COMBEST-DOMINO 新手指南（开发工具、编码语言选择优化、内置对象、限制、错误处理等）

COMBEST-DOMINO 新手指南（开发工具、编码语言选择优化、内置对象、限制、错误处理等）

1.前言

DOMINO 是一个以电子邮件为基础发展起来的标准群件平台，从 1982 年开始发布第一个版本，至今已经有数十年的历史，正式注册用户过亿。

它最大的优点就是提供了大量内置的如 SMTP、POP、LDAP、HTTP、HTTPS 等集成的系统服务，所以使用户快速构建跨平台的解决方案成为可能。

整个平台由 DOMINOSERVER（DOMINO 服务器）、ADMINISTRATOR（DOMINO 管理工具）、DESIGNER（DOMINO 开发设计工具）、NOTES（DOMINO 客户端）组成，在一些针对性应用方面还有 LOTUS-WORKFLOW（DOMINO 工作流设计系统）、LOTUS-DOMINO.DOC（DOMINO 文档管理系统）、LOTUS-QUICKPLACE（DOMINO 协作平台）、LOTUS-LEI（DOMINO 数据交互系统）、LOTUS-LEARNINGSPACE（DOMINO 培训系统）等强有力支持。

2.开发工具分析

2.1 帧结构集

帧结构类似于 HTML 开发中的框架设计，帧结构集是帧结构的集合。帧结构是较大帧结构集的一个区段或窗格，并且可以独立滚动。通过使用帧结构集，设计者可以在帧结构之间创建链接使彼此相互关联。帧结构集可以在用户转向或链接到其他页面或数据库是仍然保持某个页面的显示状态。帧结构一般不采用 3-D 边框，边框宽度设为 0。

2.2 页面

页面是用来显示信息的设计元素。与收集信息的表单不同，页面主要用来向用户展示信息，因此用户不能在页面上创建任何字段或者子表单，但可以创建 HTML 控件。因此页面可以用所见即所得的 HTML 制作工具来进行开发设计。

页面适用于静态信息或者作为其他元素的容器。可以使用页面作为用户应用的开始界面或者作为提交文件后的提示界面。

2.3 主表单

表单是用于在数据库中输入和查看信息的载体。

表单可以包含

1）存储数据的字段　　2）标注字段或者提供指示的文本

3） 存储用户想要在多个表单上使用的表单元素集合的子表单

4） 可以结合图形和字段的布局区域 它们所采用的方法可以提供更大的设计灵

活性

5）可以使表单更容易理解的图形　　6） 汇总或者组织信息的表格

7）对象 OLE 预定 Notes/FX(TM)字段 文件附件 URL 以及可以扩展 Notes 文档范围的链接

8）可以自动执行函数的动作按钮

9）可以强化文档外观的背景颜色和图形

10）在表单中包含其他设计元素的嵌入式元素

在应用上细分 DOMINO 系统中表单一般分为主表单和子表单，而主表单又分为数据存放表单和数据展现表单。

其中数据存放表单一般的设计处理方式是数据字段设计在该主表单中，公共控制信息（如流转控制）设计成子表单（component）。再将该子表单加入到主表单中。确保整体设计结构清晰，在内容上主要包含的内容有：数据信息、按钮操作、用户界面、功能子表单等。

而数据显示表单的设计处理方式是用于在浏览器中进行数据浏览的表单，为了保证界面的友好性，一般采用 CSS 定义整体风格。

2.4 视图

视图是访问数据库中文档的入口，每一个数据库至少必须包含一个视图，基于所选择的准则，视图可以显示数据库的文档子集或者所有的文档。基于文档的内容，也可以对文档进行分组和排序。

在创建视图的之前一定要：

* 视图有一个中文名称和一个英文别名，在代码中始终引用英文别名

* 视图的列样式、列头字体、颜色、大小在同一个数据库内应该统一

* 视图标题栏高度，行间距应该统一设定好，一般设定为 1 和 1 1/4

* 如果视图需要分类折叠显示，要出现可折叠标记

* 视图上的操作应该出现在操作栏的左侧

* 视图应该行数应该使用交替颜色显示

* 视图列的最后一列应该扩展到窗口宽度

* 用户不直接查看的视图应定义为隐藏视图

* 如果视图要嵌入到页面，用于 Web 访问，则应该选定“在浏览器中使用小程序”

2.5 文件夹

文件夹是用来存储文档的容器。文件夹与视图具有相同的外观，而且其设计方法也与视图大致相同。其区别仅在于应用的时候视图具有可以自动选择并显示文档的文档选择公式，而文件夹则不是，它是通过用户手动的添加来显示文档的。

所以在设计的时候，其设计知道方法可以大致跟视图相类似。

2.6 共享代码

2.6.1 代理

代理 Agent 可以让用户在 Domino 中自动执行许多任务。它们是可以在数据库中为用户执行特定任务的独立程序。例如可以归档文档、改变字段值、发送邮件消息、删除文档或者执行与外部应用进行交互这样的功能更为强大的动作 。

代理还能够进行设置在服务器上基于安排或者在出现特定事件时自行运行。

2.6.2 WEB 服务

Web 服务和 Domino

Domino 6 是一个理想的应用程序，用于宿主或使用 Web 服务。由于 Web 服务主要是由 XML 数据组成。因为 DOMINO 目前有更多被设计用来特殊处理 XML 的 LotusScript 类，因此 DOMINO 对 WEB 服务有天生的好支持。

也有一些有用的工具可提供对 Domino 里 Web 服务的附加的支持，它们是：

对 LotusScript 的 SOAPConnect-这个工具包含了一个 LotusScript 库它允许您使用和宿主 Web 服务。

MS SOAP 工具包-由 Microsoft 提供的允许 Domino 在 Windows 平台上使用或宿主 Web 服务

.NET-来自 Microsoft 的一个工具集让您使用和宿主 Web 服务这个工具集可

由 Lotus Notes/Domino 经由 Common Object Model (COM)接口访问。

以下的段落接着描述 WEB 服务的本质。Web 服务可以被定义为一个应用程序它提供了一个 API，以便将自己与其他应用程序集成在一起。Web 服务的主要功能是提供过程到过程的交互作用而不需要用户界面，也就是说您可以通过与 API 接口远程调用这个应用程序，调用这个服务的应用程序被称为客户机程序。

XML 位于 Web 服务的核心并为描述远程过程调用 Remote Procedure Call Web 服务以及 Web 服务目录提供了一种公共的语言。

Web 服务这个短语有时可能会容易误解好像它意味着使用 Web 浏览器，然而并不总是这种情况，有许多不同的调用 Web 服务的方法例如一个 HTTP 请求或者来自另一个应用程序的一个电子邮件是其中的一些方法。

调用一个 Web 服务的最常用的方法之一是通过发送一个 HTTP Get 请求到 API。经由 Internet 访问 API 有许多优点，API 可以被任何具有 Web 服务地址的全球客户机访问，Web 服务应用程序里的一个修改只需要在数据源完成，Web 服务可以以任何语言和在任何平台上书写，只要那些 Web 服务根据 Web 服务标准可访问，

为了使协同性有效，Web 服务平台必须提供一个标准系统，该系统将与使用不同的平台和/或编程语言的系统有接口。

一个 Web 服务平台需要描述此 Web 服务并提供其他应用程序为了调用这个 Web 服务所需要的信息。构成 Web 服务平台的主要技术如下：

XML-这是用于在 Web 服务平台上表示数据的基本格式；

SOAP-简单对象访问协议 (Simple Object Access Protocol) Web 服务的远程过程调用(RPC)工具 SOAP 是一个传输协议，它使用 HTTP POST 请求来传输。方法所返回的响应是一个 XML 文档。

WDSL Web 服务描述语言 Web Service Description Language-是一个基于 XML 的文法，它用于描述 Web 服务及其函数参数和返回值。

UDDI-通用描述发现和集成(Universal Description, Discovery, and Integration)

一个基于 XML 的目录它表示一种技术规范用于发布和发现业务和 Web 服务。

2.6.3 大纲

每个数据库都需要一种方法进行导航。可以利用大纲设计起来创建大纲，给用户提供一个自动链接，具有个人风格的站点导航图。大纲可以包含背景图形、定制图标、链接或者操作。

所以在设计大纲的时候，对大纲采用的大纲项图标，以及大纲项所采用的字体，字号，颜色，mouseover 的颜色等都要事先定制，颜色，图标采用不一定要相同，但是应该在保证风格统一，并且与整体界面没有冲突的前提下，体现出各自的特点。大纲项与大纲都应该采用别名的形式，在程序调用中调用英文别名

2.6.4 子表单

子表单 subform 是用户可以在多个表单中使用的表单片段。例如用户可以在子表单中建立公司的信头，然后在各种商业表单中使用这个子表单。

子表单能够包含与常规表单相同的元素，可以基于公式在表单上加载子表单。

2.6.5 共享域

共享域类似于字段，但是可以在不同的表单中使用，如果用户改变共享域的属性，那么这些改变就会应用于所有出现这个字段的场合，他的主要作用是域进行一次定义而后重复使用。便于设计改动时的工作。

2.6.6 自动化组件

向应用程序中添加自动功能可以加速执行重复任务、路有文档、更新信息、执行计算、运行程序以及检查错误的自动化组件：

* 操作按钮

通过点击它们,可以使某些任务得以自动完成。特别是对于 Web 浏览器用户，需要使用操作来模拟 Notes 菜单项。

* 热点

热点是用户单击后可执行操作、运行公式或 Script 以及转向链接的文本或图片。热点是可以到另一个 Web 站点、数据库或数据库元素的链接，还可以是按

钮、弹出式文本或公式以及操作。

2.6.7 共享操作

共享操作是在表单或视图中设置用户激活的任务。共享操作的设计位置虽然与操作不同，但是设计方法两者没有什么区别，对操作设计的一些规定，在此同样适用。

2.6.8 SCRIPT 库

脚本库是集中存放共享代码的位置具体可包含 LotusScript、JavaScript 以及 Java 库。

2.6.9 导航器

导航器是用户能够包含可以用于导航的可编程区域和热点的图形。热点通常可以指示用户前往数据库或者 Web 站点的另外部分。但是总的来讲，导航器是 NOTES 4.X 版本的产物，是一项过时的技术，所以在开发的时候不再建议开发人员使用导航器，而去使用页面嵌入大纲来对应用进行导航。

2.6.10 层

层 layer 这种设计元素不能在数据库级别建立，而是要在页面表单或者子表单中建立。层可以让用户在页面表单或者子表单上放置重叠的内容块，因为用户能够控制信息的位置大小和内容。所以层能够让用户进行灵活的设计，用户能够建立和堆砌多个层使其彼此交错。透明层可以展现其下的层不透明的层可以隐藏其下的层。 层的内容要依赖于用户要在页面还是表单上建立层，当用户在页面上建立层的时候层就可以包含与页面包含内容相同的元素，例如用户能够增加文本和图形等内容；当用户在表单上建立层的时候层就能够包含与表单包含内容相同的元素例如用户能够增加文本和图形 以及受控的访问部分 字段和子表单

2.6.11 DXL

Domino 数据的 XMLExtensible Markup Zarauage 可扩展标记语言表示称为 DXL。

DXL 可以描述特定于 Domino 的数据以及嵌入式视图表单和文档这样的设计元素。随着 XML 成为交换信息的标准基础，DXL 也为将数据的 XML 表示导入和导出到 Domino 应用提供了基础。利用 DXL 实用工具用户就能够查看和导出用户的 Domino 设计元素，用户还可以使用转换器 Transformer 实用工具和 XSL 样式表单文件将其转换为另外的格式，XSL 文件包含了针对 XML 数据的格式

3. 开发要点编码语言以及代码优化分析

3. 1 编码语言的选择

在具体的开发过程中，根据需求性质，DOMINO 可以有多种编码语言的选择。主要有公式、lotusscript、java、javascript 语言，比较起来，特别简单的功能和逻辑，用公式实现比较快；复杂的功能和逻辑，用 lotusscript 比较快。

java 语言可以实现多线程，完全面向对象，处理大量文档时，如果程序利用到多线程，比不能利用多线程的 lotusscript 快。

另外针对于底层开发，LOTUS 还有专门的 Toolkits 支持：

按 Lotus 产品系列来分，Lotus 提供下列产品的工具包： Notes/Domino 、Sametime 、QuickPlace 、Discovery Server 、Other。

其中，Notes/Domino 和 Sametime 的工具包比较多，C、C++、Java 语言的都有，其他产品的则很少。从这个角度，也可以看出文档数据处理、协同工作是 Lotus 系列产品的核心价值所在。其中，Notes/Domino 的工具包有：

Lotus C API toolkit Lotus C++ API toolkit
Lotus Domino Toolkit for Java/CORBA Lotus and Notes Toolkit for COM
Lotus Domino Driver for JDBC （简称 LDDJ） NotesSQL
Lotus XML Toolkit（简称 DXL） Custom Tag Converion kit（简称 DCT）
LotusScript Extensions toolkit（简称 LSX）

在 Domino/Notes 的 Toolkit 中，C API 的功能最为强大：可以操纵 notes 数据库中几乎所有的数据对象：数据库及 ACL、文档和域、表单、视图和文件夹、代理、可以为 Notes 客户端的增加附加菜单、可以用来创建附加的 Domino 服务任务、可以用来扩展 Domino/Notes 的事件管理。

而其主要限制如下：

不能修改已有的 Domino/Notes 软件，不能去除其已有的功能、特性，或者改变其工作机理、不能修改安全特性、不能修改用户活动记录

支持的操作系统：WindowsNT/2000, Linux, Solaris SPARC/Intel, HP-UX, MacOS, AIX, AS/400, S/390 等。

3．2 编码优化原则

1）公式语言优化

1.使用@ClientType 代替@UserRoles 来检查客户端类型

从 R4.6 开始，不必使用@UserRoles 中的$$WebClient 来检查客户端是否使用浏览器，直接使用@ClientType 比较快

2.使用简单的公式隐藏条件比较快

以下三种写隐藏的选择，性能递减。可以利用第一个，不要写判断 @ClientType 的公式；可以利用编辑模式，不要写判断@isdocbeingedited 的公式

a.根据客户端程序类型 b.根据编辑模式 c.根据公式隐藏

3.在@dblookup 和@dbcolumn 公式中使用列号比使用域名快

使用域名需要在域列表中对比，但直接使用列号会带来维护的困难。

4.打开@dblookup 和@dbcolumn 的 cache 开关

cache 不但可以使这个公式执行更加快，而且可以使‘同一个数据库’中，

‘同样公式’的另一个 lookup 运行更快。cache 是基于数据库的，不同数据库的相同 lookup 不能利用相同的 cache。注意：无论是否打开 cache，lotusscript 调用 evaluate 的 lookup 都不能利用 cache。

5.根据 4，如果公式和 lotusscript 都要利用相同的 lookup 结果，可以将信息放到 profile 文档中。

6.要 lookup 一个包含很多文档的视图时，创建一个隐藏视图，仅包含必要的列。视图越小，越快。

7.如果要取视图中的几个列，分别取的话，每个 lookup 都要花时间。将几个列组合成一个列，用特别的字符分隔，一次取出来，然后分析出几个域。

8.一段公式中，数次利用同一个结果集，用临时变量保存，而不是每次重新取

9.使用不同的搜索方法，如果处理的文档集较小（例如少于数据库中所有文档的 15%），使用 lotusscript 比公式快，如使用 getview , search, ftsearch 等方法。

10.使用‘显示时计算’域来避免不必要的重复计算。下表是使用各种类型域时，不同事件引发的计算情况：

Field Type	Create	Open (empty)	Open (data)	Show Dialog
Edit (normal)	DV	IT,IV	IT IV	
Edit (K-UF-DB)	DV	KF IT,IV	KF IT IV	KF
Edit (K-UF-Not DB)	KF,DV	KF IT,IV	KF IT IV	
Computed	V	V	V	
Computed for display	V	V	V	
Computed for composed	V	V	V	

上面缩写的解释

normal----Normal edit field (not one of the two special ones that follow)

K-UF-DB---Keyword, Use formula for choices, dialog box

K-UF-Not DB---Keyword, Use formula for choices, check boxes or radio buttons

DV -----Default value formula executes

IT -----Input translation formula executes

IV -----Input validation formula executes

KF -----Keyword formula executes

V -----Value formula executes

Create----Field is being created (either because the document is being created

or an existing document is opened that does not contain the field)

Open -----Opening an existing document — in either Read or Edit mode

Empty---Field is empty (exists in document but has no contents)

data-----Field has data

Show Dialog---User presses the icon to display the dialog box

注意两点：

1.使用 check box 和 radio box，一打开文档，公式总要计算，即使仅仅打算阅读而不是修改文档，看起来是合理的，因为需要计算出各个项目供阅读嘛

2.已经有数据的，使用对话框格式的编辑域，打开时候也要重新计算，这个就不太合理了。

当使用复杂的@dblookup 时，性能有很大影响。隐藏公式域没有作用，因为仍然会发生计算。按照下面方法解决

1）创建一个隐藏的多值‘显示时计算’域：kwDataHidden，计算公式是原来的 @dblookup 公式，但仍然要做如下处理，值公式写成这样：

```
rem "Do not run formula during doc save";
@if (@IsDocBeingSaved; @Return("");"");
rem "Only run if doc is in edit mode";
@if (@IsDocBeingEdited;@DBColumn(.....);"")
```

2）在表单的 PostModeChange 中，写：

@If(@IsDocBeingEdited;@Command([ViewRefreshFields]);"")，（实际上，如果觉得公式的值的修改频率不是太快的话，这里也可以不写）

3）真正操作那个域的公式写：kwDataHidden；并且该域属性选择‘刷新文档时刷新选项’。

2）脚本语言优化

1.使用 For loop，不使用 Do loop 或者 while loop 的循环

前者比后者快 60%（不计算循环内部时间的情况下）

2.引用数组元素的时候，使用 Forall 比 for 快

一维数组，前者快 75%，二维以上，也可以快 50%

3.简化 if 语句

lotusscript 不会象 c 那样进行逻辑语句优化，所以

```
If a=x and b=y then
```

应该写成

```
If a=x then
  if b=y then
```

后者快 40%

4.避免混合变量类型（计算）（也就是不要隐式造型）

```
dim a as single
dim z as long
z = 3
a = 1.0
z = z+a
```

上面这一段使用了 implied casting，z=z+(long)a 会比较快

1.对整数结果，使用整数除法：z& = x& \ y& 比 z& = x& / y& 快 60%

2.仅在有必要的时候才使用 variant 类型，lotus 使用更多的时间处理 variant

3.读取文件的时候，一次读取一块，而不是一行，下面的例子 A 比 B 快接近 5 倍！而且读取的文件越大，越明显。

example A

```
Open fName$ for input as #fNum
buff$ = Input$(Lof(fNum),fNum)
stPos = 1
lineNo = 1
eoFile = false
While Not eoFile
nlPos = Instr(stPos,buff$,Chr$(13))
If (nlPos) > 0) then
fData$(lineNo) = Mid$(buff$,stPos,nlPos - stPos)
stPos = nlPos + 1
else
fData$(lineNo) = Mid$(buff$,stPos)
eoFile = true
end if
lineNo = LineNo + 1
```

```
wend
close #fNum
```

example B

```
Open fName$ for Input as #fNum
lineNo = 1
while Not Eof(fNum)
Line Input #fNum , fData$(lineNo)
lineNo = lineNo + 1
wend
close #fNum
```

注意，R4.6 以及之前的版本，string 变量有 64k 的限制，所以一上例子 a 需要再增加一些东西（不能一次读取，而是要分次读取文件），但块读取的好处仍然是显而易见的。R5 开始，string 变量可以最大达到 2G，足够读取非常大的文件了。例子 A 虽然增加了代码量，但是可以大大加快速度，某些场合非常有用

5.避免无谓地使用数组，下面例子 A 比例子 B 快两倍

Example A

```
For i = 1 to 100
  sum = sum + x(i)
Next
t(1)=sum
```

Example B

```
For i = 1 to 100
 t(1) = t(1) + x(i)
Next
```

Domino R5 引入了新的数组处理函数：ArrayAppend , ArrayGetIndex , ArrayReplace , FullTrim，使用这些函数比自己写相同的功能快

6.有一个不使用字符串数组的替代方法

把所有数组中的字符串元素构造成一个字符串，用特殊分隔符分开（例如回车符），然后用 Instr 来取各个元素

7.优化字符串的处理

字符串的处理非常消耗资源，例如下面这句很常见

```
X$= X$ & "a"
```

这个语句非常消耗资源（java 中有类似现象，因为涉及到重新生成字符串），下面这句缩短字符串也是很消耗资源的

```
x$ = Right$(x$ , currentLength% - lengthStrippeddOff%)
```

8.创建一个新的字符串有时候更加快

下面例子 A 比 B 快 25%

A

```
str$ = stringA$ & stringB$
```

B

```
str$ = stringA$
str$ = str$ & stringB$
```

9.反复处理字符串时，记住处理位置

比不断修改字符串长度快（其实是不断生成新的字符串），下面例子 A 比 B 快 85%

A

```
strPiece$ = Mid$(str$ , startPos% , pieceLength%)
startPos% = startPos% + pieceLength%
```

B

```
strPiece$ = Left$(str$ , pieceLength)
strLength% = strLength% - pieceLength%
str$ = Right$(str$ , strLength%)
```

10.谨慎是用 redim 命令

可以理解，每次 redim，是内存堆数据的乾坤大挪移，所以尽可能事先决定数组大小。实在没有办法，也千万不要在循环中 redim。以下例子 A 比 B 快 20%

A

```
For i = 1 to 10000
  If ( i> iMax) then
    iMax = iMax + 100
  redim Preserve sArray(1 to iMax)
  end if
  sArray(i) = ""
  Next
```

B

```
For i = 1 to 10000
Redim Preserve sArray(1 to i)
sArray(i)=""
Next
```

以上实际是减少 redim 的次数，每次增加多些，循环后可能还需要一次 redim，让数组空间和实际一致

11.在多维数组中循环

应使用正确的顺序。在外循环中操作后面的维数，在内循环中操作前面的维数（这个最奇怪了，估计跟数组的内存排放有关），下面的例子 A 比例子 B 快 400%!（实际测试，500 的 PIII，看不出差别，都很快，当我再想增大数组元素个数的时候，宣称已经超过 32K 限制了,faint!看来意义不大）

```
A
For y = 0 to 2
 for q = 0 to 5000
   z = z + x(q,y)
    Next
  Next
B
For q = 0 to 5000
   For y = 0 to 2
    z = z + x(q,y)
     next
  next
```

12.比较整数，而不是比较字符串

下面的例子 A 比例子 B 快 50%，例子 C 比例子 D 快 30%

```
A    If(Asc(x$) = Asc("A")) then
B    If(Left$(X$,1) = "A") then
C    If(Asc(Mid$(x$ , 1, 1) = "A") then
D    If(Mid$(x,1,1) = "A") then
```

下面比较空串的例子 E 也比例子 F 快 30%

```
E    If (Len(x$) = 0) then
F    If (x$="") then
```

13.将数字转为字符串的时候使用自动造型，不用手工转换

例如 s2$ = "Text" & iNum 比 s2$="Text & Cstr(iNum%) 稍微快一点，不过不明显。

14.重要：避免过多使用环境变量

环境变量要读取 notes.ini 文件，对性能造成影响，替代办法是使用 profile。如果一定要使用环境变量，将多个环境变量合并成一个，用特殊符号分隔，取出来后再分析成多个。取 10 个单独的环境变量要比取一个长的环境变量多花 10 倍时间!

3）Java & JavaScriptino 应用程序优化

Java 需要装入 JVM，client 端第一次运行的时候装入，server 端，随着 http 装入的时候装入（所以不用 http 就不要装入了，估计慢也是受到 jvm 影响

 ）有关 java 没有什么好说，就说两点。

1.将需要的 java 类，尽可能放在类列表前面，尽可能接近 base 类，可以加快搜索速度

2.周期性调用回收函数，java 本身是自动进行垃圾收集的，但是该书的意思似乎说 java 的垃圾回收，不能回收相关 lotus objects 用掉的内存

3.Javascript，没有什么好说的，用于 web 页面，域校验，和输入转换，可以避免 browser 和 web server 来回交换数据，减少网络流量

4）DOMINO 对象模型优化

利用后期初始化和对象重用技术来最大限度地利用资源

1.后期初始化（lazy initialization）。学习到一定地步的 lotus 程序员，为了程序结果清晰，往往喜欢事先将一些对象如 db , view , doc 等初始化，然后每个函数都可以使用。从提高效率的角度，可以改进为使用后期初始化（自己翻译的，不知道是否准确，应用期初始化？），下面是后期初始化的方法

```
' *** Globals
' *** Declare the global data we share
Public gDb As NotesDatabase
Public gVwCustomerType As NotesView
'*** Functions
'*** Get the customer type view (lazy initialization)
'*** Do account validation
Sub Initialize
'*** Set up the database (NOT lazy)
Dim ss As New NotesSession
Set gDb = ss.CurrentDatabase
End Sub
Function GetCustomerTypeView() As NotesView
'*** Only get it the 1st time it is used
If (gVwCustomerType Is Nothing) Then
Set gVwCustomerType = gDb.GetView("Customer Type")
End If
Set GetCustomerTypeView = gVwCustomerType
End Function
Function ValidateAccount(custName As String, _
accountType As String) As Integer
```

'*** Return True if account is valid
D