报表系统设计中 VB 与 EXCEL 的集成应用

张帆 郑志波 胡念苏 (武汉水利电力大学 430072)

摘要:本文介绍了利用 Visual Basic 的 OLE 功能,将 EXCEL 链入 OLE 容器内制作火电厂运行经济报表的方法。同时详细讨论了在设计中如何灵活运用 Windows 编程工具解决工程应用实际中出现的各种问题。

关键词:OLE 容器 EXCEL 报表

一、概述

在实际工程应用中,经常会碰到报表打印的要求,对于火电厂的运行人员而言,报表的重要性尤为突出。笔者根据自己实际开发的景德镇电厂运行经济性日报表系统,向大家介绍一种用 Visual Basic 设计美观实用的报表打印系统的方法。

若报表篇幅较小,且只有文本表格的打印要求,可选用 VB 自带的 Grid 控件,制作方便且易于修改(可用一文本框作编辑框)。但实际工程应用中,报表篇幅通常较大,无法一屏显示,而且一般要求报表系统提供饼状图、柱状图或趋势图显示,用上述方法根本无能为力。考虑到 EXCEL 具有强大的表格制作、数据处理、相关图形显示及打印功能,我们可在 VB 中用 OLE(Object Linked and Embedded)功能将 EXCEL 工作表连入 OLE 控件中,由代码操纵 EXCEL 完成报表的显示及打印。

二、问题的提出与解决

单纯的报表制作并不复杂,但在工程应用中,它必须满足许多特殊的要求。针对我们所设计的火电厂运行经济性报表打印系统,首先要求它能在事故发生时反映问题所在,因此一般须查看相关时间段内(根据实际情况取为五天)的数据,即要保留前五天的书数据。每天转班时刻(电厂一般为凌晨二时)当天数据更新,第一天的数据转存为第二天,第二天数据转存为第三天……依次类推,第五天的数据则"抛弃"。其次,报表数据来源于热力计算结果,它以文本方式存于名为"RbDate.txt"的文件中,要将它按所需格式放入 OLE 容器中,需经过一定的转换。第三,由于报表数据采用文本方式存储,而五天共有 1500 多个数据,要将如此庞大的数据传入 OLE 容器中的 EXCEL 工作表,若用简单的循环控制赋值显然速度太慢(因对 EXCEL 单元格操纵速度很慢),如何实现数据的快速存取也是有待解决的问题之一。

1. 建立链接

建立 OLE 控件与 EXCEL 表格之间的链接一般有两种方法:

(1)在设计阶段插入 OLE 对象,步骤如下:

①选择需要链接的数据(为 RbDate.txt 文件中的一部分);

②在"Edit"菜单中选择"Copy"命令,将所需数据拷贝到剪贴板;

③在应用程序中,用鼠标右键单击 OLE 控件,然后选择"Paste Special"命令;

④选择"Paste Link"选项创建链接对象;

⑤单击"OK"按钮,创建链接对象。

(2)在运行时创建 OLE 对象

要在运行时创建链接对象,需在代码中使用 OLE 控件的方法与属性。例如:

```
Olesheet.CreateLink(App.Path & "/Data.xls!Sheet1")
```

2. 数据转存

由于一共有五个工作表,用"第一天"表示当天的工作表,"第二天"表示前一天的表……,"第五天"即表示四天前的工作表。每到凌晨两点,将原来各天的工作表依次转存为头一天的工作表(显然,"第五天"工作表将自然"抛弃"),然后把更新的数据放入"第一天"的工作表中。这样就仍然保留了从新的一天算起的五天内的工作表。代码实现为:

```
Option base 1
Dim StoreFlag(5,5) As Integer
Dim I As Integer, J As Integer
Dim Temp As Integer
For I=1 To 5
StoreFlag(I,1)=I
```

```
Next I
For J=2 to 5
For I=1 To 5
Temp=StoreFlag(I,J-1)-1
If Temp=0 Then Temp=5
StoreFlag(I,J)=Temp
Next I
Next J
```

每读入一次新数据 J 自动加 1(超 5 则赋值 1),新数据存入 StoreFlag 数组第 1 行、第 J 列所指示的表单中。例如 J 当前值等于 3,StoreFlag(1,3)等于 4,则新数据存入 Sheet4 中。利用 StoreFlag 数组,就能根据当前 J 的值确定 Sheet(I) (I=1～5)表示第几天的数据了。

这样避免频繁地在表单间交换数据,从而加快程序运行速度。

3.格式转换

在数据文件"RbDate.txt"中,五天的数据按 12 列(第一列为序号)、140 行放置。要将其分别读入五个工作表中,且由文本格式转换为所需的 EXCEL 工作表格式。代码实现为:

```
'载入文本文件,并将其作为包含单个工作表的工作簿进行分列处理
xlBook.Application.Workbooks.OpenText App.Path & "RbDate.txt",xlWindows
With xlSheet.Range("a1:l140") '选定数据区域
```

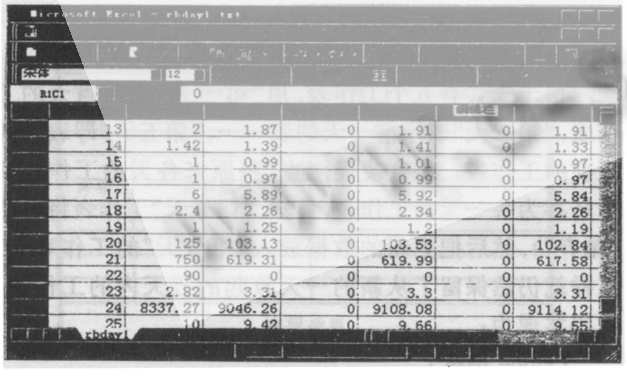

图 1 EXCEL 数据列

```
.Borders.LineStyle=1 '设定边框
.Cells.Font.Size=10 '设定字体大小
.Cells.HorizontalAlignment=3 '使数据水平居中对齐
.Numberformat="0.00_" '设定单元格格式
End With
```

其中 xlBook 为 EXCEL 中存放五张工作表的工作簿对象,xlSheet 为引入 RbDate.txt 所得的工作表对象。上述代码打开 RbDate.txt,并对相应工作表设定所需求的格式(见图 1)。

4.数据读取的加速

通常可以在 VB 中采用循环语句来对 EXCEL 表格中的各个单元赋值,代码如下:

```
Open App.Path & "RbDate.txt" For Random As #1 Len=Len(MyRc)
For I=1 To 5
For Row=1 To 28
For Col=1 To 12
'从文本文件取数据
Get #1,(Row-1)*(Col-1),MyRecord
'对每个单元格赋值
xlSheet(I).Cells(Row,Col)=MyRecord.Data
Next Col
Next Row
Next I
Close #1
```

运行该程序,屏幕上一度出现类似于"死机"的状态(持续了四分钟左右),尽管结果最终无误,但显然不符合工程实际的要求。单步跟踪运行过程,发现在 EXCEL 中,给单元格的赋值速度太慢,几乎可看到其由行表列的转换过程,五天的 1500 多个数据耗时四分钟才赋值完毕。在程序运行的漫长等待期间,很可能导致用户不必要的误操作,这个问题必须予以解决。

由于数据文本在引用时已按 EXCEL 格式分好行和列(见 2.3 所述),因此只需将其整体"贴"入一个已按要求制作好边框和项目栏的空表单中,其间充分利用了剪贴板的强大功能。具体操作如下:

```
For I=1 To 5
'拷贝原数据表所需的行和列
xlSheet.Range("b" & CStr(2+28*(I-1)) & ":l" & CStr(27+28*(I-1))).Copy
'复制到完整的工作表中
xlBook.WorkSheets("Day/&Cstr(I)).Range("b7:l33").PasteSpecial
Next I
```

修改完毕，再次运行，1秒不到即完成全过程！

5. **修改报表**

报表中有部分数据要求由运行人员手工输入，而其相关量也需随之变动，因此，报表部分单元格要求录入相关公式，用代码实现为：

xlSheet(I).Cells(R,C).Formula="=表达式"

由于我们采用了链接对象，一个数据源链接到多个应用程序中，因此当一处的数据改变时，链接程序中所看到的数据内容也将发生改变，多个应用程序之间保持数据的一致性。公式既可于设计阶段录入EXCEL源表中，也可如上由VB代码控制录入，简洁方便。

6. **报表打印**

表格制作完毕之后，利用EXCEL工作表的PrintOut方法进行打印，打印页面设置用CommonDialog控件(ActiveX控件，在Visual Basic5集成环境"工程"项"部件"子菜单中添加)也可简单实现。介绍这方面的文章很多，在此不再赘述。

三、结论

综上所述，可以看到EXCEL本身就有相当强大的表格制作、数据处理、饼状图、直方图、趋势图等特殊图形的直观显示、表单打印等功能，用Visual Basic代码可直接操纵两者的集成应用，大大简化了报表打印系统的制作过程。

参考文献

[1] 东箭工作室编，Visual Basic 5.0中文版程序设计，清华大学出版社，1997.12

[2] Reed Jacobson. Microsoft Excel97 Visual Basic. 人民邮电出版社，1998.1

(来稿时间：1998年12月)