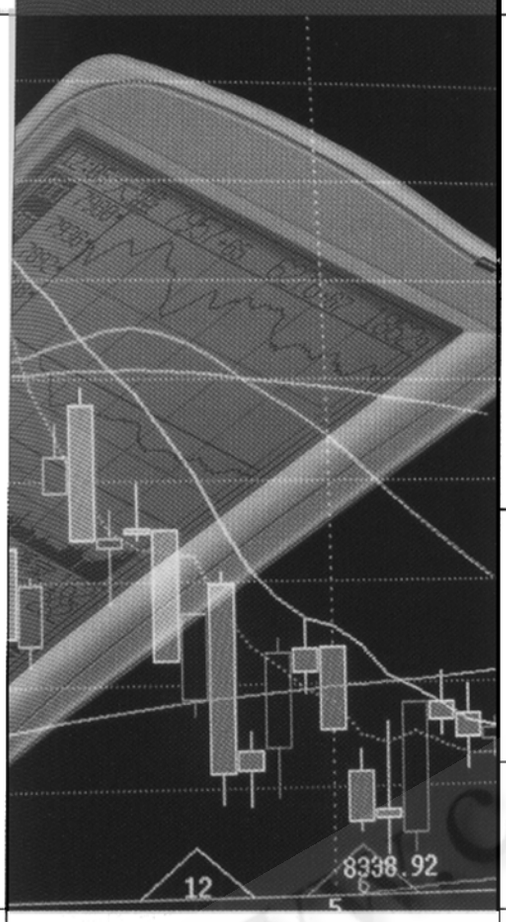

TeeChart 控件的结构分析与应用

The Structure Analyse and Application about TeeChart Component

摘　要：TeeChart 控件组是 Delphi 和 C++Builder 中的一组主要用于生成各种复杂的图表的第 3 方控件。本文从面向对象的角度出发，借助 Rose 工具，力求对其在结构上进行较深层的分析，使之能获得更加广泛和灵活的应用。

关键词：TeeChart　面向对象　分析　应用

李院春　（西安 长安大学网络中心　710064）

郑向宏（西安市国税局信息中心　710068 ）

1 TeeChart 简介

TeeChart Pro ActiveX是西班牙Steema SL公司开发的图表类控件，可以生成柱状图、折线图、饼图等形式的图表。TeeChart控件组包括3个主要控件：TChart、TDBChart和TQRChart。TChart是基本的图表控件，TDBChart在TCHart的基础上，增加了对数据库的支持功能，TQRChart主要为使用QuickReport而开发的。三者的关系如图1：

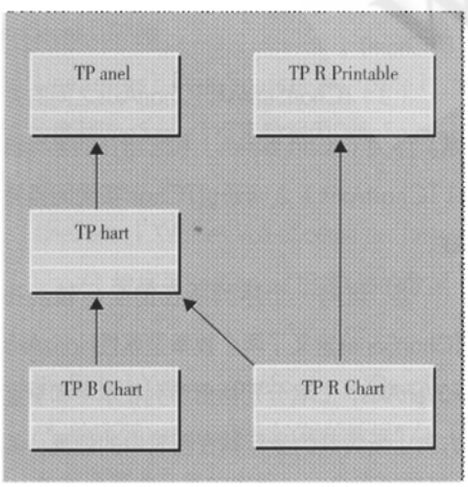

图 1　TeeChart 控件对象

TDBCHart 由TChart继承而来，后者的父类是TPanel；TQRChart是一个接口型（interface）的控件，它利用属性Chart关联一个TChart或者TDBChart。TQRChart将TChart（或者TDBChart）与QuickReports对象连接起来，使所连接的Charts能够包含在报表中。

三者之中，TChart、TDBChart的使用最频繁，因此，下面在主要分析TChart结构的基础上，进一步说明TDBChart的结构，对于TQRChart的结构，限于篇幅，本文不作介绍。

2 TChart 的组成结构

TChart中共定义了325个属性、125个方法以及28个事件，这使得TChart具有非常强大的功能。其组成如图2示：

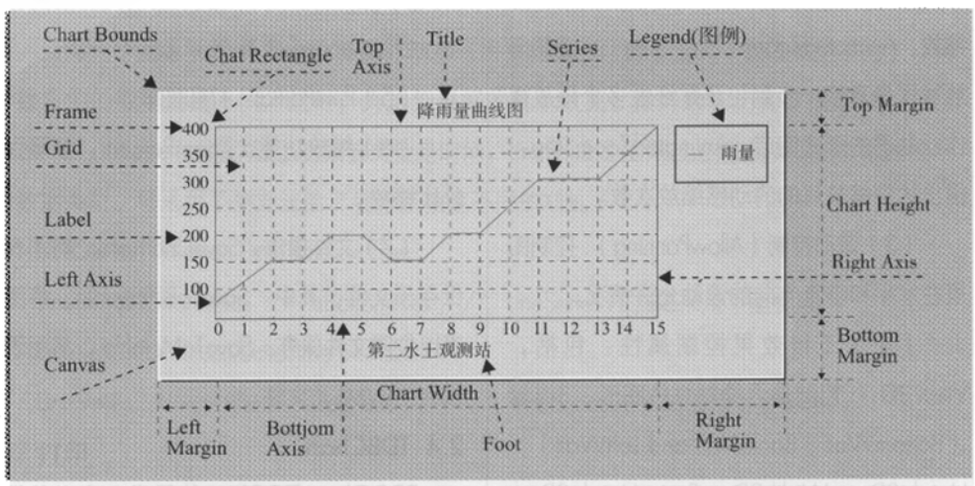

图 2　TChart 组成结构图

2.1 组成元素

（1）标题（Header）：图表的题头(Titles)。

（2）画布（Canvas）：Canvas可以让设计者在Chart控件上绘制自己的图形。有TextOut、LineTo、Arc等各种画图的方法可以调用。

（3）坐标轴（Axis）：总共上、下、左、右、纵深五个坐标轴，即LeftAxis, RightAxis, TopAxis, BottomAxis 和 DepthAxis.（纵深坐标轴DepthAxis用于3维图表）。在缺省的情况下，坐标轴可以自动地根据不同的数据设置好标度范围和间隔。

（4）序列（Series）：图表所绘制的实际内容，在一个图表中可以有一个或多个序列，每个序列可以有不同的显示类型，如折线图、直方图、饼图等。图表中用属性Series数组列出它包含的所有序列。

（5）图例（legend）：图例是图表中的一个长方形区域，用来显示标注。

（6）脚注（Foot）：位于图表底部的一段说明性文字。

2.2 属性

（1）背景属性：由于TChart是Tpanel的子类，所以可以如同对一般Panel那样，控制背景的颜色（BackColor），背景画面（BackImage）等。

（2）Zoom缩放控制：包括，AllowZoom：指出是否运行随鼠标右键拖动而缩放。AnimatedZoom（动感缩放）：控制缩放过程是直接一步到位还是经过多个缩放操作以动感的形式完成。AnimatedZoomStep：指出达到缩放目的的动感缩放次数。

（3）滚动控制（AllowPanning）：控制图表在水平和垂直方向的滚动允许情况。

（4）三维效果控制属性：包括，View3D：控制是否绘制立体图形；3维墙（BottomWall，BackWall and LeftWall）；Height3D，Width3D，SeriesHeight3D，SeriesWidth3D；Chart3Dpercent：指出图表深度与图表大小之间的比例关系；View3Doptions：指出旋转、缩放和滚动时的3维特性。

（5）图表边缘（ChartBounds）：是一个表示图表在屏幕上绘制的位置与大小的矩形区域（用Left, Top, Width，Height表示）。

（6）图表区（ChartRect）：图表中上下左右4个坐标轴所构成的绘制series的矩形区域。

（7）页边（margin）：包括LeftMargin, RightMargin, TopMargin, BottomMargin，是图表区中左右上下4个方向所保留的空白区域。

（8）分页属性：图表可分页进行显示。其中，MaxPointsPerPage指出每页所容纳的最大点数。ScaleLastPage：指出最后一页的显示方式。

（9）缓存式显示（BufferedDisplay）：使图表在显示之前先显示在内部的缓存"canvas"中，以加快显示速度，避免闪烁。

（10）打印控制属性：包括页边打印（PrintMargins），打印分辨率(PrintResolution)。

2.3 方法

（1）AddSeries：在图表中增加series。

（2）RemoveSeries：从图表中删除series。

（3）CheckDataSource：刷新所有series所关联的数据源的值，在图表中反映数据源最新的变化情况。

(4) GetCursorPos: 获取当前光标的坐标（以Chart bounds为基准）。

（5）PrintPartial：打印图表。

（6）缩放功能：ZoomPercent，依照百分比缩放。

（7）文件操作：SaveChartToFile, 将图表保存于文件；其中：SaveToBitmapFile，将图表以位图文件保存；SaveToMetafile，将图表保存为WMF元文件。

2.4 TDBChart

TDBChart是TChart的子类，它在父类TChart的基础上，增加了读取数据库记录的功能。TDBChart通过访问它所包含的series的DataSource属性获取数据库数据。这些DataSource可以是TTable，TQuery，TClientDataset 以及其它可用的DataSet 控件。TDBChart能自动从DataSource中读取记录。可以在series的OnBeforeAdd 事件中设定对记录的过滤条件，以限制对一些记录的显示。TChart 和TDBChart的最大不同是后者需要数据库引擎的支持，而前者不需要。

TDBChart设置下列属性控制对数据库记录的读取：

（1）AutoRefresh：控制是否自动读取数据库记录；

（2）RefreshInterval：定义刷新Datasets的间隙（以妙为单位），0值表示不刷新。

3 TChart的OO分析

3.1 TChart，series，axis之间的关系

（1）TChart与其它对象的关联。在TChart中定义属性Series数组关联它所包含的序列（TChartSeries）；定义属性LeftAxis, RightAxis, TopAxis, BottomAxis 和 DepthAxis分别关联左右上下和深度5个坐标轴；属性Legend关联图例（TChartLedgend）;属性title和foot关联标题和脚注（TchartTitle）；属性Canvas关联画布（TCanvas3D）。

（2）其他对象与TChart的关联。在TchartSeries、TChartAxis、TChartLedgend中分别定义属性ParentChart指向他们所在的图表（TChart）。

（3）TChartSeries和TChartAxis之间的关系。序列（TChartSeries）和它所在的坐标轴（TChartAxis）之间通过TChart实现间接关联。

① 访问一个序列所在的坐标轴。在TChartSeries定义了两个枚举型属性HorizAxis和VertAxis，二者分别指出序列在水平和垂直方向上所关联的坐标轴在图表中的位置。其中，HorizAxis指出水平（X）坐标轴是上部坐标还是下部坐标(取值：aTopAxis，

aBottomAxis)；VertAxis指出垂直（Y）坐标轴是左部坐标还是右部坐标 (取值：aLeftAxis, aRightAxis)。所以，首先根据TChartSeries的属性ParentChart得到本series所在的图表，然后依据HorizAxis和VertAxis的取值就可从TChart的LeftAxis, RightAxis, TopAxis, BottomAxis中定位出水平和垂直坐标。至于深度坐标，就是TChart的DepthAxis。

② 问一个坐标轴所关联的序列 。在TChart中定义了一个方法GetLabelsSeries (Axis : TChartAxis), TChartAxis由ParentChart 得到它所在的图表，再由图表的GetLabelsSeries方法取得它所要求的序列。

3.2 序列与数据源的绑定

TchartSeries类中定义属性DataSource指明序列的数据源。数据源的可用类型有：TDataSet（有TTable, TQuery, TClientDataset等)和其它序列的算术运算（function）结果。如果没有指定数据源，则需采用人工增加point的方式来绘制序列。TChartSerie和数据源的关系如图3 示。关于TChartSeries在X，Y坐标轴上的值，TChartSeries中定义了XValues和YValues（类型为TChartValueList）。在TchartValueList中定义两个属性，一个是ValueSource，记录它所绑定的字段的名称（string）；另一个是Value 数组，由ValueSource对应字段的各记录的值组成。例：将数据源Table1 的字段'Quantity' 指定给序列LineSeries1的Y轴，字段"Time"指定给序列LineSeries1的X轴：

```
LineSeries1.DataSource := Table1 ;
/* 给序列LineSeries1指定数据源DataSource */
LineSeries1.XValues.ValueSource := 'Time' ;
        /* 给序列LineSeries1指定X轴的数据 */
LineSeries1.YValues.ValueSource := 'Quantity' ;
  /* 给序列LineSeries1指定Y轴的数据*/
```

数据源算术运算的实现：一个序列的数据源可以是其它几个序列的数据源进行算术

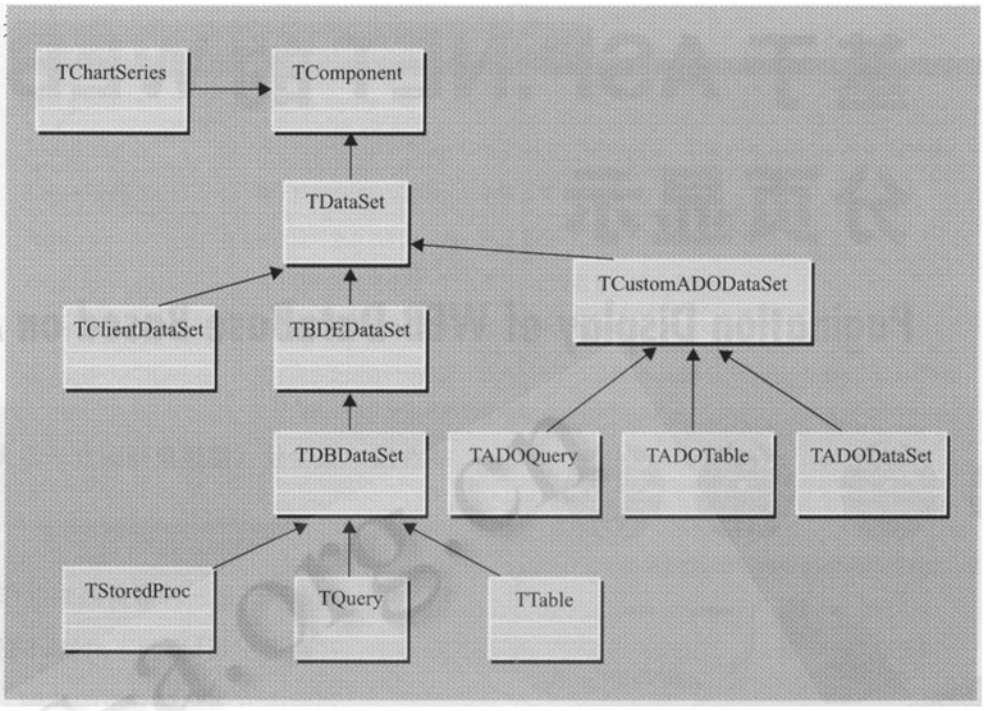

图 3 TchartSerie 和数据源的关系

结果，如：series3 = （series1 + series2 ）/ 2；为实现此功能，TChartSeries定义了一个属性DataSources,它是一个指针列表（TList），列表中的每个元素指向一个参加运算的序列。下面例子定义一个新的序列series3，其各点的Y轴取值为series1和series2的Y轴取值的平均值。

```
With series3.DataSources do
begin
  Add( Series1 );    Add( Series2 );
end;
series3.SetFunction( TAverageTeeFunction.Create(Self) );
```

4 应用示例

4.1 不同计量单位的 series 共存于同一个 chart 中

通常情况下，每一个series都采用底坐标轴和左坐标轴作为X轴和Y轴。特殊地，可以指定序列采用哪一个坐标轴作为X轴和Y轴。TeeChart 为series所采用的每一个axis计算最大值和最小值。当同一个图表中的多个series在坐标轴上采用不同的的计量单位时（数值类型不同），将导致它们的取值范围不同。这时，为达到良好的显示效果，最好给它们各自指定X轴（或Y轴）。如：LineSeries1在X轴上的数值类型是时间，LineSeries2在X轴上的数值类型是‘非时间’的数据类型。可采用下列方法给二者分别指定不同的X轴。

```
LineSeries1.XValues.DateTime := True ;
LineSeries1.HorizAxis  := aBottomAxis ;  /* LineSeries1的X轴采用下坐标轴*/
LineSeries2.XValues.DateTime := False  ;
LineSeries2.HorizAxis  := aTopAxis ;     /* LineSeries2的X轴采用上坐标轴*/
```

4.2 打印

下面例子完成打印图表Chart1的功能。

下转第53页 >>

```
    int tmpW, tmpH, tmpWMargin, tmpHMargin;     // 宽度，高度和页边变量
    Printer()->Orientation = poLandscape; // 设置为横向打印
    Printer()->BeginDoc(); // 开始打印作业
    Printer()->Title = "打印标题";
    Chart1->PrintResolution= -100;          // 设置打印分辨率。-100 = printer Resolution
    tmpW = Printer()->PageWidth; // 取得打印纸张宽度
    tmpWMargin = floor(5.0 * tmpW/100.0);    // 指定 5% 的页边
    tmpW = tmpW - 2 * tmpWMargin;  // 实际打印宽度= 纸张宽度－2个页边
    tmpH = Printer()->PageHeight;          // 取得打印纸张高度
    tmpHMargin = floor(5.0 * tmpH/100.0);     // 指定 5% 的页边
    tmpH = tmpH - 2 * tmpHMargin;          // 实际打印高度= 纸张高度－2个页边
    Chart1->PrintPartial(                  // 打印函数
        Rect (tmpWMargin , tmpHMargin, tmpWMargin + tmpW , tmpHMargin + tmpH ) );
    Printer()->EndDoc();                   // 结束打印作业
```

5 结束语

TeeChart控件有许多优良的特性，且还在不断的更新和发展。笔者认为，只有以面向对象的思想和观点去分析、运用它，在程序的开发过程中，采用面向对象方法和技术，这样才能将TeeChart控件与整个程序有机地结合起来，开发出高质量的软件。

参考文献

1 Borland C++ Builder Help

2 Rational Rose 98 Help