

第4章 使用帧结构、大纲和其他设计元素

4.1 简介

在这个章节中,你们将学习怎样通过使用特殊的设计元素,包括帧结构集、大纲、页面等等,来提高Domino Website的性能。

4.2 在帧结构中设计

帧结构集允许将屏幕分隔成多个独立的视窗,并把每一个视窗看成一个帧。每一个帧都包含一个目标为另一个帧的链接,也就是说,这个链接所指向的页面将出现在另一个帧内。

帧结构集在某一领域中是非常有用的。例如,你有一个叫 index.html 的页面,它包含到这个站点所有页面的链接。不论用户想要打开哪一页,通常都需要在 index视图下,以便对站点进行操作。一种方法是通过在站点中对每一页的链接拷贝和粘贴来完成。但这样需要花费许多的时间,而且每当你增加或移动一页时,都会不可避免地重新启动一页。如果这时使用帧结构,你可以在一个帧中显示 index.html,在另一个特别指定的帧中显示它所链接的页面。

在版本5中,你可以使用帧结构集中的设计元素很容易地创建帧。当然,如果你想要将帧运用到你的应用程序中,理解帧在 HTML中的形成是非常有利的。在 HTML中创建帧结构,需要使用 `FRAMESET` 标签,它所有的属性已列在表 4-1中。

表4-1 <FRAMESET>标签属性

属 性	Netscape Navigator	Microsoft Internet Explorer
border=n	是	
bordercolor=color	是	
cols=list	是	是
frameborder=[yes/no]	是	
frameborder=[1/0]		是
framespacing=n		是
rows=list	是	是

在帧结构集中创建一个单独的帧,你只要将 `FRAME` 插入到 `FRAMESET` 和 `/FRAMESET` 中。`FRAME` 的属性已列在表 4-2中。

表4-2 <FRAME>标签属性

属 性	Netscape Navigator	Microsoft Internet Explorer
bordercolor=color	是	
frameborder=[1/0]		是

(续)

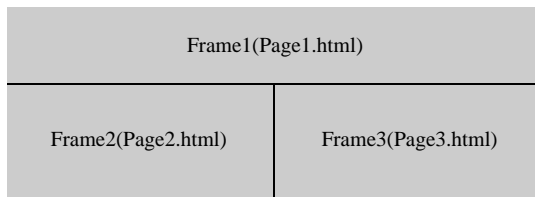
属 性	Netscape Navigator	Microsoft Internet Explorer
frameborder=[yes/no]	是	
marginheight=n	是	是
marginwidth=n	是	是
name=string	是	是
noresize	是	是
scrolling=[yes/no/auto]	是	是
src=url	是	是

使用scrolling=N0时请注意,这时,不仅在你的布局中禁止使用滚动条,而且在用户导航窗口中也禁止使用滚动条。如果用户使用不同的字体、窗口或显示分辨率,则该选项对用户来说非常糟糕,特别是加上“noresize”属性更是这样。一般情况下,最好选用auto。

接下来在HTML中再在左右各创建一个帧且尺寸一致。左边的帧显示导航页面,右边的帧显示网页。

```
<frameset cols="50%,50%">
<frame name="Netscape" src="http://home.netscape.com">
<frame name="Microsoft" src="http://www.microsoft.com">
</frameset>
```

可以在帧结构集中包括帧结构集。例如,在随后的HTML中创建一个帧结构集就像这样:



```
<frameset rows="25%,75%">
  <frame name="frame1" src="http://www.acme.com/page1.html">
  <frameset cols="50%,50%">
    <frame name="frame2" src="http://www.acme.com/page2.html">
    <frame name="frame3" src="http://www.acme.com/page3.html">
  </frameset>
</frameset>
```

在默认情况下,在帧中的链接显示的目标帧是同一个帧。如果想要将其他的帧确定为目标,就需要使用 A HREF 标签中TARGET属性。例如,如果想在Frame1或Frame2中显示链接的网页,就要在Frame3中设置:

```
<a href="http://www.acme.com" target="Frame3">Click here</a>
```

4.2.1 帧的不利之处

虽然帧非常有用,但也会给操作带来不便之处,特别是对视觉有损害的用户而言。帧也

使得打印页面很困难——许多浏览器强迫用户在每一次打印时只能选择一页。在有些时候，你可以发挥你的想像力去避免使用帧。例如，首先，将所有问题都详细理清。你把站点中每一页中一系列的链接都罗列出来，并确定需要使用的，如果想要将所有的链接都只集中在一个特定区域时，Domino不用帧也能容易地达到目的：仅仅在每一个窗体使用一个视图列或大纲行元素。帧结构的页面也比不包含帧结构的 HTML 页面要求更多的设计测试。在浏览器视图设置中帧的尺寸是很容易改变的，但也同样容易导致可怕的重叠或对其他用户完全隐藏元素。通常，你可以在设计中使用表格以避免涉及到帧的不利点。

4.2.2 使用帧结构集设计元素

当你创建一个新的帧结构集时，将得到一个预先确定的关于帧结构集外形的窗体，可在其中选择帧结构集的结构，如图 4-1。

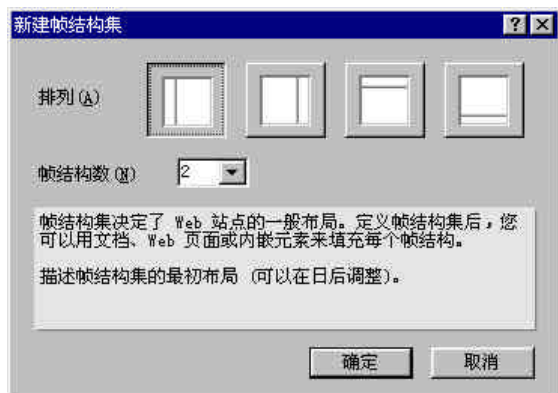


图4-1 创建帧结构的开始

假如你并不知道需要什么样的结构，不用焦虑，可以在新元素形成后马上改变它。你可以通过拖动边界改变尺寸，也可以通过点按钮来将帧分离成多行多列（或有效地变为另一个帧结构集）。你也可以通过改变属性来控制帧的边界是否可见，尺寸是否可调等等。

1. 一个简单的例子：Frameset1

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

Explorer3.0™ Explorer4.01™ Domino5.0™

看一看图 4-2 中 Frameset1 一个例为 Framesets.nsf 的数据库文件。帧结构集中 4 个帧分别显示了不同的窗体。需要说明的是，帧的初始内容是通过在“Named Element”（相当于一个窗体或一视窗）中进行设置，或输入一个 URL（或一个可以返回 URL 的计算公式），或在帧中粘贴链接。

为了在 Web 浏览器中显示帧结构使用如下 URL 语法：

```
http://server/database/frameset?OpenFrameset
http://server/database/frameset
```

图 4-3 是 Frameset1 在 Netscape Navigator 中的显示情况。

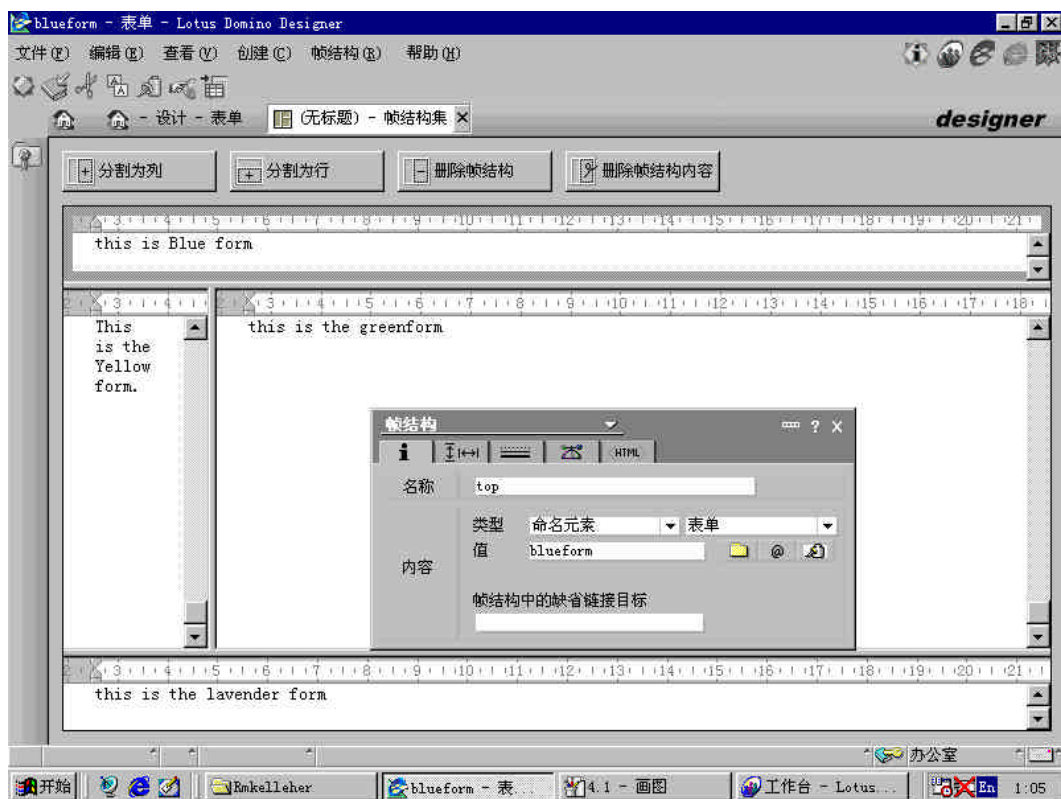


图4-2 在这个帧结构的每个帧中包含一个元素，在此是一个表单

提示 你可以通过使用 Domino 中一个特殊的链接特征在你的 Web 应用程序中来建立 HTML 链接。通过从剪贴板上粘贴链接，从菜单中编辑——拷贝得到链接，然后再选择链接类型：Anchor Link，文件链接、数据库链接。现在你粘贴链接到文件中去而后选择元素。当在 Web 浏览器上浏览时这些链接会被当成 HTML 链接执行。

2. 一个更有用的例子：创建一个包含帧的视图

Navigator3.04™ Navigator4.05™ Domino4.6.10

Explorer3.0™ Explorer4.01™ Domino5.0™

帧在 Domino Web 应用程序中的一个共同之处是：在一个帧内显示视图而在另一个帧内显示所选择的内容，图 4-4 中的帧结构集正是这样：在 Framesets.nsf 数据库中的 Frameset2，当你点击左边窗口的文档链接时，这个文档的内容就会出现在右边的帧内。

要建立一个帧视图，要做的只是建立一个视图，然后建立帧结构集。在左边的帧中，通过在“Named Element”说明左边的帧作为目标链接帧。在左边的帧视图的 HTML 代码生成时，Domino 自动默认你的链接。在右边的帧中，Named Element 提供一个 URL 或链接帧的初始内容。

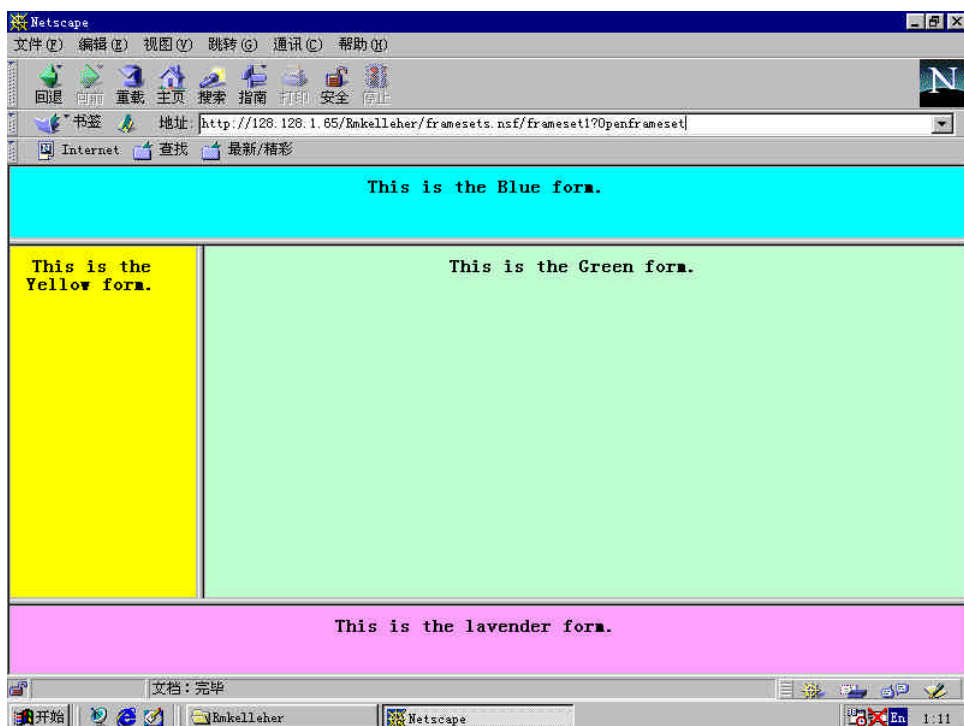


图4-3 使用帧结构元素而不是<FRAMESET>标签的特点使你可以看到最后的结果

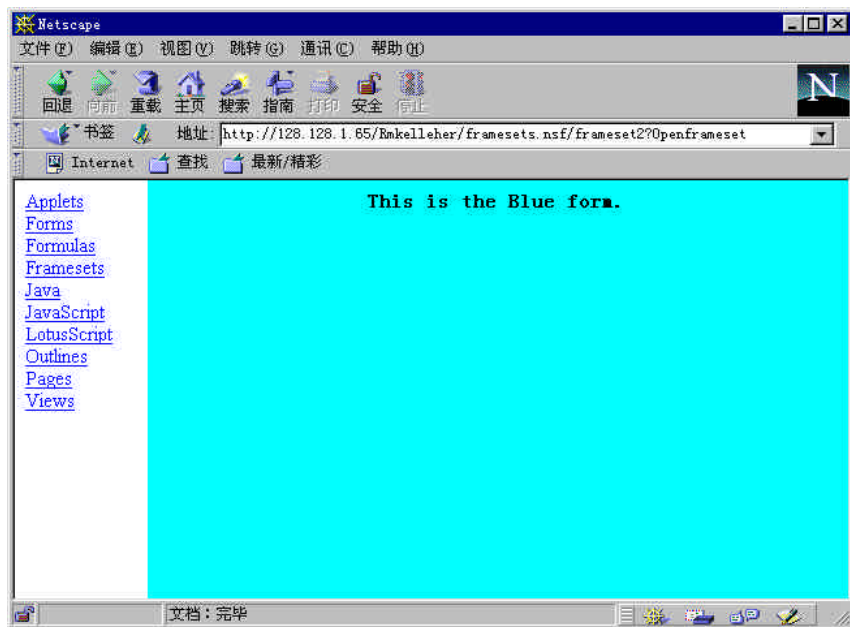


图4-4 对于某些类型的视图来说，帧可使用户查看文档内容变得更为简单

在Domino的早期版本中，也可以建立如图4-4中这样的帧视图，但是并不容易。在标题为

Topics46的视图中示范了在版本4.6中建立帧的技巧。这个视图为“display contents as HTML”，其意是这整个视图是作为HTML解释执行的。视图的第一列是被隐藏的，它主要显示分类的主题域。第二列使用一个公式来计算得出一个目标链接：

```
db := @ReplaceSubstring (@Subset (@DbName;-1;"\";" /");
url := "/" + db + "/Topics46/" + @Text (@DocumentUniqueID +
"?OpenDocument";
"<a href=\"\" + url + \"\" target=\"Right\">\" + Subject + "</a><br>"
```

图4-5表明这个视图的设计模式。

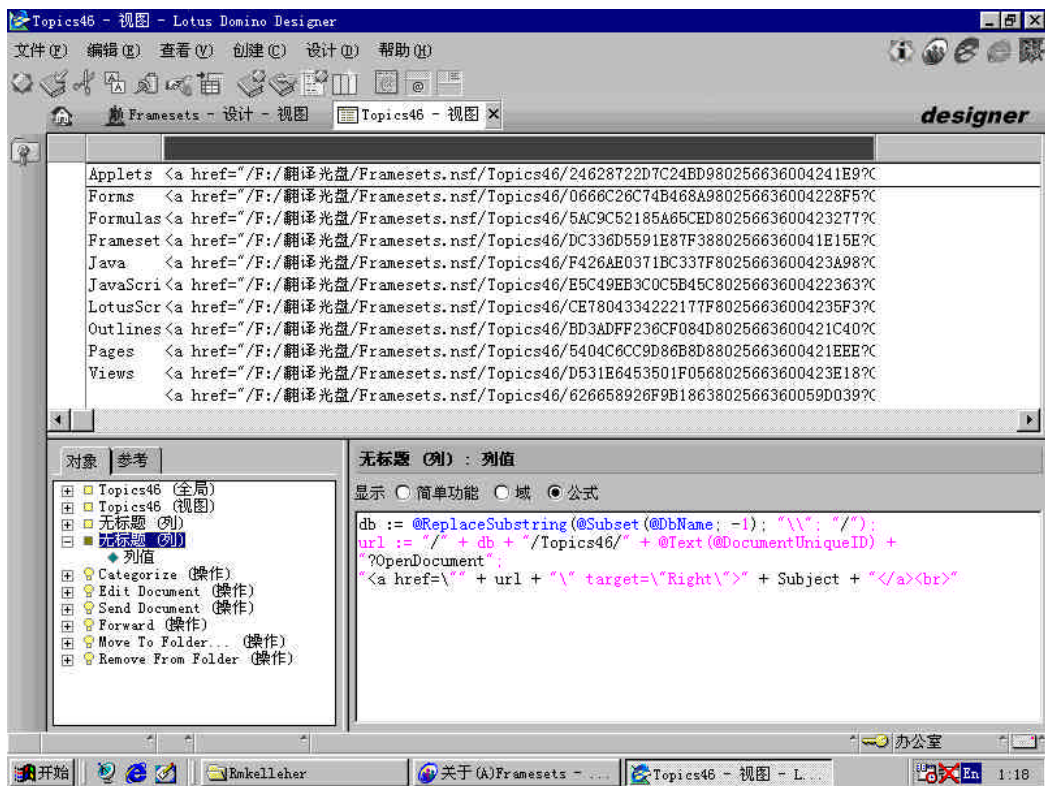


图4-5 在早期的版本中创建帧视图也是可能的，尽管并不简单

JavaScript 和帧

使用JavaScript 控制是访问帧属性的一个便利的方法。一个帧中插入一段脚本就可以通过访问“top window”中的帧数组中提及到的另一帧的元素（“top window”是外层的窗口，即帧结构的详细说明）。例如，设想帧结构包括三个帧，假使分别命名为Frame1、Frame2和Frame3。Frame1或Frame3通过以下的多种方法访问到Frame2：

```
top.frames[1]
top.frames["Frame2"]
top.Frame2
```


脚本语言也能通过调用 window.open()函数来改变Frame2的内容,或设置帧的重要属性。例如:

```
window.open("http://www.acme.com", "Frame2");  
top.Frame2.location = "http://www.acme.com";
```

如果想要查看怎样使用 JavaScript的范例,请翻阅这一章节中的例子 New Mail Notifier和第5章的例子three_frame。

4.2.3 使用<FRAMESET>标签

如果你使用的是早期版本的 Domino,或打算将你的应用程序在早期的服务器上运行,你就不能使用帧结构设计元素。这时,你不得不使用 HTML建立自己的<FRAMESET>标签。建立步骤如下:

- 1) 建立一个包含HTML标签的窗体来建立帧结构集。
- 2) 形成一个文档内容被当作 HTML执行的窗体。
- 3) 创建一个包含文档的新窗体。
- 4) 将这个窗体在Web浏览器上浏览。

1. 一个简单的<FRAMESET>例子:包含帧结构集的窗体

下面的HTML代码讲述了Domino怎样创造图4-3中的帧结构集。很明显的,其中包括两个帧结构集。外面的帧结构集有三行。里面的帧结构集将第二行分成了两列。

```
<HTML>  
<!-- Lotus-Domino (Build 159.1 (Developer Test Build 2) (Intl) - 22  
May 1998 on Windows NT/Intel) -->  
<HEAD>  
<TITLE>Frameset1</TITLE></HEAD>  
  
<FRAMESET FRAMEBORDER=1 ROWS="15%,*,14%">  
  
<FRAME FRAMEBORDER=1 NORESIZE SCROLLING=no NAME="Top" SRC="/  
Framesets.nsf/BlueForm?OpenForm">  
  
<FRAMESET COLS="20%,*">  
  
<FRAME FRAMEBORDER=1 NAME="Left" SRC="/Framesets.nsf/YellowForm?  
OpenForm">  
  
<FRAME FRAMEBORDER=1 NAME="Right" SRC="/Framesets.nsf/GreenForm?  
OpenForm">  
</FRAMESET>  
  
<FRAME FRAMEBORDER=1 NAME="Bottom" SRC="/Framesets.nsf/LavenderForm?  
OpenForm">  
</FRAMESET>  
</HTML>
```

在版本 4.6 中可以使用这些 HTML 代码来创建帧结构集。如果在 Web 浏览器中打开 Frameset.nsf，会注意到这一页包括一个指向一个帧标签的链接，即为如下的 URL：

http://server/RMKelleher/Framesets.nsf/FramesetView/FramesetForm?OpenDocument

这个链接指向创建正在使用的帧结构集窗体的文档。这个窗体包括与创建 Frameset1 中的帧同样的 HTML<FRAMESET> 标签。在窗体的属性中，这个窗体形成为 “ treat document contents as HTML ”。所以，当你打开一个窗体并对其中的文档进行编辑时，其他 4 个窗体的显示内容与这个一致。

2. 一个更有用的关于 <FRAMESET> 的例子：A New Mail Notifier

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

Explorer3.0™ Explorer4.01™ Domino5.0™

这个关于 Lotus 查询的例子是由马德里的 Daniel de la fuente 提供的。这个文件名为 WebMailFR.nsf。这个用户自定义的 Web 邮件模板数据库利用 <FRAMESET> 建立多功能的界面。界面中的一个帧是被隐藏并含有特殊意图的。每隔 20 条记录，这个隐藏的帧就会检查用户的收信箱，看是否已有新的邮件。如果有，就通过改变另一个帧的内容来通知用户。

图 4-6 是收信箱在 Netscape Navigator 中的显示。当你第一次打开这个数据库时最顶层的帧将会给出提示信息告诉你没有新的邮件。

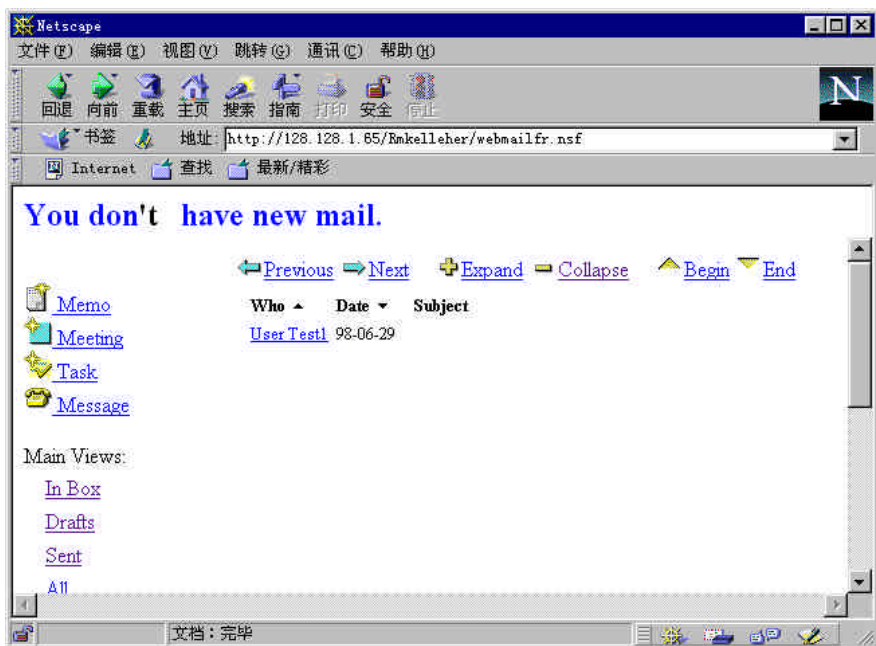


图 4-6 当你第一次打开数据库的时候，你看到默认的“没有新邮件”消息

只要数据库还继续打开，不论是哪一个视图或帧，这个隐藏的帧都会检查是否有新的邮件到来。当有新的邮件堆积在收信箱中时，这个隐藏的帧就会发出关于新邮件的信息到最顶层的帧里，如同图 4-7。

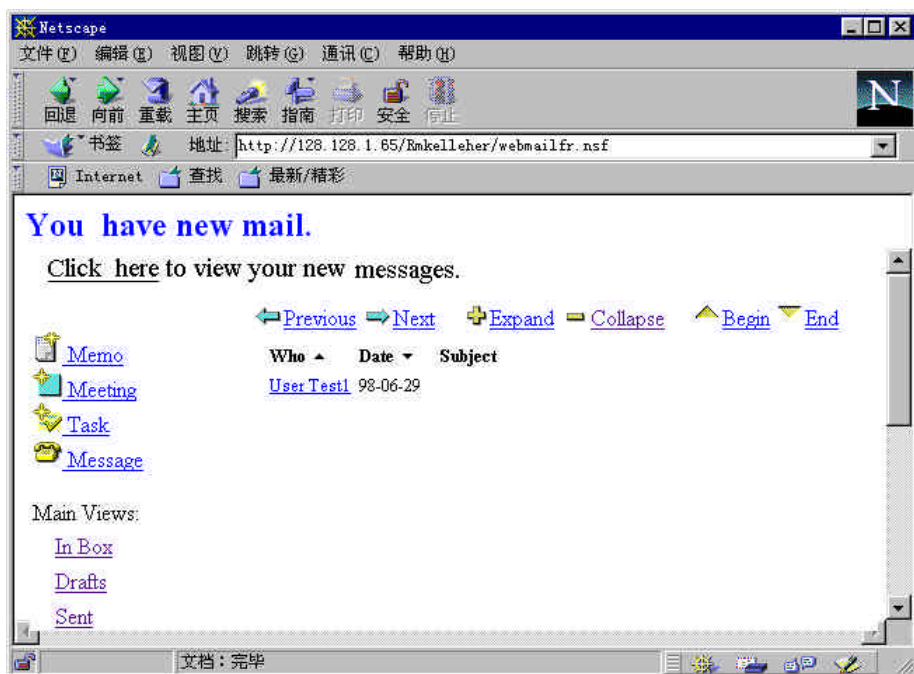


图4-7 当有新邮件到达的时候，你看到“新邮件”消息

不需要在服务器上运行任何代理，这个特点是多么的好。你设想一下：你拥有 20名用户，每个用户每隔20条记录在服务器上运行一次 Java或LotusScript代理，你的服务器将会变得不负重荷！这个解决方案减轻了主机在实际上逻辑调整；服务器只需要启动一个包含少量简单公式的Domino表单。

- 帧结构集文档。
- the Cookies form。
- 收信箱的视图模板表单。

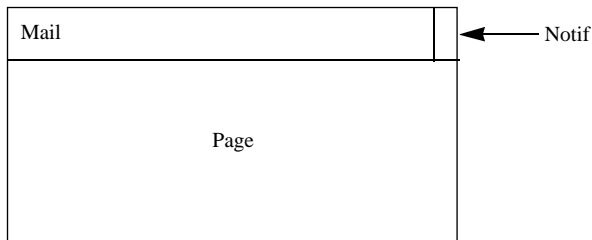
3. 关于帧结构集文档

配置WebMailFR数据库以启动第一个关于数据库文档的链接。所以，在你的 Web浏览器中打开这个数据库时，你能立即指向指定的目标链接文档。这种特殊的文档在帧结构集窗体中通过将窗体属性设置为“ treat document contents as HTML ”形成。这个窗体包括以下的HTML：

```
<frameset frameborder="No" frameborder="0" framespacing=0 border="0"
rows="12%,*">
  <frameset frameborder="No" frameborder="0" cols="99%,1%">
    <frame src="/RMKelleher/WebMailFR.nsf/NoMail?ReadForm" name="Mail"
      frameborder="No" frameborder="0" scrolling="No" >
    <frame src="/RMKelleher/WebMailFR.nsf/Cookies?ReadForm" name="Notif"
      frameborder="No" frameborder="0" scrolling="No" >
  </frameset>
```

```
<frame src="/RMKelleher/WebMailFR.nsf/($Sent)?OpenView" name="Page"
  frameborder="No" frameborder="0" scrolling="Yes" >
</frameset>
```

这段HTML建立了一个包含上、下两部分的帧结构集。顶部的帧占用了整个屏幕高度的12%，剩余的为下边这个部分。而上边的帧结构集又是一个被分为两部分的帧结构集。其中将左边的命名为Mail，占用了这个帧结构集宽度的99%。右边被称为Notif占用1%的空间。大致的表格如下：



在默认的情况下，帧Mail显示信息“ You don ' t have new mail ”的窗体。而帧Notif包含cookie窗体，它含有邮件通知系统。帧Page在默认情况下显示用户的发信箱文件夹，但你点击其他帧或视图时这其中的内容也会随之改变。

过去版本中的方法

如果你使用的是Domino5.0，那么最容易的办法是通过使用新的帧结构集设计元素来创建帧结构集。在版本4.6中可以在窗体属性中设置成“ treat document contents as HTML ”，还可以将<FRAMESET>标签放到文档中。这种做法允许你将<FRAMESET>标签放置在HTML页面的顶部。

在4.5的版本中，这些特点是不具备的。惟一的方法是在<FRAMESET>标签中用一个公式计算出一个名叫“ \$\$HTMLHead ”的域，并将其添加到HTML<HEAD>中。当然，书写一个返回<FRAMESET>标签的公式比书写标签本身要难得多。如果是直接书写公式便是这样：

```
<FRAMESET COLS="50%,50%">
<FRAME NAME="Left" SRC="http://www.acme.com/page1">
<FRAME NAME="Right" SRC="http://www.acme.com/page2">
</FRAMESET>
```

如是书写返回域的公式就不得不像这样：

```
"<FRAMESET COLS=\"50%,50%\">" +
"<FRAME NAME=\"Left\" SRC=\"http://www.acme.com/page1\">" +
"<FRAME NAME=\"Right\" SRC=\"http://www.acme.com/page2\">" +
"</FRAMESET>"
```

如果是这样的使用方法，将Domino作为Web发展的工具来使用时，反而是将简单的事情变得复杂了，使我们更加迷惑究竟Domino至此发展了多少。尽管如此，多掌握几种生成帧的方法有时也是有用处的；如果新的方法由于某种原因实现不了，那就只有使用旧的方法了。

4. 关于Cookies form

这个解决方案的核心是放置了一个 cookie窗体。这个窗体是用 JavaScript来检查并比较你的收信箱的邮件数目和 cookie中信息的数量。如这个数目大于 cookie的值，脚本语言就会给出关于新邮件的消息。

什么是cookie

Cookie是一条由 Web浏览器贮存在你机器上的信息。通常，一个 Web服务器最后能得到这则信息，但是只适用于同类型的服务器产生的。换句话说讲，Microsoft不能偷看由 Netscape产生的Cookie，反之亦然。最简单的方法是用 JavaScript建立和设置Cookie。Netscape Navigator将所有被激活的 cookie存储在一个文件中（通常命名为 cookies.txt）；在IE中将每一个cookies 放置在\Window\cookies路径下的不同的文件中。

这是cookies窗体的第一行HTML代码：

```
<META HTTP-EQUIV="Refresh" CONTENT=20>
```

HTML<META>标签有多个用处，其中一个说明用 Internet搜索引擎文件的关键字。当使用HTTP——EQUIV属性进行刷新时，<META>标签也会被客户端刷新。而CONTENT属性是说明在刷新（在这种情况下一般为20）和时间间隔已经过去后有选择地装载一个URL中的记录数目。如果当时没有URL可装载，当前的页面将会被默认地刷新。属性值“CONTENT=20”相当于：

```
CONTENT="20;url=http://server/database/Cookies?OpenForm"
```

每当页面被刷新时，页面的JavaScript就执行一次，JavaScript代码插在<script language="javascript">和</script>两个标签之间，这段脚本语言包括三个函数：getMessages(), getExpParam(), saveCookie()。这些函数只有被调用才会执行。那些不属于函数的代码行在页面被装载时就会执行。这些行用黑体字写在页底的显著位置，主要是根据情况调用所需的函数。

```
<script language="javascript">  
  cookieName = "Messages";  
  var base = new Date(0)  
  dateAdjustment = base.getTime()  
  function getMessages() {  
    if(document.cookie) {  
      index = document.cookie.indexOf(cookieName);  
      if (index != -1) {  
        countbegin = (document.cookie.indexOf("=", index) + 1);  
        countend = document.cookie.indexOf(";", index);  
        if (countend == -1) {  
          countend = document.cookie.length;  
        }  
        count = document.cookie.substring(countbegin, countend);  
        return(count);  
      }  
    }  
  }  
  return (null);
```

```

}

function getExpParam(cookieName) {
    var expDate = new Date();
    var dateInSecs = expDate.getTime();
    var sixHrsFromNow =
        (6 * 60 * 60 * 1000) + dateInSecs - (2 * dateAdjustment);
    var oneWkFromNow n =
        (7 * 24 * 60 * 60 * 1000) + dateInSecs - (2 * dateAdjustment);
    if (cookieName != "tmpCookie") {
        if (cookieName == "6HrCookie") {
            expDate.setTime(sixHrsFromNow);
        } else {
            expDate.setTime(oneWkFromNow);
        }
        return ("; expires=" + expDate.toGMTString());
    }
    return "";
}

function saveCookie(cookieName) {
    var entry = [Number field];
    if (entry != null) {
        var expParam = getExpParam(cookieName);
        document.cookie = cookieName + "=" + escape(entry) + expParam;
    }
}

var oldmsgs = getMessages();
if (oldmsgs == -1) {
    top.Mail.location = "[Dbsource_1 field]/NoMail?ReadForm";
} else {
    if (oldmsgs < [Number_1 field]) {
        top.Mail.location = "[Dbsource field]/NewMail?ReadForm";
    }
}

saveCookie('Messages');
document.write (getMessages());
</script>

```

如果仔细阅读了这段脚本就会了解关于用正方形括弧围绕的 " field "。在窗体中设置这个 NOTES域是为了使域内的值成为脚本的一部分。最重要的是这些都是数值域。这些域会显示用户收信箱信息量的数目。其取得值的公式为：

```
@Elements(@DbColumn("": "NoCache"; "" ($Inbox); 1))
```

当你在浏览器上第一次装载 Cookie页时，脚本语言会检查 cookies信息的值，但这时并没有任何信息。然后会输出 " no new mail " 的信息到 Mail帧中。之后会创建一个 messages cookies并将它的值放到数据域中。随后，当 Cookie窗体被刷新时，脚本语言就会将数据域中

的值与cookies的信息数目作比较。如果 Messages Cookies中的数目少一些,那么脚本语言就会得出结论:从cookies最后一次的刷新起已经至少有一份新邮件到达。这时会在帧 Mail中显示“new mail”信息。

这个Mail notifier的特征就是这个章节的最后的的地方。注意到新邮件后,当你一打开收信箱时之后脚本最前面的部分被作为文档装载:

```
"<script> cookieName="Messages\"; function
SetCookie(){document.cookie=cookieName +\"=-1;expires=Tuesday,
01-Apr-1999 09:00:00
GMT\";}</script>"
```

这个公式是窗体组成域 \$\$HTMLHead 的一部分。这个公式可能的改进是以计算的方式计算截止日期代替了绝对定值。例如,如果想要详细说明 1999,你需要输入一个类似于“@Year+1”的公式。这种方法让你不用总是不得不记住在 1999年5月去修改。

所有的脚本语言都定义了一个 SetCookie()的函数;但却不是都被调用了。它只有一个HTML的属性公式中被调用:

```
"OnLoad = \"SetCookie()\""
```

这段HTML将成为<BODY>标签文档的一部分。当<BODY>中的“onLoad”事件发生时,SetCookie函数就被激活。如果你把以下的这段脚本语言放置在页面的顶部也会起到相同的效果。

```
<script language="javascript">
document.cookie = "Messages =-1;expires=Tuesday, 01-Apr-1999
09:00:00 GMT";
</script>
```

当这段代码被激活时,cookie的值被置为-1。不久以后,当Cookie窗体的这段脚本再次被装载时,它就会检查到这个值,并且在 Mail窗体中显示出“no new mail”的信息。在这个脚本环境中,如果你收到新的邮件,它会显示提示信息告诉你。

这个应用是说明帧相当有用的一个很好的例子。因为通知信息和 JavaScript都在不同的窗体中,用户能在没有任何打扰下轻松自若地使用邮件数据库。用户在一个帧中处理他们的工作:阅读和发送邮件,检查日历等等,而此帧不会被其他两帧的内容所影响。

5. 谁使用这些特性?

在这些样式中,尽管使用帧和 cookies比使用代理更好,但当你有大量的并发用户时,这种解决方案仍然可能使服务器负载过重。

4.3 使用大纲

在前面的部分中,我们描述了一个站点内假想的问题:你想在站点内的每一页面中都包含有站点内其他页面的链接,但是却不想分开为每一页去做一个链接;你想一次一批地去处理。

这就是使用大纲的缘由。在每一个数据库中,可以设计这样的大纲元素,它们由一系列到数据库的主要元素的链接组成。一个大纲项可以指向一个命名元素(比如,一个视图或表单),一个URL资源,或者一个粘贴式链接(比如,连接到一个特殊的文档)。此外,每一个大纲可能包含一个图标,图表4-8显示了示例数据库大纲元素。

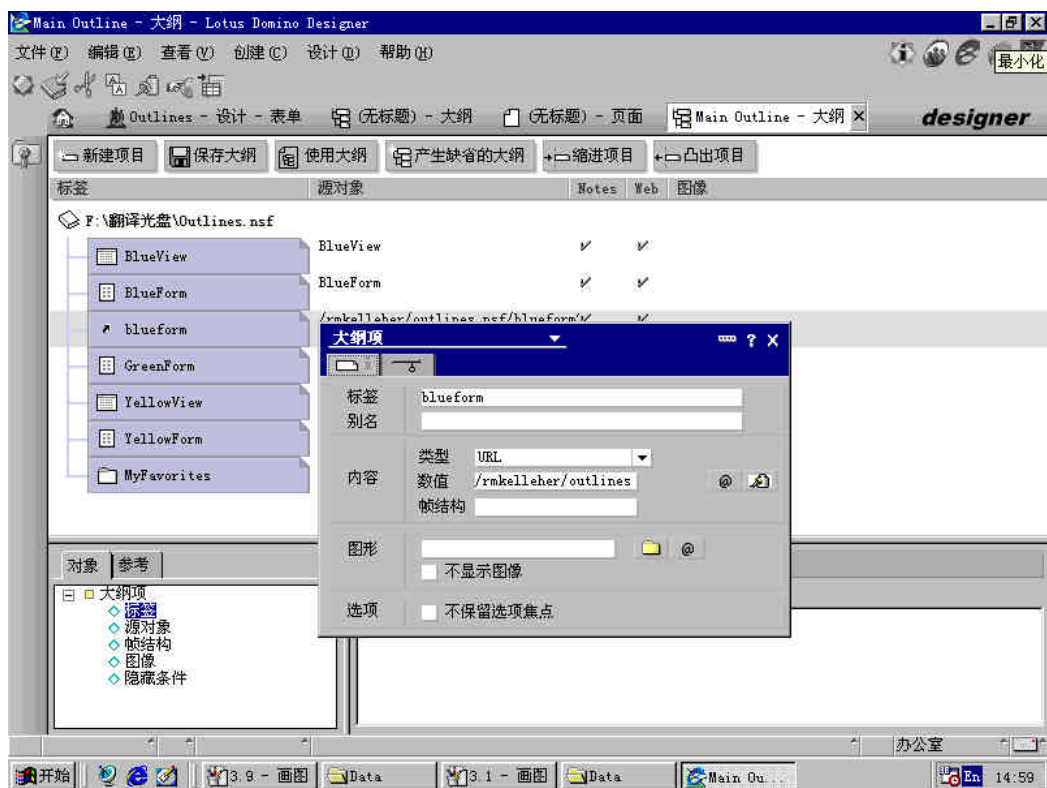


图4-8 在你的站点中的一个大纲帮助用户导航

创建大纲的最快捷的方式是轻轻单击“默认大纲”按钮，这个按钮创建一个由数据库中所含视图和文件夹组成的大纲，你能通过添加、编辑、删除大纲项来修改这个大纲，也可以通过上下拖动大纲项来重排大纲中的元素的次序，通过单击缩进和凸出按钮来缩进和凸出大纲项。一个大纲能被用于一个简单的数据库，或者整个站点，不必为所有大纲项指定同一个数据库内的资源，事实上，甚至可以创建这样一个大纲项，它链接到其他站点内的一个 URL 资源。

4.3.1 同时使用帧和大纲

一旦创建一个大纲以后，就能在表单和页面内插入这个大纲。一个快捷方式是在一个页面内插入一个大纲，然后将此页面内置于一个帧内。在帧结构属性框内，可以指定其他的帧作为默认的链接对象。图 4-9 显示了在 outlines.nsf 数据库中包含有帧、大纲的帧结构集，在这个示例中，outlinepage 的页面内插入了一个大纲，而这个页面又在帧 Frame1 中显示，在 Frame1 的帧中，使用帧 Frame 又作为它的默认的链接对象。

在同一个大纲中能够显示垂直滚动条。图 4-10 显示了数据库 Outlines.nsf 中的帧结构集：Frameset Outline3。在按钮帧内，是包含有相同的大纲元素的。在此页面内，插入的大纲控件被用来显示水平滚动条，尽管是同样的大纲，那些大纲控件能够用来显示从页面到文本的各种不同的元素。

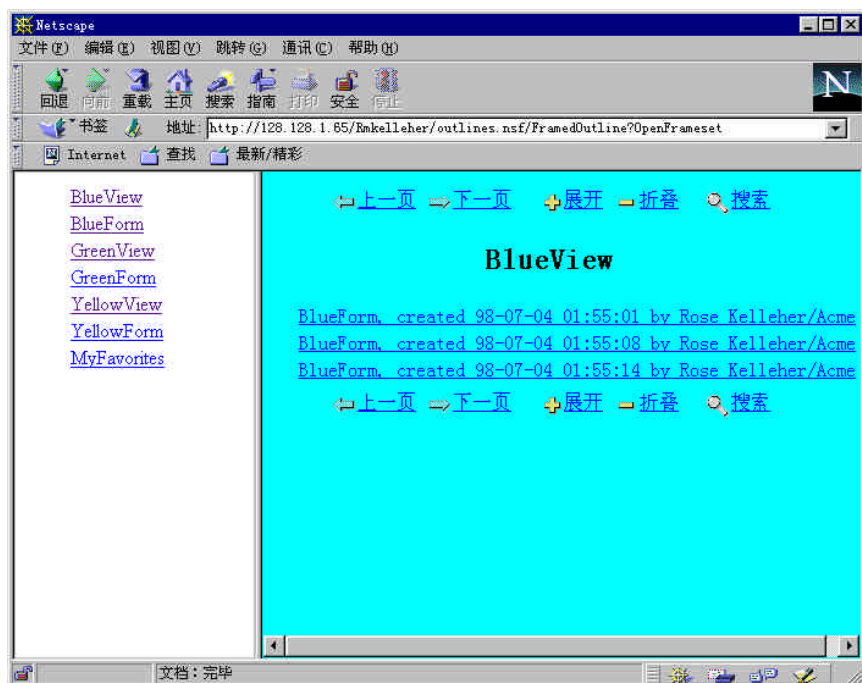


图4-9 在整个数据库打开时你可以使用帧中的大纲提供一个导航帮助

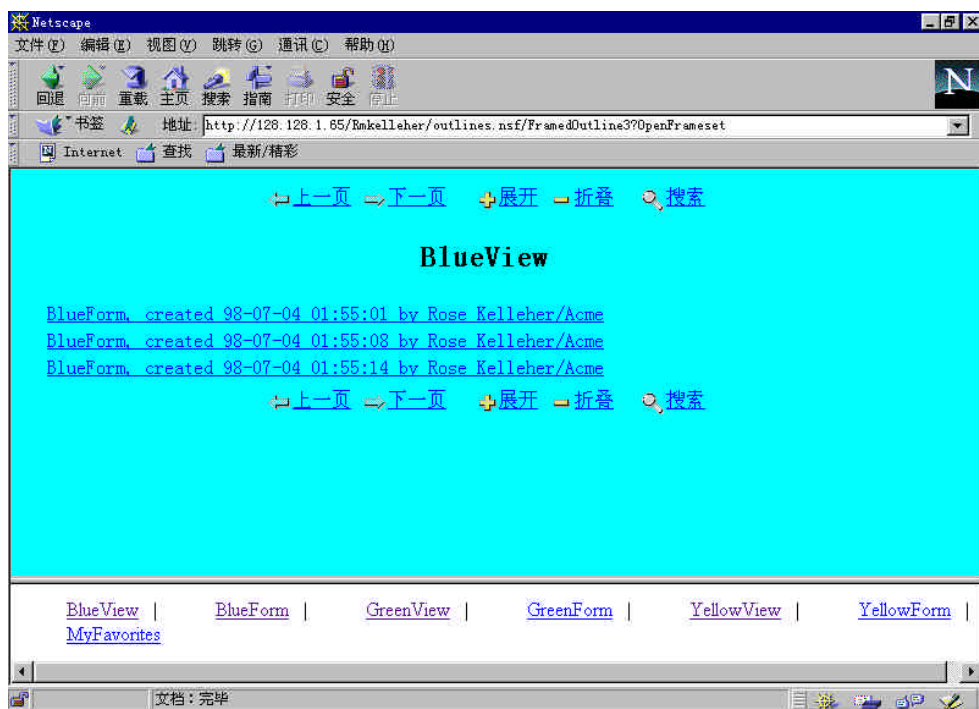


图4-10 根据你对嵌入的大纲控制的设置同样的大纲可以垂直或水平地显示

4.3.2 显示大纲小程序

大纲控件嵌入到一个页面或表单以后，能够配置这个控件使得在 Web浏览器中显示Java小程序，既然视图小程序可使 Web视图运行时有如 Notes客户端的视图，则大纲小程序的设计可使得Web大纲更像Notes客户端的大纲。

4.4 使用其他的设计元素

1. 页面

如果在开发 Web时使用的是以前版本的 Domino。有一个非常重要的消息让你惊喜，请坐下来继续往下读：你不必再为任何东西使用表单了！

引入页面元素的主要目的是使那些不熟悉 Notes的Web开发者对 Domino不感到陌生。在页面元素被引入之前，Domino开发者为了创建一个 Web页，必须先创建一个表单，然后使用这个表单来创建一个文档。

对Notes开发人员来说会产生这样一种感觉：毕竟，表单是数据库设计的一部分，而使用表单创建的文档是数据的一部分，但是，对于一个习惯于使用 HTML的典型的Web开发人员来说，这是一个令人讨厌的烦事，必须做完第一步以后才能做第二步。

现在，有了页面这种设计元素，我们可以认为 Domino表单仅仅只是表单，而不是为了与在可编辑域中输入数据完全无关的事去使用这些表单。

例如，假设你想为用户创建一系列的帮助文档。在 R5之前版本中，必须先创建包含有帮助信息的表单，然后，在每个表单中递交按钮。跳出循环并隐藏，而在 R5中，你的操作进程更为直接：只需简单地创建一个页面即可。简单即是漂亮，是吗？

在一个页面内，除了不包含域和子表单，它的所有设计元素和表单一样；可以使用计算文本来代替域。页面内的计算文本类似于表单中的“显示时计算”域。

2. 导航器

在Domino中创建的导航器在使用 Web浏览器显示时变成了一幅图片。导航器类似于大纲，它用来给用户一个应用中或一个站点内导航。不同之处是导航器的形式更为自由：在导航器内可以包括文本框，图形和按钮。可以定义一个热点区域，当用户点击它们的时候，激活操作（指向Web中的URL资源）。最容易创建导航器链接的类型是按钮。创建按钮时，按钮的大小决定了热点的范围。例如，如图 4-11所示的简单的导航器。它包含三个按钮，每个按钮使用“简单操作”，当点击时打开不同的视图。

在Web浏览器中显示一个导航器，可以使用 Open Navigator命令：

`http://server/database/navigator?Open Navigator`

下面是图4-11所示的导航器的HTML语言：

```
<MAP NAME="19A.map">
<AREA SHAPE=rect COORDS="7,97,173,133"
HREF="/RMKelleher/Outlines.nsf/YellowView?OpenView">
<AREA SHAPE=rect COORDS="6,50,173,85"
HREF="/RMKelleher/Outlines.nsf/GreenView?OpenView">
```

```
<AREA SHAPE=rect COORDS="6,6,174,40"  
HREF="/RMKelleher/Outlines.nsf/BlueView?OpenView">  
</MAP>  
<A HREF="/RMKelleher/Outlines.nsf/af9c60e0c94a37fb802566360074a917?Open  
Navigator">  
<IMG  
SRC="/RMKelleher/Outlines.nsf/af9c60e0c94a37fb802566360074a917/  
$NavImagemap/0.52?OpenElement&FieldElemFormat=gif" WIDTH=174  
HEIGHT=133 USEMAP="#19A.map"  
BORDER=0 ISMAP></A></BODY>
```

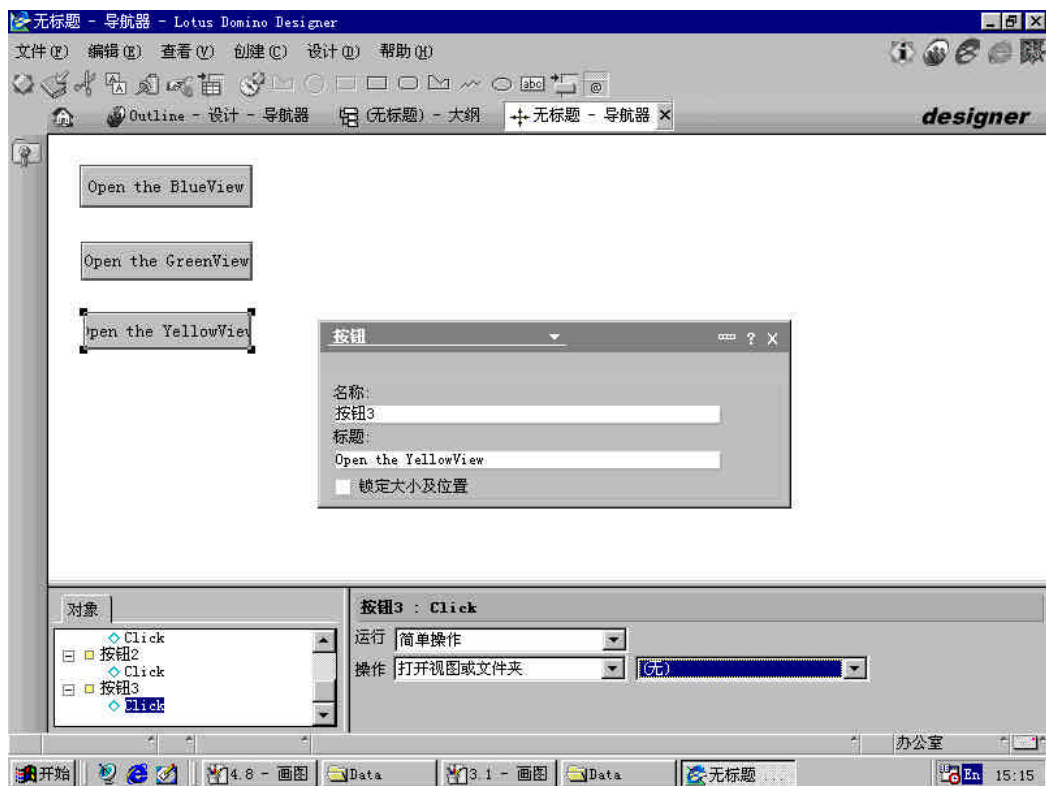


图4-11 导航的最简单的方式是创建一系列的按钮来出发简单操作

创建按钮时，可以单击的热点区域由矩形的大小和位置来决定，也可以在页面上先创建图形和文字，再在其上划热点区域来创建热点。可以为每一个热点区域定义 script，公式或简单操作，当用户单击热点区域的时候激活这些文件（注意：在 Web 中不支持 LotusScript 热点）。

也可以在一个表单、页面或文档内创建图形式的热点，它们由 Domino 转化成 <A HREF> 链接。为什么要创建一个导航器？导航器的一个优点是要以使用一幅图像而不是一系列的图标，就连接到特殊的区域。例如，在一个为全美国不同地方提供天气信息的站点内，可以使

用一幅美国地图，利用热点为每个区域连接到不同的页面，如图所示，图 4-12，在简单的图片中，使用导航器创建链接到各个区域中，可以在表单、视图和页面内插入导航器，使得在一个中心区域只留有一个拷贝。

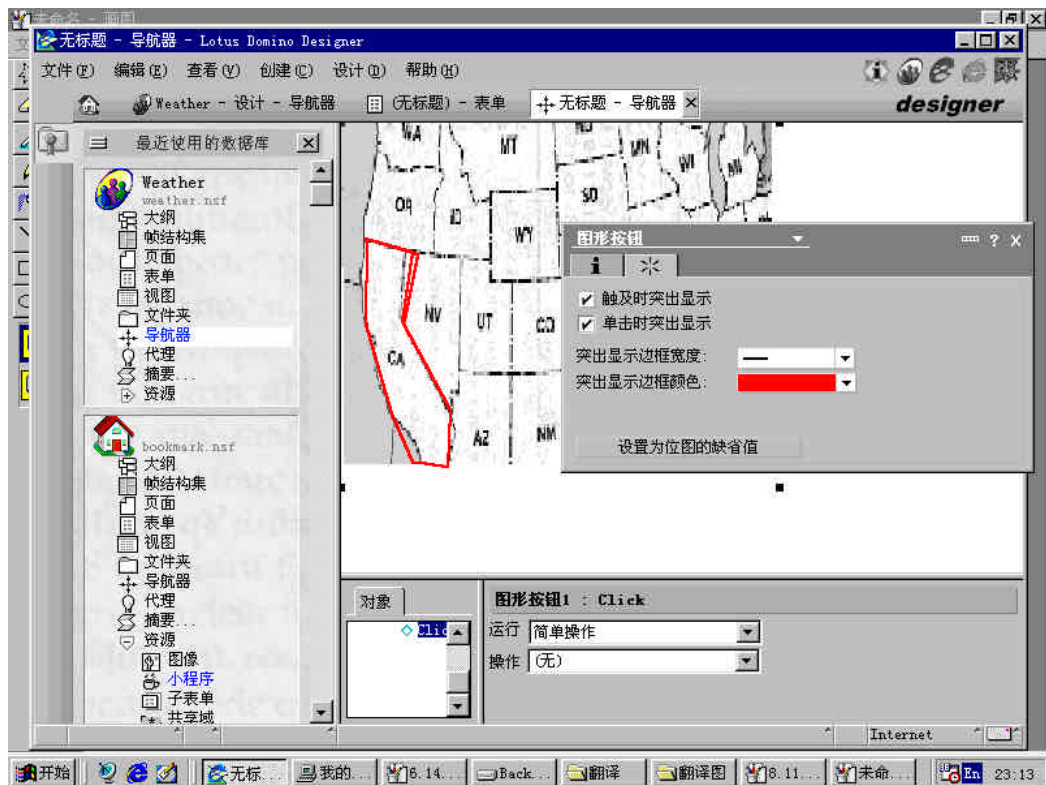


图4-12 一个导航器使你在一个简单的图形中创建一个多边形热点

在导航器内创建热点时，并不局限于只使用矩形。可以创建一个和你想链接区域的外形非常相似的热点。回到上面的美国地图的例示，假设为加利福尼亚州创建一个热点，则可以使用导航器设计菜单中的热点多边形来操作，一个多边形包括若干条直线，而一系列的短直线可以组成一条曲线，在最后显示的图像中，Domino将热点转换成适当坐标。

关于导航器的最后一条建议：在 Web 浏览器中显示导航器时，必须将 HTML 和导航器的图片一起下载，导航器越大，图片也越大，既然下载一幅图片的时间远比下载文字的时间长，这更加适合于在本地的 Intranet 中使用，而不是用于那些全球各个地方的用户来访问的公共 Web 应用程序。既然图片不会损害用户，则在基于文本的导航式帮助文档内增加一些图片就是一个好主意了。例如：在 HTML 大纲之中。

3. 图像资源

这是一句谜语：导航器是图像资源，但图像资源并不是导航器。

当 Domino 将导航器转变成 HTML 时，它就变成了一幅图片。但是在 R5 中，不必创建导航器就可以轻松地创建一幅图片，在任何一个 RTF 文本域中插入一幅图片（想起一个表单或页

面设计本身即是一个RTF文本域), 则新菜单选项立即变得可用, 让你在此图片中创建各种类型的热点, 之后, 当在Web浏览器显示页面时, 图片和热点被转换成图像资源, 图4-13, 单击这幅图片的某个地方, 激活一个JavaScript URL, 显示了在Form Examples.nsf数据库中的ClassPicture表单, 在表单中引入了一幅JPEG图片, 在图片的一小块区域内设置了一个“多边形热点”, 单击此区域激活下列URL:

```
javascript:alert('What a geek!')
```



图4-13 当点击这个图形的某个区域的时候激活一个JavaScript URL

比起导航器, 图像资源的更大好处是, 与导航器不一样, 它并不依赖于设计元素, 最终用户能在其编辑的文档中创建自己的图像资源, 对于出版Web应用来说, 这是一个重要的特点: 用户和数据库设计人员是一些不同的人。

4. 插入元素

可以在一个Web页面内很容易地插入导航器、视图、大纲和过时的设计元素, 比如文件夹树或视图列表, 只需在设计菜单中选择创建一插入, 并选择想要手稿的对象即可。图4-14, 此页面包含了几种插入的元素, 显示了Outlines.nsf数据库中的ElementPagw页面, 在其左上角插入了一个导航器。在以前的版本中, 要完成此工作, 需要创建一个称为`$$NavigatorBody`的域——它具有向后兼容性。这个域的值是想插入的导航器的名称。注意: 导航器的灰色背景仅仅只能在这个导航器边缘以内保持。如果使用`OpenNavigator URL`来打开此导航器, 则整个页面的背景都变成了灰色。

在右上角插入的元素称为文件夹树。现在, 在R5中引入了更为有用的大纲, 这种设计元素的存在就有疑问了。文件夹树和视图列表一样, 可以通过创建一个`$$ViewList`的域来插入。

它们按字母的排列顺序列出了数据库内的所有视图和文件夹。

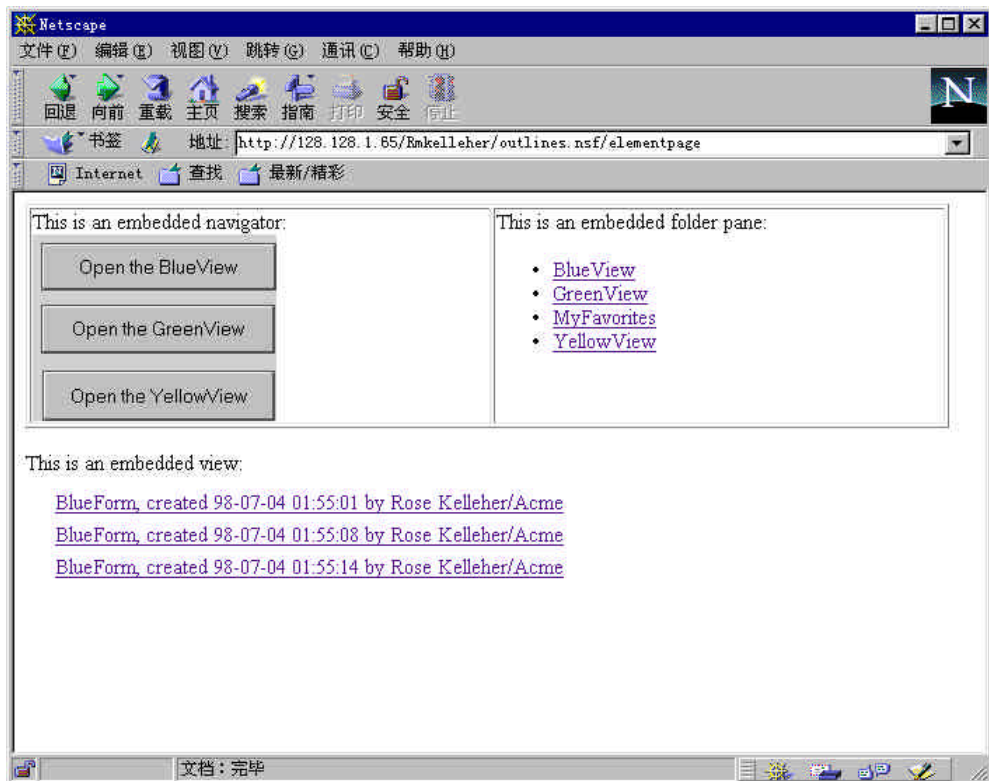


图4-14 这个页面包含几个嵌入的元素

在此页面的中间插入了视图，通过创建一个名为 `$$ViewBody` 的域来插入一个视图，此域的值即为想要插入的视图的名称。

参考信息

在早期的 Domino 版本中，要想获得关于创建帧结构集的更多信息，查看 R4.6 中的帧结构示例数据库（`Framew46.nsf`）。

本章小结

在 R5 中新引入的帧结构使得创建 Web 帧更容易，在每一个 Web 帧内可以显示不同的 URL 内容。使用“默认目标帧”属性使得一个帧链接到另一个帧，对帧内的大纲和视图而言，这种技术非常有用。在较早一点的 Domino 中，在表单内通过使用 HTML 的 `<FRAMESET>` 附件来创建帧结构，此表单被设置成“以 HTML 方式显示文档内容”。

在 R5 中，大纲也是新增加的。大纲比以前版本的老式的文件夹树更为有用；和文件夹树不同，大纲可以包括视图和文件夹以外的设计元素，并能隐藏不想显示的任何元素；在文

文件夹树中的内容总是按照字母的排列顺序来显示而大纲则能根据实际的需求来排列；插入的大纲能被设置成垂直显示（默认方式）或水平显示；在 Java小程序内的表单中也可选择显示大纲。

新的页面元素使得创建 Web页面时：

- 表单，是数据库设计的一部分。
- 不像表单，不包含可编辑的域（尽管有计算文本）。
- 像文档，包含在数据库的全文索引内。
- 不像文档，不涉及到数据（而这是数据库模板的一部分），可以在页面表单或文档中插入视图、导航器、文件夹树和大纲。