表单创建的文档

2.9.2 例子：定制搜索界面

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

在FormExamples.nsf数据库中的 \$\$Search表单由上一节的方法1定制。但是使用方法1的问题是你能使用公式计算查询域的值。你可能设想使用下面的计算域创建一个查询：

```
"FIELD Categories CONTAINS " + Category
```

不幸的是，由于提交 \$\$Search表单公式不能计算，这种方法将不生效。

对于需要计算查询值的简单查询来说，你最好使用方法2，可以创建一个 \$\$Return域包含如下公式：

```
db := @Subset (@DbName; -1);
"/" + db + "/All?SearchView&Query=FIELD+Categories+CONTAINS+" +
Category + "\""]"
```

例如，如果用户选择Cats类，结果URL是：

```
/db/All?SearchView&Query=FIELD+Categories+CONTAINS+"Cats"
```

对于更复杂的查询来说，使用方法1可能更好。但是你将面临一个问题：如果不能使用公式的话，那么应该怎样计算查询域的值。

由于这种情况如此频繁发生，JavaScript就派上用场了，在此我们不是使用公式在表单提

交以后计算查询域的值，而是使用 JavaScript 在表单提交之前计算查询域的值。

图2-12显示了FormExamples.nsf数据库中的\$\$Search表单。打开All by Category视图点击Search按钮可以显示这个表单。

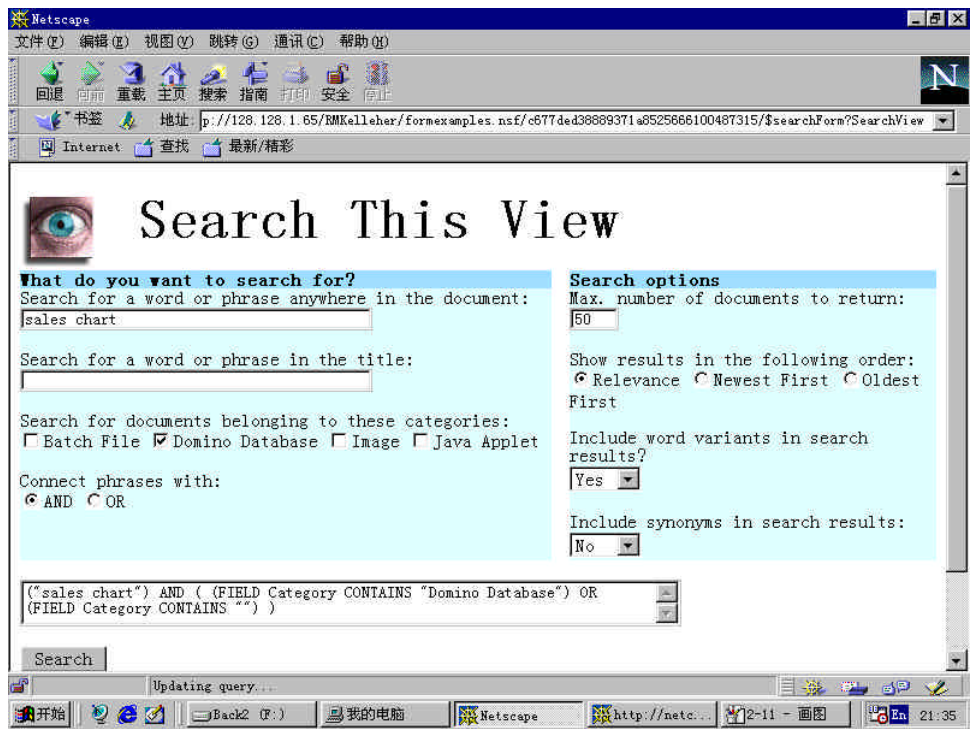


图2-12 这个定制的视图搜索表单使用 JavaScript 帮助用户建立一个复杂的查询

当JavaScript函数被使用的时候，你改变任何一个表 2-5 中域的值都将引发查询的重新计算。

表2-5 由JavaScript函数重新计算的域

域	HTML属性公式
Words	"size = 40 onChange=\" update();\""
Title	"size = 40 onChange=\" update();\""
Categories	"size=5 onClick=\" Update();\""
Connector	"onClick=\" update();\""

当单击Search按钮调用update()函数，这通过在<FORM>标签中使用 onSubmit事件实现：

```
[</FORM><FORM NAME="MySearch" METHOD=POST ACTION="?SearchView"
onSubmit="update();">]
```

其中update()函数如下：

```
function update() {
    window.status = "Updating query...";
    form = document.forms["MySearch"];
```

```

words = form.Words.value;
title = form.Title.value;
categories = getCheckboxValue(form.Categories);
connector = getRadioValue(form.Connector);
conditions = new Array();
cond1 = (words == "") ? "" : "(" + words + ")" + connector + " ";
cond2 = (title == "") ? "" : "(FIELD Title CONTAINS \"" + title +
    "\"" + connector + " ";
for (i = 0; i < categories.length; i++) {
    categories[i] = "(FIELD Category CONTAINS \"" + categories[i] +
        "\"" + connector + " ";
}
cond3 = (categories.length == 0) ? "" : "(" +
    categories.join(" OR ") + " ";
if (cond1 != "") conditions[conditions.length] = cond1;
if (cond2 != "") conditions[conditions.length] = cond2;
if (cond3 != "") conditions[conditions.length] = cond3;
query = conditions.join(" " + connector + " ");
form.Query.value = query;
}

```

这个函数为每个域建立一个分离的查询条件，把每个条件包含在圆括号内，并使用 AND 或者OR将它们连接起来，如果你指定了多个类，第三个条件事实上是由 OR连接的多个条件，例如，假设你想发现属于下面任何一个类的文档：

- Database
- Java
- Groupware

那么这将生成如下查询：

```

(FIELD Category CONTAINS "Database") OR (FIELD Category CONTAINS "Java") OR
(FIELD Category CONTAINS "Groupware")

```

由于某种原因，JavaScript不提供内嵌的获得单选框组或复选框组的值的方法。

GetRadioValue()函数返回字符串代表一个单选框组选定的值：

```

function getRadioValue(radio) {
    for (i = 0; i < radio.length; i++) {
        if (radio[i].checked) {
            return radio[i].value;
        }
    }
    return null;
}

```

getCheckboxValue()函数返回一个字符串数组，代表在复选框组中选定的值：

```

function getCheckboxValue(checkbox) {
    s = "";
    for (i = 0; i < checkbox.length; i++) {

```

```

    if (checkbox[i].checked) {
        s += checkbox[i].value + " ";
    }
}
s = s.substring(0, s.length-1);
if (s=="") return s;
return s.split(';');
}

```

这个函数创建了一个字符串，在其中各个值之间以分号（；）分开，然后使用 `split()` 函数把字符串转换为数组。以后数组还可以使用 `join()` 函数把其转换为单一的字符串。JavaScript 中的 `split()` 和 `join()` 函数类似于 Notes 公式语句 `@Explode` 和 `@Implode`。

如图 2-12，用户在 Words 域中输入“sales chart”，选择 Domino Database 和 Java Applet 类别，指定 AND 作为链接，将导致如下查询：

```

("sales chart") AND ( (FIELD Category CONTAINS "Domino Database")
OR (FIELD Category CONTAINS "Java Applet") )

```

你能想像让普通用户输入这样的查询语句吗？使用一个定制的 `$$Search` 表单，和一点 JavaScript 的帮助，你可以帮助用户创建一个他们确实想要的查询。

2.9.3 定制 Site Search 表单

定制一个 Site Search 表单比定制一个数据库视图搜索表单要容易的多，因为你可以使用现存的 Notes 表单：Web Search Simple 和 Web Search Advanced 表单。这些表单被包含在 Site Search 模板中，你可以在你的 Site Search 数据库中改变这些表单的设计。

2.9.4 定制 Search Results 表单

默认情况下，Domino 的搜索结果是很简单的文档列表。你可以把搜索结果使用两种方式定制：

- 为特定视图的搜索结果创建一个表单。
- 为数据库中的任意视图的搜索结果创建一个表单。

1. 为特定视图创建一个搜索结果表单

可以创建一个表单，Domino 使用这个表单为一个特定的视图显示搜索结果。只需要把视图的名称或别名设置为 `$$SearchTemplate`，在其中包含 `$$ViewBody` 域或把视图嵌入到你想要显示搜索结果的表单中。则结果将被显示为特定的视图方式。

2. 创建一个搜索结果表单的默认视图

你可以为数据库中的任意视图的搜索结果创建一个表单，把表单命名为 `$$SearchTemplateDefault`，在其中包含一个 `$$ViewBody` 域或在你想显示的位置嵌入视图。则结果被以 `$$SearchTemplateDefault` 的方式显示。如果视图已经包含了它自己的 `$$SearchTemplate` 表单，则搜索结果以 `$$SearchTemplate` 表单的方式显示而非 `$$SearchTemplateDefault` 方式显示。

定制搜索结果表单的缺点是无法得到返回文档的数量,这意味着不能在搜索结果的顶部加上诸如“发现500个文档”之类的东西。

3. 创建一个站点搜索结果表单

现在还无法定制站点搜索结果。如果为站点搜索创建 \$\$SearchTemplatedefault表单,站点搜索应用程序不能识别它,因此仍然使用默认形式,这种对站点搜索结果进行定制的功能我们只能寄希望于以后的版本了。

2.9.5 例子:对搜索结果表单的定制

Navigator3.04™

Navigator4.05™

Domino4.6.1™

Explorer3.0™

Explorer4.01™

Domino5.0™

定制搜索结果的一个突出特点是它可以包括表单的操作按钮,以便帮助用户导航。图 2-13 显示了在例子“程序错误数据库”(Bugs.nsf)中的\$\$SearchTemplateDefault表单。在搜索完一个视图以后,这个表单包括 Login和Help按钮帮助用户进行下一步工作。

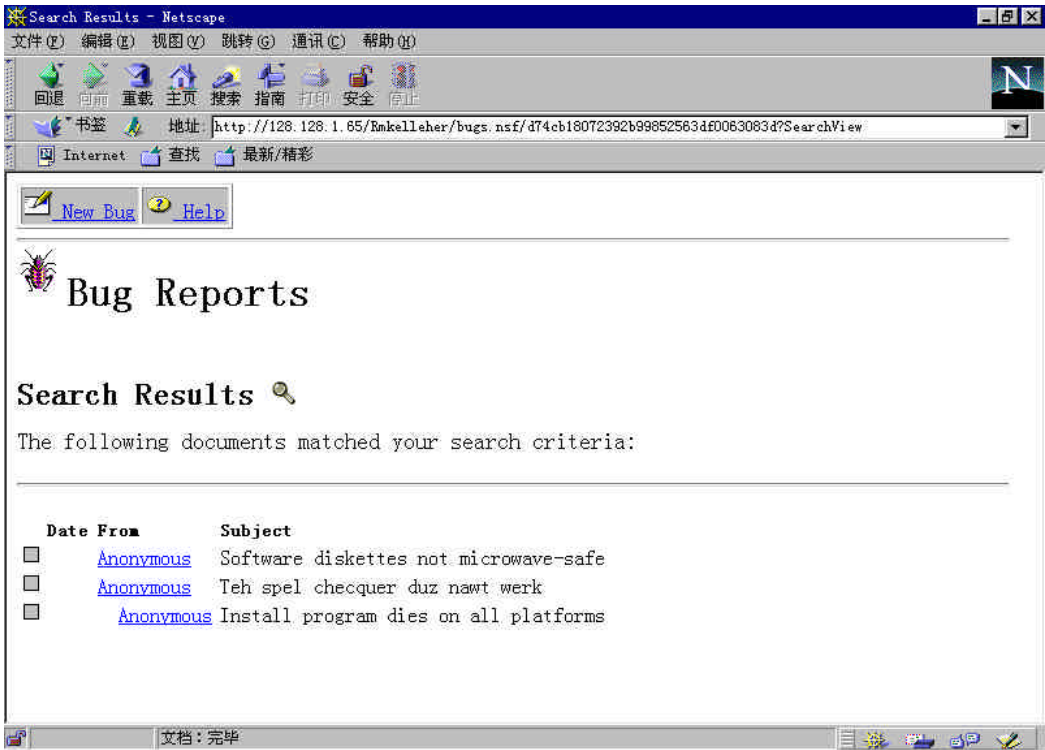


图2-13 这个定制搜索结果表单提供操作按钮帮助用户导航

这个表单的创建非常简单。仅有的特别元素是它的名字和 \$\$ViewBody域,通过名字告诉 Domino,当用户搜索一个无自己 \$\$SearchTemplate表单的视图的时候使用该表单。 \$\$ViewBody域告诉用户搜索结果应该放在页面的哪个位置。 \$\$ViewBody是R4.5时代的产品

(现在你可以使用嵌入视图元素), 但是在R5中仍然有效。

2.10 定制表单提交按钮

默认情况下, Domino在每个表单中包含一个 Submit按钮, 当你选择此按钮的时候, Domino从表单中收集数据并使用它们创建一个新的文档。你可以创建一个自己的 Submit按钮, 使其拥有一个更有意义的标签, 比如 Search或Register。为此, 只需要创建一个按钮, 你不需要输入一个公式, Domino自然知道你提交表单的时候应该做什么。

有时你可能不希望显示Submit按钮, 下面是几种“官方”认同的途径:

使用页面元素而不是表单元素。然而你可能需要一些表单所特有的功能。

启动数据库的属性“Use JavaScript when generating pages”具体情况请查阅第5章。

在R4.6.1版本你可以通过在表单中创建一个按钮并使用隐藏属性使其不可见从而抑止Submit按钮。但是, R5中该功能无效。

还有一个在R4和R5中都有效的方式就是使用HTML注释标签把Submit按钮注释掉。在一个孤立的行上建立一个按钮, 在按钮左边输入:

```
<!--
```

在按钮的右边, 输入:

```
-->
```



图2-14 你可以使用HTML的注释标签注释表单的提交按钮

然后选择整个段落并把它设置为HTML通用文本格式。图

2-14显示了段落的样子。

2.11 使用隐藏域

默认情况下, 你在表单中隐藏了一个域, 当Web用户调用一个表单的时候 Domino将不产生任何HTML语句。通常情况下, 这就是Domino应该采取的操作。如果设计者隐藏了一个域的话, 它常常意味着用户不应该看到它的值。

但是, 有些时候, 你可能因为某种原因隐藏一个域但是仍然希望浏览者可以访问该隐藏域的值, 因此需要HTML的隐藏对象:

```
<input type = "hidden" name = "HiddenField" value = "I am hiding!">
```

当Domino把一个表单转换为一个HTML的时候, 你可以通过启动表单的“Generate HTML for all fields”属性通知Domino为隐藏域产生一个隐藏标签。Domino把隐藏对象的名字属性设置为隐藏域的名字, 把它的值属性设置为隐藏域的属性。

简单的例子: 从一个文件上载控件中得到文件名

Navigator3.04™

Navigator4.05™

Domino4.6.1™

Explorer3.0™

Explorer4.01™

Domino5.0™

想像一下你有一个包含文件上载控件的表单，除了上载文件的名字，你还需要该文件在用户端的位置信息。你可以通过在表单中创建一个名为 @AttachmentNames 的计算域非常容易地得到文件名。但是若想得到原来的路径信息，你需要编写一个在客户端执行的脚本。

在 FormExamples.nsf 数据库中的 HiddenField 表单使用一个隐藏域，让它捕获通过上载控制或文件对象上载的文件的原始名字。你可以启动表单的“Generate HTML for all fields”属性，使隐藏的 Filename 域被转换为如下 HTML：

```
<INPUT NAME="Filename" TYPE="hidden" VALUE="">
```

提交一个表单的时候触发一个名为 populateHiddenField() 的函数，它从上载控制面板值属性中得到文件名：

```
<script language="JavaScript">
function getFileUpload(form) {
    for (i = 0; i form.elements.length; i++) {
        var obj = form.elements[i];
        if (obj.type == "file") {
            return obj;
        }
    }
    return null;
}

function populateHiddenField(form) {
    var obj = getFileUpload(form);
    if (obj != null) {
        alert(obj.value);
        form.Filename.value = obj.value;
    }
}
</script>
```

为了测试这个例子，在 Web 浏览器中打开 HiddenField 表单并上载一个文件，如果你上载的文件名为 C:\myfiles\userguid.doc，那么 Filename 域的值就是 C:\myfiles\userguid.doc。

2.12 把 RTF 文本显示为一个小应用程序

到现在为止，你看到的任何关于 RTF 文本的例子都使用非常原始的方法，默认情况下，Domino 把 RTF 文本转换为文本区域。例如

```
<textarea name="Body" rows=5 cols=30 wrap=virtual></textarea>
```

HTML 的文本区域效果不错，但是与你在 Notes 中访问的 RTF 文本域相比，你不能对其中的文字设置任何格式如加粗或斜体等。然而，在某些情况下其非常重要，例如，有时句子中的斜体用来强调单词“is”。失去文字格式有时会改变文本的实际含义。当格式很重要的时候，使用如图 2-15 所示的 rich text 小应用程序。

与 HTML 文本区域不同，rich text 小应用程序允许用户使用 Web 表单提交格式化文本。这个小应用程序由 Domino 提供，为了显示它只需要启动小应用程序的 RTF 显示属性。但是请注

意，它将花费用户更多的下载时间。

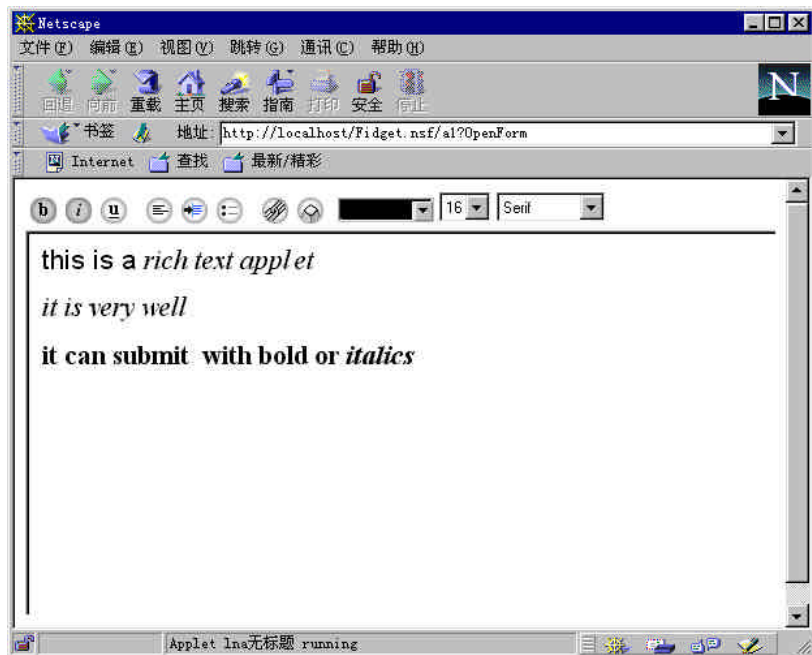


图2-15 RTF文本小应用程序允许用户经过 Web提交格式化文本

参考信息

- Notes.net中的 Iris Today 包含一些使用表单的有用的信息。为此，你可以访问 <http://notes.net/today.nsf/date?SearchView&Query=forms>.
- 在第3章，使用视图工作，提供了在表单中嵌入视图的额外例子和技巧。
- 关于层叠式表格的信息，访问 <http://www.htmlhelp.com/reference/css/>。

本章小结

为了简单的应用，你不需要为可以使用 Web的Domino表单做任何特别的事情。无论你是为Notes客户端或为Web浏览器创建表单，还是在创建域中公式，以公式为条件隐藏，计算文本，指定域，HTML属性，表单操作按钮，热点等情况下 Notes的公式语句都是不可缺少的。

你可以通过创建一个特定的名为 \$\$Return的域为提交表单定制特定 Domino反应。这个域的值可以是如下两个反应之一：在表单被提交后 Domino显示的文本,你想让浏览器指向的包含在方括号中的URL。

你可以在表单中嵌入其他设计元素，包括视图，导航器，大纲，文件上载控件。当你要求视图显示特定的表单，把表单命名为：

\$\$ViewTemplate for viewname

如果要为数据库所有的视图设置一个默认的表单，把表单命名为 `$$ViewTemplateDefault`。当你希望某个导航器使用特定的表单去显示。把表单命名为：

`$$NavigatorTemplate for navname`

如果你希望数据库中的导航器都使用特定的表单去显示，把表单命名为 `$$NavigatorTemplateDefault`。

为了改变表单的 `<BODY>` 标签的属性，创建一个 HTML Body Attribute 公式。为了把文本放在表单 `<HEAD>` 和 `</HEAD>` 标签之间，创建一个 HTML Head Attributes 公式。你还可以改变表单中的其他元素的 HTML 属性，比如域或按钮。这在对象激发一个特定的实践触发 JavaScript 函数或规定文本域或文本区域的大小的时候非常有用。

默认情况下，只有可以转换为 URL 的表单操作按钮在浏览器中是可见的（启动 “ Use JavaScript when generating pages ” 将改变这种默认操作，详细情况见第 5 章）。为用户提供指向不同位置或函数的链接的最简单的办法是创建一个表单操作按钮。其优点（也可以说是缺点）是外观相当一致。

热点与操作按钮基本一样，只不过对于操作来说其外观可以由我们控制。操作热点可以触发公式以便引发其他操作，比如删除一个文档。连接热点可以指向一个 URL。在 Web 应用程序中，两种热点都被转换为 URL 链接。热点的最有用的功能是创建菜单。

HTML 通用文本是 Domino 不予编译的文本。你可以通过使用方括号括住它们以把表单中的文本标志为 HTML 通用文本。它必须与由尖括号括住的 HTML 标签连接。例如 `[<h1>hello</h1>]`。你还可以选定文本并将其文本属性设置为 HTML 通用文本完成相同的任务。表单中的 HTML 通用文本有很多应用，本章中的例子包括通过类别过滤视图，显示图像附件，把一个文本域显示为一个 `<textarea>` 对象。

你可以在表单中添加以 CGI 变量命名的域，来捕捉 CGI 变量。可编辑的 CGI 域只是在表单第一次打开的时候捕捉变量信息。CGI 计算域在每次文档保存之前更新，CGI 显示并计算域每当文档被打开的时候更新，但是其结果不在数据库中保存。最有用的 CGI 变量包括 `QUERY_STRING`（得到 URL 参数），`HTTP_REFERER`（得到原来的 URL），`HTTP_USER_AGENT`（决定浏览器类型），`PATH_INFO`（得到当前的 URL），`HTTP_COOKIE`（得到 COOKIE 数据）。

为数据库中所有视图的搜索定制表单，只需要将表单命名为 `$$Search`，并在表单提交的时候完成以下两个动作之一：

- 方法 1：使用你自己产生的 HTML 通用文本 `<FORM>` 标签覆盖 Domino 产生的 `<FORM>` 标签，例如：

```
[</FORM><FORM METHOD =POST ACTION= " ?Search View " >]
```

- 方法 2：使用 `$$Return` 域或者 `WebQuerySave` 代理分析用户的输入然后把浏览器指向 `SearchView` URL，例如：

```
[/db.nsf/All?SearchView&Query=FIELD+Category+contains+(Java+Applet)]
```

在表单中包含 `Query`，`SearchMax`，`SearchOrder`，`SearchWV`，`SearchThesaurus`，`SaveOptions` 域。

你还可以为一个特定的视图创建一个搜索结果显示的默认表单，只需要将该表单命名为 `$$SearchTemplate for viewname`

你还可以为数据库中所有的视图的搜索结果创建一个显示的默认的表单，只要把它命名为 `$$SearchTemplateDefault`。

为了抑止 Domino 自动产生 Submit 按钮，用 HTML 注释标签将其注释掉（`<!-- and -->`）。为了允许用户经过 Web 提交格式化文本，启动 RTF 域的小应用程序显示属性。此时 Domino 将不显示通常的 HTML 文本区域，而是显示一个可以进行各种文本格式设置的 rich text 小应用程序。