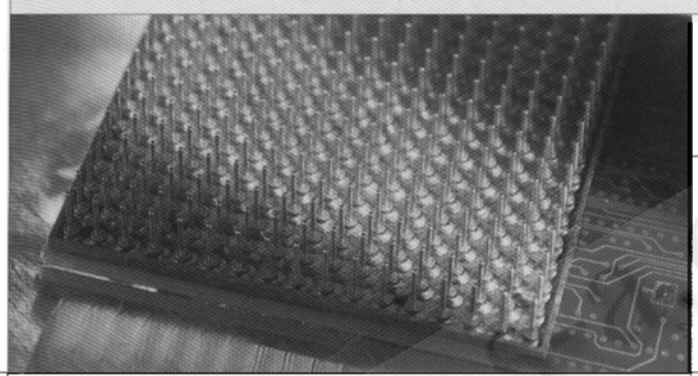
在C#中实现TREEVIEW内容加载

The Implement of Adding TreeView Data on MicroSoft C#



徐立娟 文建全 （湖南铁道职业技术学院）

摘 要：本文介绍了一种在MIROSOFT C#中用递归算法实现TREEVIEW内容加载的方法。

关键词：递归 TREEVIEW C# 表结构 数据 父结点 子结点

1 引言

MIROSOFT C#是微软公司最新推出的基于.NET平台的开发工具，TreeView是一个重要的控件，无论是在VB.NET，C# 还是VB、Delphi等各种语言中，都充当了导航器的作用。在实际工作中，很多情况下需要将TreeView与数据库进行连接，以填充其节点。在Windows Form和Web Form中，我们可以用TreeView来显示树形结构，如显示目录树、显示地区、分类显示商品等。可以说，在大部分软件的开发中，TreeView都是一个不可缺少的展示控件。因此，树形结构的设计就成了软件开发人员一个永恒的话题。下面就以一个具体目录树的例子介绍这段程序如何实现。

2 实现步骤

2.1 TREEVIEW 的内容加载方式

树形结构的内容加载一般来讲有三种方式：

（1）界面设计时在TreeView设计器或者代码中直接填充TreeView控件。

（2）从XML文件中建立树形结构。

（3）从数据库中得到数据，建立树形结构。

第一种方式是最简单的，这种方式主要用于树形结构一般没有变化的应用程序，第二种方式从XML文件中提取，由于XML本身就是树形结构的，微软提供的文档对象模型DOM 可以方便的读取、操作和修改 XML 文档。在.NET中，应用System.Xml类可以方便地将XML文件加载到TreeView控件中，此处就不再多说。第三种方式，树形结构的数据，从数据库中获得。一般来讲，我们的应用程序多数是基于数据库的。采用这种方式，增加、修改、删除一颗树的节点很方便，只要操作数据库中的数据就可以了。而且，这种方式可以和数据库中的其它表做关联、查询和汇总，通过设计视图或存储过程，很容易查询出你想要的相关数据。下面，我们主要讨论这种方式的设计和实现。

2.2 数据库的设计

在设计表结构时必需考虑记录之间的联系记录的层次，所以设计了ID和父ID字段，每条记录的ID是唯一且不为空值（一般设计为自动增长，步长为1的字段类型），以保证数据第一级父ID（根记录）为0或空值，所有子级父ID为其上级的ID。这样就使整个表中数据形成一个树形结构。下面以一个后台数据库为DB2的简化的表结构为例说明。

```
CREATE TABLE "TREESAMPLE" (
"ID" DECIMAL(18,0) NOT NULL GENERATED BY DEFAULT AS IDEN-
TITY ( START WITH +1 , INCREMENT BY +1 , CACHE 20 , MINVALUE
+1 , MAXVALUE +9999999999999999999 ) ,
"P_ID" DECIMAL(18,0) NOT NULL WITH DEFAULT 0,//父ID
"NAME" VARCHAR(100) ,//结点名
"CONTENT" VARCHAR(1000),//内容描述
"PYCODE" VARCHAR(50),//拼音码
"WBCODE" VARCHAR(50),//五笔码
"CREATDATE" TIMESTAMP) //记录产生时间
IN "BASESPACE" INDEX IN "IDXSPACE" ;
//建立索引
CREATE INDEX."PK_TREESAMPLE" ON "TREESAMPLE"
("ID" ASC, "P_ID" ASC) CLUSTER ;
```

在上述表结构定义中ID与P_ID是一对多的关系，P_ID选择默认值为0，单没有指定父ID时认为它是根记录。

2.3 SQL查询

要在TREEVIEW控件中把树结构显示出来，首先要得到数据，下面函数通过父ID返回属于该父ID的数据集，cCon是数据库连接参数。

```
private DataTable GetContent(OleDbConnection cCon,int pid)
{
  DataTable myTb=new DataTable();
  //SQL查询字符串
   string sSql="select id,p_id,name from TREESAMPLE where p_id="+pid +"order by id";
  myTb=myOpen(cCon,sSql);
  return myTb;//返回记录集
      }
```

2.4 用递归函数实现数据加载

在C#中，TreeView树的节点是一个集合，每个 TreeNode 都可以包含其他 TreeNode 对象的集合。要确定您在树结构中的位置，得使用 FullPath 属性。我们知道，添加节点只能是在找到节点之后在此节点下添加。现在C#少了Key属性，对操作是一个很大的不便。所以，添加许多层节点的树形结构，只能是递归调用。下面的代码很精炼，只需要传给递归过程一个父ID，就会将这个编号下的所有节点加载到树形结构中！充分体现了：简单就是好的思想。

```
private void  creat_node(string        name,TreeNode t1,int pId)
{
//定义一个新的结点
   TreeNode t2=new TreeNode();
   t2.Tag=pId;
   t2.Text=name;
   //判断是否为根结点
   if(t1==null)
   {
    tvContent.Nodes.Add(t2);
   }
   else
   {
    t1.Nodes.Add(t2);
              }
   //调用函数取得这个结点的所有子结点，利用递归自动生成
   DataTable myTb1=GetContent(cCon, pId);
   for (int i = 0; i < myTb1.Rows.Count; i++)
   {
     creat_node( myTb1.Rows[i]["name"].ToString ().Trim (),t2,Int32.Parse ( myTb1.Rows[i]["id"].ToString ()));
   }
}
```

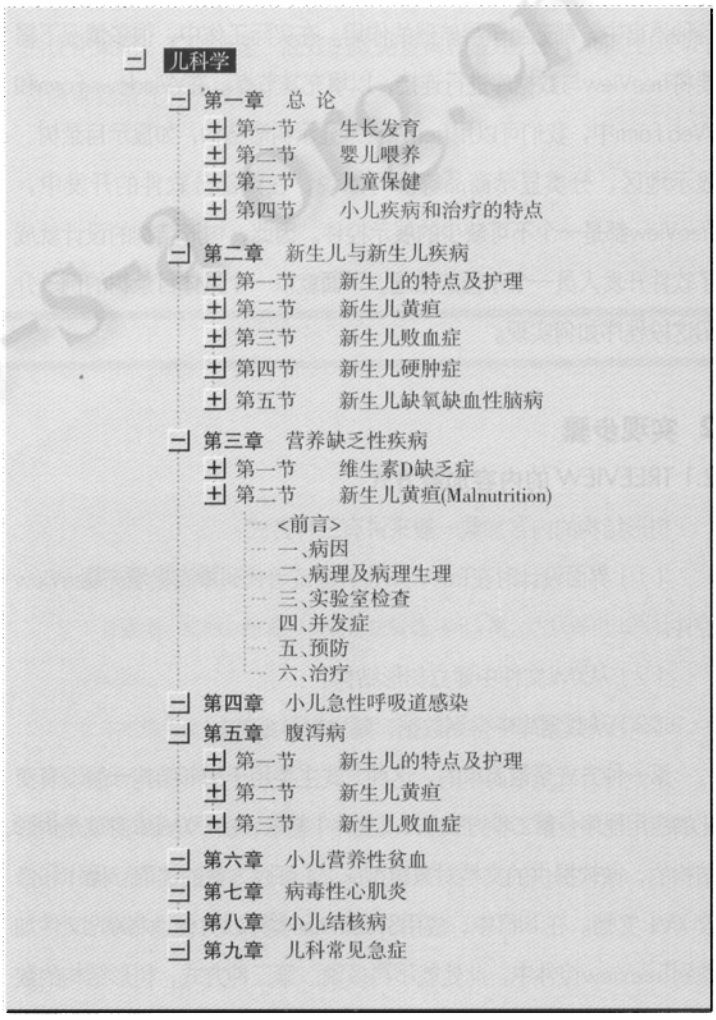
2.5 实例展示

在微软的.NET开发环境中，首先创建一个工程，然后从工具栏中将TREEVIEW控件拖放到设计窗体上，具体过程比较简单，这里就不做详细描述，读者可参考相关的C#书籍即可。

在窗体加载时调用初始化树结构函数就能完成对树的内容加载。

```
private void InitTreeView()
{
TreeNode pNode=new TreeNode ();
   pNode=null;
   //调用递归函数，完成树形结构的生成
   creat_node("儿科学",pNode,7171);
}
```

下图是一个实际产生的目录导航图，读者可以自己试一试。



(1) 遍历TreeView节点(递归算法)

```
void GetAllNodeText(TreeNodeCollection tnc)
{
foreach(TreeNode node in tnc)
    {
        if(node.Nodes.Count!=0)
            GetAllNodeText(node.Nodes);
      MessageBox.Show(node.Text + " ");
        } .
        }
```

(2) 得到所选节点的Text，ID或NodeData

```
private TreeNode searchTree(string txt)
{
  for(int i=0;i<tvContent.Nodes.Count;i++)
  {
        if(tvContent.Nodes[i].Text==txt)
        {
            return tvContent.Nodes[i];
        }
  }
        return null;
}
```

(3) 实现结点的拖放功能

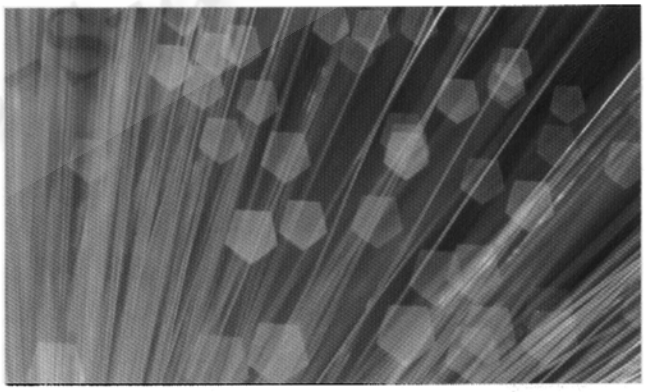
假定TREEVIEW命名为tvContent，实现拖放功能的三个事件函数如下。

```
private void tvContent_ItemDrag(object sender, System.Windows.
Forms.ItemDragEventArgs e)
    {
        NodeToBeDeleted = (TreeNode)e.Item;
        string strItem = e.Item.ToString();
        DoDragDrop(strItem, DragDropEffects.Copy |
DragDropEffects.Move);

    }
    private void tvContent_DragEnter(object sender, System.Windows.
Forms.DragEventArgs e)
    {
        if (e.Data.GetDataPresent(DataFormats.Text))
        e.Effect = DragDropEffects.Move;
        else
            e.Effect = DragDropEffects.None;
    }
    private void tvContent_DragDrop(object sender, System.Windows.
Forms.DragEventArgs e)
    {
        Position.X = e.X;
        Position.Y = e.Y;
        Position = tvContent.PointToClient(Position);
        TreeNode DropNode = this.tvContent.GetNodeAt
(Position);
        if (DropNode != null )
        {
            TreeNode DragNode = this.NodeToBeDeleted;
            DropNode.Parent.Nodes.Remove(this.
NodeToBeDeleted);
            DropNode.Parent.Nodes.Insert(DropNode.
Index+1, DragNode);
        }
    }
```

3 结束语

在C#中TREEVIEW还有许多事件和方法，在这里不能一一列出，文中方法希望能对大家进行TREEVIEW编程过程中有所帮助。



参考文献

1 Karli Watson,Marco Bellinaso著，康博译，C#入门经典，清华大学出版社，2002年。

2 刘鎏柏著，DB2入门与提高，清华大学出版社，2002年。