第一篇

应用基础

本篇导览

本篇主要介绍了一些开发应用程序的基础知识，包括 Application 的创建、窗口的创建、菜单的创建、如何使用系统提供的函数和事件、PowerBuilder 脚本语言的使用等等。本篇是基础，以后的各篇都是在此基础之上进行的。如果您是一位初学者，那么就要仔细地阅读本篇；如果您已经具有一定的开发基础，则可以跳过本篇。

通过本篇中实例的学习，相信您一定能掌握如何进行一个实际应用程序的开发。本章的实例主要介绍了 PowerBuilder 的一些基本的控件、函数和事件的应用，常用函数 Message（）、Beep（）、Open（）、Close（）、类型转换函数、Run（）和 SetForus（）等的使用，还介绍了 PowerBuilder 中的三类事件：Application 事件、Windows 事件和 Menu 事件。

实例 1　为应用程序加密

实例说明

本例将利用 PowerBuilder 7.0 建立一个简单的应用程序，并且为该应用程序加密。本例仅制作一个密码校验窗口，要求用户输入密码。其主要有两个目的：教会读者如何建立应用程序和如何简单地为应用程序加密。本例最终效果如图 1-1 所示。

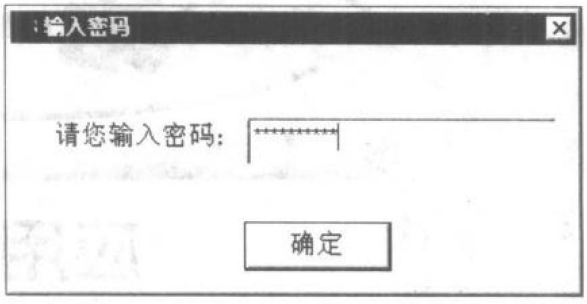

图 1-1　效果图

编程思路

本例主要是在应用程序运行之前弹出一个密码校验窗口，要求用户输入密码，并对用户输入的密码进行处理：如果用户输入错误密码，则提示退出应用程序；否则，进行正常的应用处理。为了能够实现加密功能，需要创建一个 Application。Application 是建立 PowerBuilder 应用程序的前提，即所有的 PowerBuilder 程序都是通过调用 Application 来实现的（就好像 C 程序调用 main()函数一样）。而创建密码校验窗口则是本程序的核心部分。

创作步骤

一、建立 Application

1．打开 PowerBuilder 7.0 应用程序，弹出“PowerBuilder requires an application”窗口，选取“New”页面下的“Application”图标，并单击“OK”按钮。

“PowerBuilder requires an application”窗口共分三个页面：New、Browse、Recent。其中“New”页面用来新建程序，其内包含了“Application”、“Template Application”、“Jaguer Component”等图标，分别用来建立不同类型的应用程序。我们建立的应用类型大部分都属于 Application；而“Browse”页面主要是通过浏览的方式打开 PB（PowerBuilder）库文件（*.pbl）。PB 应用程序的所有内容，包括 Application、窗口、数据库、表、用户对象等等存放在 PB 库文件中，这样有利于用户管理。“Recent”页面列出了 PB 最近使用过的库文件。如果我们经常对一个应用程序进行修改，那么“Recent”页面非常方便。

2．在弹出的“Specify New Application and Library”窗口中，要求用户指定所要建立的 Application 名称和所存放的库文件名称。在“Application Name”文本框中输入应用的名称“sample1”；在“Library”文本框中输入库文件的名称“C:\PB\sample1.pbl”。按照 PowerBuilder 的缺省规定，库文件名称与 Application 的名称相同。单击“Finish”按钮。

二、创建加密窗口

1．单击“File→Open”菜单命令，将会弹出一个“Open”窗体，要求用户选择所要打开的 Application 名称。

其中，最下面的“Object Type”下拉列表列出了所要打开的各种对象的类型，这里我们选中“Application”；“Application Libraries”中列出了当前路径下的库文件；“Comments”中显示了对所选择的应用的注释；在上面的文本框中列出了所选库文件中所有的 Application 名称；“Object”中显示的是选择的 Application 名称。这里我们选择刚刚建立好的“sample1”。单击“OK”按钮，PowerBuilder 将打开该应用程序。

2．单击“File→New”菜单命令，在弹出的“New”窗体中选择所要新建的对象类型。选择“Object”页面下的“Window”图标，如图 1-2 所示，建立一个窗体对象。单击“OK”按钮。

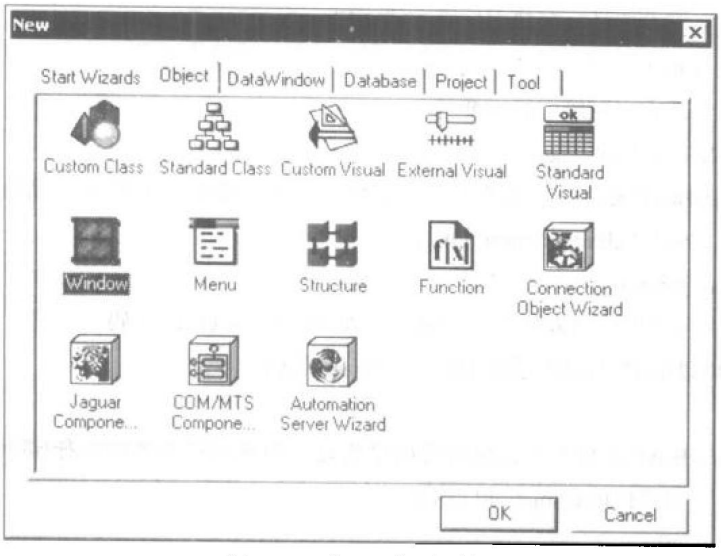

图 1-2 “New”对话框

该窗体共包含六个页面：“Start Wizards”与前面的“Application requires an application”窗体中的“New”页面相同；“Object”页面用来新建各种对象，例如窗体、菜单等；“DataWindow”页面用来建立各种形式的数据窗口对象。“Database”页面用来建立数据库；“Project”页面用来为应用程序建立工程；“Tool”页面用来建立各种工具。

3．打开窗体的属性页，在“Title”中输入“输入密码”作为窗体的标题。取消“MaxBox”、“MinBox”和“Resizable”前面的复选标志，使窗体大小不可改变。

4．在窗体上添加“StaticText”、“SingleLineEdit”和“CommandButton”三个控件。将“StaticText”的“Text”属性设为“请您输入密码：”作为提示信息；将“SingleLineEdit”的“Text”属性置空，选中“Password”前面的复选标志，这样用户在输入密码时，单行编辑框中显示的只是一串*，这样可以防止其他人在用户输入密码时偷窥；将“CommandButton”的“Text”属性设为“确定”。其余各属性值均保持缺省。对控件进行合理的布局。

在窗体上添加控件有两种方法：第一种为单击“Insert→Control”菜单，从中选择所要添加的控件名称；第二种方法是单击图标 OK ▾，打开下拉列表，从中选择控件。

5．保存窗体，命名为“w_mima”。

三、添加脚本

1．为 Application 的 Open 事件添加脚本　应用程序一开始运行，首先执行的就是 Application 的 Open 事件，所以在该事件中我们可以加一些应用程序的初始化内容。本例中我们只是打开密码窗口，至于密码校验则在密码窗口中进行。在 Application 的 Open 事件中添加如下脚本：

```
open(w_mima)
//打开密码窗口
```

单击“Save”图标，对所输入的脚本进行保存。

2．为密码校验添加脚本　打开刚刚建立的密码窗口。程序在运行时，要求用户输入密码，并按“确定”按钮进行密码校验。故在按钮的 Clicked 事件中添加如下脚本：

```
string strMima
//定义一个字符串变量，存储用户输入的密码
strMima=sle_1.text
//将用户输入的密码赋给字符串变量
If strMima="zhanglixin" Then
    MessageBox("您输入了正确的密码","您建立的应用程序正在运行!请按确定按钮退出应用
        程序!",Exclamation!,OK!)
    close(w_mima)
    //以上两个语句用来处理用户输入正确的密码后执行的代码
    //您可以根据自己的实际需要添加不同的代码
Else
    MessageBox("您输入了错误的密码","您建立的应用程序不能运行!请按确定按钮退出应用
        程序!",Exclamation!,OK!)
    close(w_mima)
    //以上两个语句用来处理用户输入错误的密码后执行的代码
    //您也可以根据自己的实际需要添加不同的代码
End If
```

单击“Save”图标，保存脚本。

四、运行应用程序

当应用程序创建完成以后，接下来要运行该程序。单击“**Run**”图标，便可以运行程序了。运行结果如图 1-1 所示。

实例2 创建窗口和菜单

实例说明

本例将创建一个 MDI 窗口和一个菜单，介绍一些有关窗体和菜单的概念和属性，使您能够认识各种类型的窗体，并且学会如何创建它们。本实例是其他实例的基础。实例中未给出脚本，主要是教会读者如何进行操作，至于实现功能的脚本，会在以后的实例中具体给出。本例最终效果如图 2-1 所示。

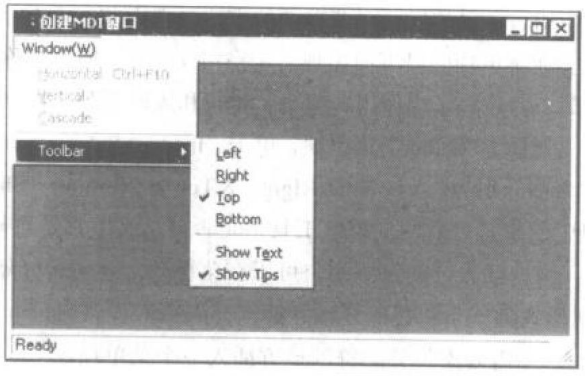

图 2-1 效果图

编程思路

窗口是 Windows 应用程序的基本用户界面元素，窗口也是 PowerBuilder 中的一个重要对象，它是用户与 PowerBuilder 应用程序之间的接口，其他的应用程序组件都是通过窗口联系起来并展现给用户的。

本实例主要通过建立一个 MDI 窗口来介绍不同类型的窗口的概念以及它们的区别。又因为 MDI 窗口必须带有菜单，故本实例还介绍了如何创建菜单。

创作步骤

一、建立 Application

按照实例 1 的方法，创建一个 Application，取名为“sample2”，存放在“C:\PB\sample2.pbl”中。

二、创建菜单

1. 单击“File→New”菜单或者单击工具条上的“New”图标，弹出“New”对话框。

2. 在弹出的对话框中选择“Object”页，选择“Menu”，表明我们要新建一个菜单。

3. 单击“OK”按钮，打开菜单画笔。由于这是一个新建的菜单，所以打开的菜单工作区没有菜单项。

4. 插入菜单项 单击“Insert→Submenu Item”菜单或者从单击鼠标右键弹出的菜单中选择“Insert Submenu Item”，PowerBuilder 将显示一个空白文本框；在空白文本框内输入菜单项名并按“Enter”键。

5. 设置快捷键 每一个菜单项都可以有一个快捷键，使用户在菜单显示时可以通过键盘上的“Alt+Key”键来选择该菜单。快捷键在菜单项上显示为划线的字母，设置方法是在菜单项名前加“&”符号，即符号“&”后面的字母为该菜单的快捷键。

我们要建立一个 Window 菜单项，可在空白文本框内输入“&Window”。则设置该菜单项的快捷键为 W，即通过按下“Alt+W”键可以访问该菜单。

6. 添加下拉菜单 有一些菜单条具有下拉菜单，如我们刚刚建立的 Window 菜单项。其方法是在菜

单条中选中要添加下拉菜单的菜单项，按鼠标右键，在弹出的菜单中单击“Insert Submenu Item”，将会出现一个空白文本框，输入菜单项名并按“Enter”键。

为 Window 菜单项的下拉菜单添加四个菜单项，名称分别为：&Horizontal、&Vertical、&Cascade 和 &Toolbar。添加后三个菜单项的方法是选中前一个菜单项，单击“Insert→Menu Item At End”菜单命令，将会在菜单项的最下面出现一个空白文本框，输入名称。前三个菜单项的作用是当 MDI 窗口中有多个子窗体时，对这些子窗体进行水平、竖直和级联排列；Toolbar 菜单项有一个子菜单，其作用是对工具条进行控制。选择“Toolbar”菜单项，单击“Insert→Submenu Item”菜单，为其建立一个子菜单，共有六个菜单项，其名称分别为：&Left、&Right、&Top、&Bottom、Show T&Ext 和 Show T&Ips。前四个控制工具条在窗口中的位置；后两个控制工具条是否显示文本和是否显示提示信息。

7．**插入分隔符**　用户可以在菜单项之间插入分隔符，以划分不同的功能区。分隔符在菜单条上的表现形式为一条灰色的横线。插入分隔符的步骤为：

在需要添加分隔符的地方插入一个菜单项；

在菜单项的文本框中输入减号（-），并按“Enter”键。

本实例中，我们要在 Cascade 和 Toolbar 之间、Toolbar 子菜单的 Bottom 和 Show Text 之间插入分隔符。方法为：选中“Toolbar”菜单项，单击“Insert→Menu Item”菜单，在出现的菜单项文本框中输入减号（-）；用同样的方法在 Bottom 和 Show Text 之间插入分隔符。

8．设置菜单项属性　菜单对象的属性视图共分两个页面：General（通用）属性和 Toolbar（工具条）属性。

在通用属性页中，用户可以为菜单项指定名称、帮助、快捷键、ShiftToRight 以及诸如 Visible、Enabled 等外观属性。如表 2-1 所示。

表 2-1　General 常用属性列表

属　性	作　用	缺省值
Name	设置菜单项的名称	M 菜单项文本
MicroHelp	设置菜单项的帮助，显示在 MDI 窗口底部	
Visible	决定菜单项是否可视	选中
Enabled	决定菜单项是否可用	选中
Checked	决定是否在菜单项前加一个复选标志	不选中
Default	决定菜单项前的文本是否被加粗	不选中

在 MDI 应用中用户可以为菜单指定工具条按钮和提示信息，也可以指定工具条图标是否显示以及其显示的位置。每一个菜单项都可以对应一个工具条按钮，如无必要，亦可不指定。

工具条属性页面可以指定工具条按钮的提示信息（ToolBarItemText）、工具条按钮的图标文件（ToolBarItemName）、图标按钮是否可视（ToolBarItemVisible）等信息。

三、创建 MDI 窗口

创建窗口有两种方式：普通方式和继承方式。普通方式是不以已存在的窗口为祖先而直接以系统提供的方式创建；继承方式是以一个已经存在的窗口为祖先窗口，在它的基础上生成新的窗口。下面将介绍以普通方式创建 MDI 窗口，其步骤如下所述：

1．在工具条上选择“New”图标，打开“New”对话框。

2．在弹出的对话框中选择“Object”页面，并选中“Window”图标。

3．选择“OK”按钮，打开窗口画笔。

4．设置属性 创建完的窗口并不符合要求，所以我们要对该窗体进行属性控制。在窗体的属性视图中共包含四个页面：General、Scroll、Toolbar 和 Other。

General 页主要设置窗口有关表现风格的属性。各属性的作用如表 2-2 所示。

表 2-2 General 常用属性列表

属 性	作 用	缺省值
Title	设置窗口标题，用于显示在窗口的标题栏上	Untitled
MenuName	指定与窗口关联的菜单名	空
Visible	指定该窗口是否可视	选中
Enabled	指定该窗口是否激活	选中
TitleBar	指定标题条是否显示	选中
ControlMenu	指定是否在标题条中显示菜单	选中
MaxBox	指定该窗口是否在执行过程中可最大化	选中
MinBox	指定该窗口是否在执行过程中可最小化	选中
Resizable	指定窗口执行时大小是否可变	选中
Border	指定窗口是否有边框	选中
WindowType	从下拉列表中选择窗口类型	Main!
WindowState	从下拉列表中选择窗口运行状态	Normal!
Icon	指定窗口最小化时显示的图标文件	空

在“Title”文本框内输入标题“创建 MDI 窗口”；由于我们要创建 MDI 窗口，故在“WindowType”下拉列表中选择“mdihelp!”；在“WindowState”下拉列表中选择“maximized!”，使窗口运行时最大化；指定菜单 m_mdi。“WindowType”下拉列表共有六个选项，分别为：“child!”、“main!”、“mdi!”、“mdihelp!”、“popup!”和“response!”。

child 窗口

子窗口总是与父窗口关联在一起并完全处于父窗体之中，其行为受到父窗体的影响。子窗口无菜单，一般只考虑其父窗体的行为。在 PowerBuilder 新版本中，child 窗口很少用到。

main 窗口

主窗口就是一个普通窗口，可以覆盖其他窗体，也可以被其他窗体所覆盖。它不与任何窗体相关联，可被最大化、最小化及改变大小。在 PowerBuilder 中，它是最常用的，经常把它作为 MDI 窗口的子窗口使用。

mdi 窗口以及 mdihelp 窗口

多文档窗口以及带微帮助的多文档窗口，是可在其中显示其他窗体的主窗口。在该窗口内可打开多个子窗口并任意切换。这两种类型的窗口都必须带菜单，不同之处在于是否在状态条上显示菜单项的微帮助信息。

popup 窗口

弹出式窗口也总是有一个父窗口，但它不像子窗口那样严格地限制在父窗口之中，它可以显示在父窗体的边框之外，并且它总是显示在父窗口的前面。当父窗体最小化时，弹出窗体将被隐藏；当弹出窗体最小化时，将在屏幕的底部显示图标。

response 窗口

响应式窗口用来向用户请求信息。它们总是从另一个窗口（它的父窗口）中打开。一般说来，响应窗口在父窗口的某个事件被触发后打开。它除了使应用程序模态之外，其他方面与弹出式窗口相同。当打开一个应用程序的模态窗口后，就不能访问其他的应用程序窗口，直到关闭该窗口为止。

Scroll 页面是用来设置有关垂直和水平滚动条的。

表 2-3 Scroll 属性列表

属　性	功　能	缺省值
HScrollBar	设置水平滚动条	不选中
VScrollBar	设置垂直滚动条	不选中
UnitsPerLine	设置用户单击垂直滚动条中上箭头或下箭头时，向上或向下滚动的距离（以 PowerBuilder 单位计算）	0
UnitsPerColumn	设置用户单击水平滚动条中左箭头或右箭头时，向左或向右滚动的距离（以 PowerBuilder 单位计算）	0
ColumnsPerPage	单击水平滚动条时窗口滚动的列数（每页 10 列）	0
LinesPerPage	单击垂直滚动条时窗口滚动的行数（每页 10 行）	0

ToolBar 页面是用来设置有关工具条属性的。

表 2-4 Toolbar 属性列表

属　性	功　能	缺省值
ToolbarVisible	指定窗口中的工具条是否可见	选中
ToolbarAlignment	初始化时，指定在 MDI 窗口中显示的位置	Alignattop!
ToolbarX	指定工具条浮动时横坐标值	0
ToolbarY	指定工具条浮动时纵坐标值	0
ToolbarWidth	指定工具条浮动时宽度值	0
ToolbarHeight	指定工具条浮动时高度值	0

Other 页面是用来设置有关窗口显示位置和大小属性的。

5．设置完窗体的各个属性后，取名为“w_mdi”，保存。

四、预览 MDI 窗口

通过单击 Preview 图标 进行窗体的预览。

注释：正如前面所述，本实例从严格意义上讲并不算做一个可以执行的效果，之所以把它单列出来，主要是通过实例教会读者如何制作菜单和窗体，并且如何设置它们的各个属性。这主要为我们以后的实例做准备。本实例不包含脚本，在以后的例子中会给出实现各个功能的脚本。

实例 3 创建多文档应用程序

实例说明

本例将真正创建一个 MDI 应用程序。通过本例，您将学会创建一个完整的 MDI 应用程序，包括菜单和各种窗体的创建、脚本的添加等。本实例不再给出菜单和窗体的具体制作步骤，有关内容请查看前面的实例。本实例 MDI 窗口的三个子窗口上无任何控件，有关使用控件的方法详见下面的实例。本例最终效果如图 3-1 所示。

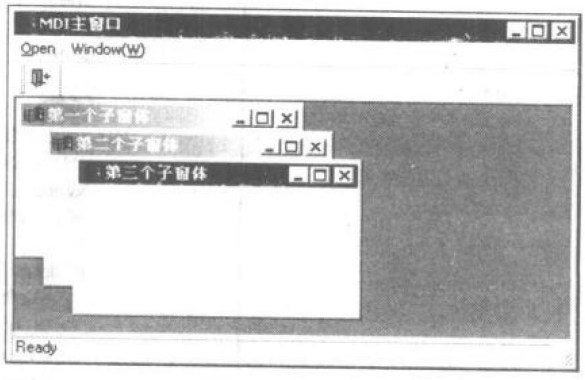

图 3-1 效果图

编程思路

本例创建的 MDI 应用程序在开始时打开一个 MDI 窗口，通过菜单 Open 可以控制三个子窗口的打开；通过菜单 Window 可以控制工具条的位置和是否显示微帮助等信息。

MDI 是 Multiple Document Interface 的英文缩写，它的中文含义是“多文档界面”，MDI 代表着一种应用窗口的风格。在 MDI 应用中，有一个称为 MDI 框架的主窗口，在这个主窗口中可以打开任意个子窗口，并可在打开的窗口之间切换。

MDI 应用程序是一种很流行的应用风格，Windows 上的许多软件都采用这种风格。我们使用的 PowerBuilder 7.0 的开发环境主窗口就是一个 MDI 窗口。若想建立一个能在其中打开多个窗口，并且易于在打开窗口之间进行切换的应用程序，最好采用 MDI 界面风格。建立一个 MDI 应用，首先应该定义一个多文档窗口或者带微帮助的多文档窗口，然后再根据需要创建一些子窗口。

创作步骤

一、建立 Application

创建一个名为“sample3”的 Application，存放在“C:\PB\sample3.pbl”中。

二、创建菜单

由于 MDI 窗口必须要求带有菜单，所以我们首先要创建一个名为 m_mdi 的菜单，其菜单结构如图 3-2 所示。

三、创建窗口

本实例共用到四个窗体：一个 MDI 主窗体、三个 main 类型的子窗体。其窗口的各个属性设置如表 3-1 所示。

这四个窗体的其他属性均保持缺省。

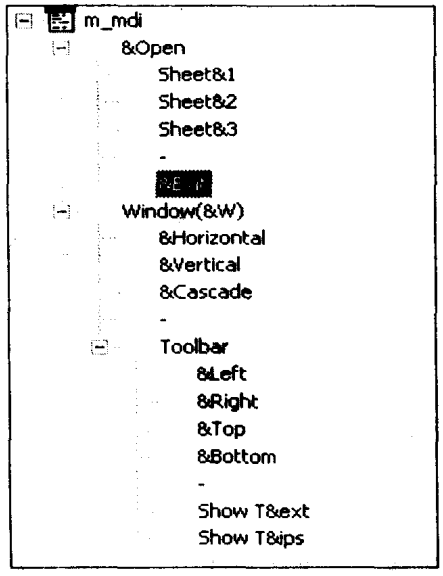

图 3-2　菜单结构

表 3-1　窗口设置

窗　口	名　称	Title	MenuName	WindowType	WindowState
主窗口	W_mdi	MDI 主窗口	M_mdi	Mdihelp!	Maximized!
子窗口	W_sheet1	第一个子窗体		Main!	Normal!
子窗口	W_sheet2	第二个子窗体		Main!	Normal!
子窗口	W_sheet3	第三个子窗体		Main!	Normal!

四、添加脚本

1．给 Application 添加脚本　在 PowerBuilder 中，应用是一切开发编程的起点。一个应用就是一个对象的集合体，此集合体中包含了窗口、数据窗口、菜单、用户对象、函数、结构、应用对象或其他 PowerBuilder 支持的对象。通过编程，可使集合中的对象（例如，窗口和菜单）完成某些特定的功能，而这些功能的集成就是一个应用程序。开始某个应用程序是通过集合中的应用对象来完成整个应用程序的启动。因此，应用对象是一个应用程序的入口。

由于本实例并未涉及到数据库，故不需对数据库管理系统进行初始化；在应用对象中，我们只需打开 MDI 窗口即可。所以，在 Application 的 Open 事件中添加如下脚本：

```
open(w_mdi)
//打开 MDI 主窗口
```

2．给菜单项添加脚本　本实例中的菜单用来实现打开子窗口、对子窗口进行排列、对工具条进行控制的功能。

在 Open→Sheet1 菜单的 Click 事件中添加如下脚本：

```
//打开子窗口 Sheet1
opensheet(w_sheet1,"w_sheet1",w_mdi,-2,Original!)
```

Open→Sheet2 菜单和 Open→Sheet3 菜单的 Click 事件中的脚本与 sheet1 脚本相似，详见本书所附光盘。

Open→Exit 菜单是用来关闭应用程序的，本实例中只需将 MDI 主窗口关闭即可。所以，在其 Click 事件中添加如下脚本：

```
//关闭 MDI 主窗口
close(w_mdi)
```

Window 菜单是用来控制工具条和子窗口的。在 Window→Horizontal 菜单的 Click 事件中添加如下脚本：

```
//将子窗口水平排列
w_mdi.arrangesheets(TileHorizontal!)
```

在 Window→Vertical 菜单和 Window→Cascade 菜单的 Click 事件中的脚本与 Window→Horizontal 菜单的 Click 事件中的脚本相似，详见光盘。

Window→Toolbar 菜单是用来控制工具条的。在 Window→Toolbar→Left 菜单的 Click 事件中添加脚本：

```
//工具条在窗口的左部
w_mdi.toolbaralignment=alignatleft!
```

在 Window→Toolbar→Right、Window→Toolbar→Top、Window→Toolbar→Bottom 菜单的 Click 事件中的脚本与 Window→Toolbar→Left 脚本相似，详见光盘。

在 Window→Toolbar→Show Text 菜单的 Click 事件中添加如下脚本：

```
application app
//定义 Application 类型的变量用来存储应用对象
app=getapplication()
//取得应用程序的应用对象
this.checked=not this.checked
//变换复选标志
app.toolbartext=this.checked
//设置显示文本
```

在 Window→Toolbar→Show Tips 菜单的 Click 事件中添加如下脚本：

```
application app
app=getapplication()
this.checked=not this.checked
app.toolbartips=this.checked
```

3．添加完脚本之后，可以通过单击 Run 图标，运行该应用程序。其结果如图 3-1 所示。

到此，整个实例制作完毕。

实例 4　使用常用控件

实例说明

本实例教会读者如何使用常用控件的应用程序。由于 PowerBuilder 中为方便用户提供了大量的控件，在这里我们不可能对每一个控件都进行介绍，只能选择一些用户开发应用程序时经常用到的。本实例并无实际意义，只是展示一下如何使用这些常用控件。其他控件的使用方法，将在以后用到时进行具体的介绍。本例最终效果如图 4-1 所示。

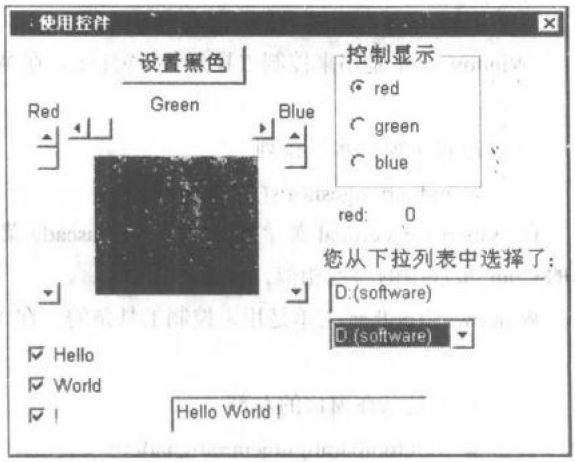

图 4-1　效果图

编程思路

用户可以使用颜色和工具条之类的东西来美化窗口的外观，但是控件才真正具有某些窗口的功能。在运行应用程序时，用户主要通过放置在窗口中的控件与应用程序进行交互。

本实例主要演示了如何使用滚动条、按钮、静态文本框、复选框、圆形按钮等控件。用户可以通过滚动条改变静态文本框的背景颜色；可以通过圆形按钮控制显示哪一个色域值；当您改变下拉列表时，可以在文本框中显示您的选择；可以通过复选框控制显示的内容。

创作步骤

一、建立 Application

创建一个 Application，取名为“sample4”，存放在“C:\PB\sample4.pbl”中。

二、创建窗口

本实例只需要一个窗口作为各个控件的容器，故我们只创建一个名为“w_main”的类型为“main!”的窗口。其“Title”属性设置为“使用控件”；取消“MaxBox”、“MinBox”和“Resizable”前面的复选标志。

三、添加控件

根据控件功能的不同，将 PowerBuilder 提供的控件分成四类，如表 4-1 所示。

除了上述系统提供的控件外，还可以定义用户对象。如果经常使用某个或者某些具有特定属性的控件，

如一组相关的单选按钮，那么就可以创建一个用户对象，它就是一个或一组事先定义好的可以放置在窗口中的控件。

本实例主要使用了 CommandButton、SatatictText、DropDownListBox、SinglcLineEdit、HScrollBar、VScrollBar、CheckBox、RadioBox、GroupBox。

表 4-1 控件分类

功 能	控 件 名
激活动作	CommandButton、PictureButton、PictureHyperLink、StaticHyperLink、Tab、User Object
显示数据	ListBox、PictureListBox、DropDownListBox、DropDownPictureListBox、DataWindow(Control)、StaticText、ListView、RichTextEdit、Graph、Picture、ProgressBar、ScrollBar、SingleLineEdit、MultiLineEdit、EditMask、Tab、User Object、OLE
指示选择	RadioButton、CheckBox、TrackBar
修饰	Line、Rectangle、RoundRectangle、Oval

向窗口上添加控件有两种方法：

单击“Insert→Control”菜单，从其子菜单中选择所要添加的控件。

选择控件图标 OK ▾，从其下拉列表中选择所要添加的控件。

选择好要添加的控件后，只需在窗口的空白处单击，便可将该控件添加进来。用上面的方法向窗体 w_main 上添加控件，并设置属性。

1. CommandButton 本例只需要一个 CommandButton，起作用时设置“StaticText”的背景色为“黑色”。该控件是使用最频繁的控件之一，其属性设置较为简单，这里不做叙述。本例中，将其“Text”属性设为“设置黑色”。

2. StaticText StaticText 也是使用较多的一种控件，它的作用是给用户以提示信息。本例中，我们共用到七个该类型控件。我们只需对其“Text”属性进行设置，分别为：“Red”、“Green”、“Blue”、“red”（name 为 st_control）、“0”（name 为 st_value）、“您从下拉列表中选择了：”，还有一个是用来显示颜色的，其“Text”属性设为空，取名为“st_color”。

3. VscrollBar 本例共需要两个 VScrollBar，分别用来表示背景颜色的 red 值和 blue 值。VScrollBar 有三个比较重要的属性：

Position 指定滚动条的初始位置。在运行时，它应当表示滚动条上的滑块位置，处于 Min Position 和 Max Position 之间。

Min Position 指定滚动条上的滑块位置的最小值。

Max Position 指定滚动条上的滑块位置的最大值。

为这两个 VScrollBar 控件取名为“vsb_red”和“vsb_blue”，其“Min Position”、“Max Position”和“Position”属性均设置为 0、255 和 0。

4. HscrollBar 我们用该控件来表示背景颜色的 green 值。其属性大部分与“VScrollBar”相似。我们为其取名为“hsb_green”，其“Min Position”、“Max Position”和“Position”属性分别设置为 0、255 和 0。

5. GroupBox GroupBox 是一个简单的控件。它本身不做任何事情，只是把其他控件按功能分成组，使用户界面更加清晰，也更易于理解。

该控件的“Text”属性定义显示在分组框上面的标题，它是左对齐的；还可以通过选择 BorderStyle 下拉列表的选项为该控件指定边界样式。本例中，我们为其“Text”属性设为“控制显示”。

6．RadioButton　RadioButton 是一种圆形按钮，用来表示彼此排斥的选项。该按钮总是用在 GroupBox 中，且每个 GroupBox 中只能有一个 RadioButton 被选中。当用户单击某一 RadioButton 时，将选中该按钮，并使同组其他按钮不被选中。用户可以设置其 Checked 属性来设置 RadioButton 的初始状态是否被选中。

这里，我们共需要三个该类型按钮，其“name”属性分别设为：“rb_red”、“rb_green”和“rb_blue”；其“Text”属性分别设为：“red”、“green”和“blue”；其中选中“rb_red”的“Checked”属性的复选标志。

7．SingleLineEdit　SingleLineEdit 是一个用来输入单行文本的方框。通常情况下，用于数据的输入和输出。用户可以指定单行编辑器是否有边框（Border 属性），是否在需要时显示滚动条（AutoHScroll 属性），是否将文本显示为星号（Password 属性）。SingleLineEdit 控件与 StaticText 控件都可用来显示数据。不同之处在于：在执行过程中（不通过脚本），StaticText 内容不可修改；而在 SingleLineEdit 中，用户可以输入或者删除数据。

这里需要两个 SingleLineEdit 控件。其中一个用来显示下拉列表所做的选择，其“name”设为“sle_choice”，“Text”属性设为空；另一个用来显示选中的 CheckBox 的标题，其“name”属性设为“sle_display”，“Text”属性设为“Hello World！”。

8．CheckBox　CheckBox 是一个方框，用来表示彼此独立的选项。通常情况下，CheckBox 有两种状态：选中和未选中。CheckBox 有许多属性，大部分与 RadioButton 相同。复选框除具有两种状态外，还可能有第三种状态，即介于选中和未选中之间的中间状态，用灰框显示。这可以通过 General 页的 ThreeState 的复选框设置。

这里需要三个 CheckBox 控件，用来表示是否显示 CheckBox 上的 Text。其“name”属性分别设置为：“cbx_hello”、“cbx_world”和“cbx_exclaim”；其“Text”属性分别设为：“Hello”、“World”、“!”。

9．DropDownListBox　DropDownListBox 具有 SingleLineEdit 和 ListBox 两者的特点。通常情况下，DropDownListBox 只显示类似 SingleLineEdit 的方框，而将可选项放置于隐藏的列表框中，当用户选择列表中的某一项时，该项将显示在方框中，然后隐藏列表框。

这里需要一个 DropDownListBox 控件，用来提供用户选择。该控件属性视图中有一个 Item 页面，用来输入隐藏在列表框中的项名。本例中，我们输入三项：“C:(system)”、“D:（software）”、“E:(data)”。

按照上面的方法设置完控件后，需要对控件进行合理的布局。读者可参考图 4-1 进行布局。

四、添加脚本

为了使控件能够运行起来，需要添加脚本。

1．为 Application 添加脚本　在 Application 的“Open”事件中添加打开窗口的脚本。详见本书实例 5。

2．给 CommandButton 控件添加脚本　该命令按钮用来设置“st_color”的背景色为“黑色”，并相应设置滚动条的 Position 值。其 Click 事件的脚本如下：

```
vsb_red.position=0
vsb_blue.position=0
hsb_green.position=0
st_color.Backcolor=RGB(0,0,0)
st_value.text="0"
```

3．给滚动条添加脚本　当用户拖动滚动条时，会相应改变 st_color 背景色的 RGB 值。故在 vsb_blue 的 moved 事件中添加如下脚本：

```
st_color.backcolor=RGB(vsb_red.position,hsb_green.position,vsb_blue.position)
```

```
If rb_blue.checked=true Then
    st_value.text=string(this.position)
End If
```

当用户用鼠标拖动滑块时，该部分代码被执行。其他两个滚动条的脚本详见光盘。

4. 给 RadioButton 添加脚本 该控件是控制在下面的 StaticText 中显示哪一个（R、G、B）值。在 rb_red 的 Clicked 事件中添加如下脚本：

```
If this.checked=true Then
    st_control.text="red:"
    st_value.text=string(vsb_red.position)
End If
```

其两个 RadioButton 的脚本详见光盘。

5. 给 DropDownListBox 添加脚本 该控件提供用户进行选择，当用户从下拉列表中选择一项时，将会在其上的 SingleLineEdit 控件中显示用户的选择。故在其 SelectionChanged 事件中添加如下脚本：

```
sle_choice.text=ddlb_choice.text
```

每当用户改变下拉列表的选项时，触发该事件。

6. 给 CheckBox 添加脚本 该控件用来控制显示在 sle_display 中的内容。当用户选中一个 CheckBox 时，该控件的 Text 将会被显示在 SingleLineEdit 方框中。故在这三个 CheckBox 控件的 Clicked 事件中均添加如下脚本：

```
sle_display.text=""
If cbx_hello.checked=true Then
   sle_display.text=cbx_hello.text
End If
If cbx_world.checked=true Then
   sle_display.text=sle_display.text+" "+cbx_world.text
End If
If cbx_exclaim.checked=true Then
   sle_display.text=sle_display.text+" "+cbx_exclaim.text
End If
```

7. 添加完脚本以后，可以运行该应用程序。其结果如图 4-1 所示。

实例 5　使用新增控件

实例说明

前一个实例介绍了常用控件的使用方法，本实例将介绍如何使用 PowerBuilder 7.0 中新增加的三类控件。本实例通过在应用程序中使用这三类控件，来达到使用户了解它们用法的目的。在实际应用过程中，您可以根据自己的需要，参考本实例具体使用这些控件。本例最终效果如图 5-1 所示。

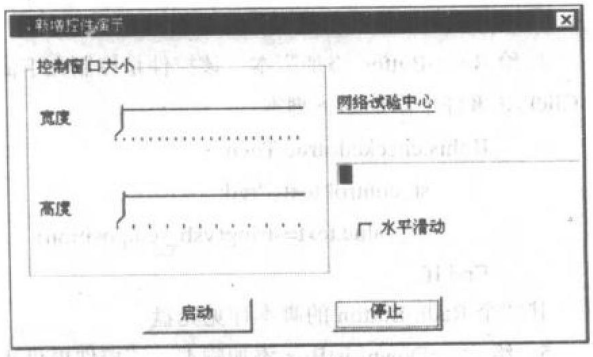

图 5-1　效果图

编程思路

PowerBuilder 7.0 中新增了三类控件：超级链接控件、进度条控件、刻度条控件。本实例将对这三种控件的属性和用法进行讲解，并且给出一个演示的例子，以使读者能够了解这些新控件的用法。

1．超级链接控件　超级链接控件有两种：静态文本超级链接（StaticHyperLink）控件和图片超级链接（PictureHyperLink）控件，分别用于在窗口上放置文本和图片。它们有一个共同的属性是 URL（Uniform Resource Locator，即统一资源定位器），用于指定要跳转的 Internet 地址，例如 net.ncut.edu.cn。这类控件没有单击事件，当用户将鼠标指针移到控件上时，指针自动变成一只小手，单击鼠标左键即可打开您机器上的浏览器，并且查找 URL 指定的位置。该控件为用户开发 Internet 程序提供了很大的方便。

2．进度条控件　进度条控件分为两种：水平进度条（HProgressBar）控件和垂直进度条（VProgressBar）控件。它们通常用于指示长时间操作的进度，例如安装程序的拷贝过程等等。

3．刻度条控件　刻度条控件也分为两种：水平刻度条（HTrackBar）控件和垂直刻度条（VTrackBar）控件。和滚动条类似，刻度条也是一类滚动控制，不同的是在刻度条上单击鼠标时，它的增量是不连续的，并且不需要编程来移动标尺（Slider）的位置。刻度条控件通常让用户选择一些离散值，例如窗口的大小、颜色等。

下面将举例说明上述三类控件的用法．本实例要完成以下功能：

通过窗口高度和窗口宽度来调整窗口的大小；

通过静态文本超级链接启动本级的浏览器访问 net.ncut.edu.cn；

演示进度条的推进。

创作步骤

一、建立 Application

创建一个名为“sample5”的 Application，存放在“C:\PB\sample5.pbl”中。

二、创建加密窗口

创建一个名为“w_main”的窗口，其“Title”属性设置为“新增控件演示”；取消“MaxBox”、“MinBox”、“Resizable”前面的复选标志；设置窗体的“Width”为“2100”、“Height”为“1100”。

三、添加控件

该窗口控件如表 5-1 所示。

表 5-1 控件列表

控件类型	控 件 名	用 途
水平刻度条	htb_width	控制窗口的宽度
	htb_height	控制窗口的高度
水平进度条	hpb_demo	演示进度条
复选框	cbx_smooth	设置水平进度条的 SmoothScroll 属性
按钮	cb_start	开始推进水平进度条
	cb_stop	停止推进水平进度条
静态文本超级链接	shl_home	演示静态文本超链接

各种控件的其他属性设置为：“htb_width”的“MinPosition”为“11”、“MaxPosition”为“22”、“Position”为“11”、“TickFrequency”为“1”；“htb_height”的“MinPosition”为“21”、“MaxPosition”为“42”、“Position”为“21”、“TickFrequency”为“1”。

除了上述控件外，本实例还用到了两个 StaticText 控件和一个 GroupBox 控件。

四、添加脚本

除了在 Application 的 Open 事件中加入打开主窗口的脚本外（详见光盘），我们还需要给控件添加实现功能的脚本。

1．本实例在处理进度条的推进时需要为 SmoothScroll 设置一个变量，由于该变量要在不同事件的脚本中引用，故其为全局的。在 PowerBuilder 中设置全局变量的方法为在任意事件的脚本编辑器中单击鼠标右键，在弹出的菜单中单击“Go to→Global Variables”菜单命令。设置变量如下：

```
boolean ib_stop
```

2．当用户单击水平刻度条时，要改变窗口的宽度和高度。在 htb_width 的 Moved 事件中添加如下脚本：

```
//改变窗口的宽度和高度
w_main.width=this.position*100
w_main.height=htb_height.position*100
```

在 htb_height 的 Moved 事件中添加如下脚本：

```
w_main.width=htb_width.position*100
w_main.height=this.position*100
```

当用户用鼠标单击水平刻度条时触发该事件。

3．当用户选择 cbx_smooth 时，决定水平进度条是否为 SmoothScroll，故在其 Clicked 事件中添加如下脚本：

```
hpb_demo.SmoothScroll=cbx_smooth.checked
```