生成日期: 2013年9月17日星期二

# TreeView 类

## 简介

### 父类

Control

### 简介

带有根节点和子节点的树形数据展示.

### 详细说明

这是一个Control类, 以多级父节点与子节点的形式, 依照用户设置的模板ItemFactory显示用户通过DataModel设置的数据. ItemFactory与DataModel必须由用户设置, 且需要满足一定的条件. 见ItemFactory与DataModel小节.

布局上的Horizontal ScrollBar和Vertical ScrollBar可以通过方法替换.

布局说明

滚动条:

和ItemView不同, TreeView会渲染所有被展开的节点, 不管是不是在界面上的可见范围内. 这样布局的结构比ItemView实现起来简单很多.

在TreeView Control内的根控件是ScrollPanel, 这是一个带有两个滚动条和一个内部对象的控件. 滚动条HScrollBar和VScrollBar会控制内部对象的位置. 内部对象的大小会影响滚动条的滚动范围.

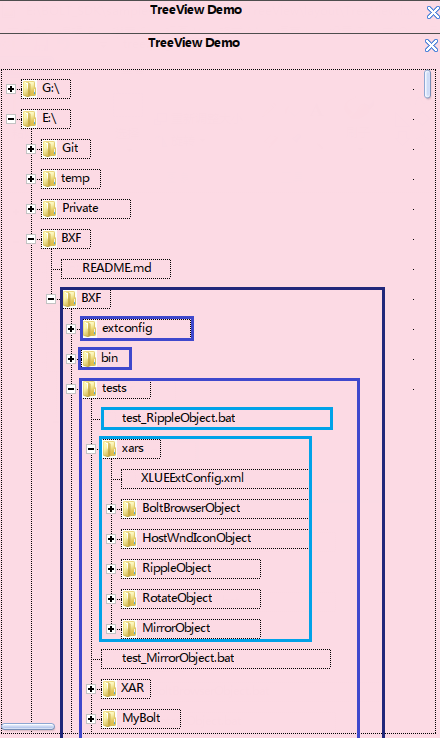
TreeView会把包含所有节点对象的layout传给scroll panel作为内部对象来显示.

父节点, 子节点layout关系:

TreeView为每个节点创建一个layout和一个节点UI对象. 节点layout的children包括节点对象和所有子节点的layout. 如果没有子节点, 节点layout和节点UI对象时重合的.

数据结构:

因为会频繁用到size, pos, rectangle这样的结构, 在此统一说明文档/代码中涉及的XXXsize即{width=int, height=int}格式的表, XXXpos即{left=int, top=int}, XXXRect即{left=int, top=int, width=int, height=int}. 如果用户自己定义样式, 样式回调方法的返回值也需要是相应格式的表.



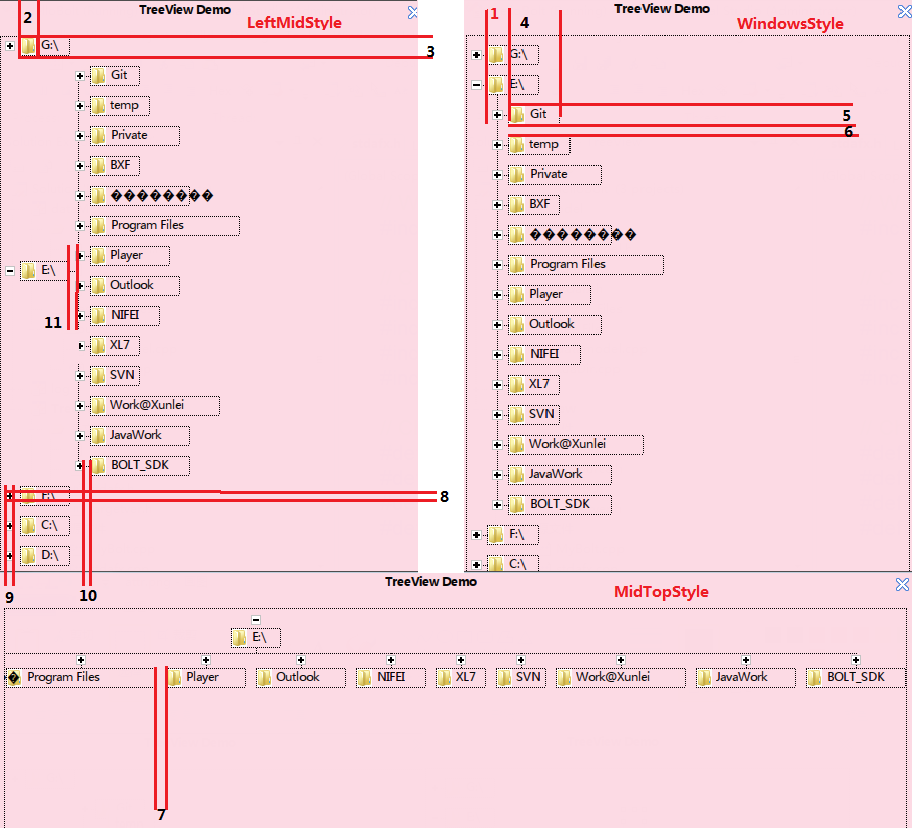
### 成员方法列表

|  |  |
| --- | --- |
| 名称 | 简介 |
| [RenderTreeFromDataTable](#_RenderTreeFromDataTable()方法)() | 显示table存储的数据树 |
| [SetStyle](#_SetStyle方法)() | 设置样式 |
| [SetItemFactory](#_SetItemFactory()方法)() | 设置创建节点对象的Item Factory |
| [SetTreeModel](#_SetTreeModel_()方法)() | 设置数据树的DataModel |
| [SetHScrollBar](#_SetHScrollBar方法)() | 设置水平滚动条 |
| [SetVScrollBar](#_SetVScrollBar方法)[HorizontalScrollPosChanged事件](#_HorizontalScrollPosChanged事件)() | 设置垂直滚动条 |

### 属性列表

|  |  |  |  |
| --- | --- | --- | --- |
| 名称 | 类型 | 默认值 | 描述 |
| style\_name | String | WindowsStyle | 样式名称 |
| vertical\_indent | int | 0 | 垂直方向父子节点相对缩进 |
| horizontal\_indent | int | 30 | 水平方向父子节点相对缩进 (1) |
| icon\_width | int | 50 | 图标宽度 (2) |
| icon\_height | int | 20 | 图标高度 (3) |
| node\_width | int | 200 | 节点对象宽度 (4) |
| node\_height | int | 20 | 节点对象高度 (5) |
| vertical\_padding | int | 50 | 垂直方向同级节点间距 (6) |
| horizontal\_padding | int | 50 | 水平方向同级节点间距 (7) |
| joint\_open\_icon | string | nil | 展开节点图标 |
| joint\_close\_icon | string | nil | 收拢节点图标 |
| joint\_width | int | 15 | 展开/收拢节点图标宽度 (8) |
| joint\_height | int | 15 | 展开/收拢节点图标高度 (9) |
| joint\_node\_padding | int | 0 | 展开/收拢节点图标和节点间距 (10) |
| knob\_size | int | 5 | 节点到展开/收拢图标之间的间距 (11) |

样式不一定会用到全部属性, 像水平间距horizontal\_padding(7)就只在同级节点水平排开的时候才会用到. 下图标出了以上数值属性值所表示的距离.



### 事件列表

|  |  |
| --- | --- |
| 名称 | 简介 |
| [HorizontalScrollPosChange](#_HorizontalScrollPosChanged事件) | 水平滚动条位置变化 |
| [VerticalScrollPosChange](#_VerticalScrollPosChanged事件) | 垂直滚动条位置变化 |
| [VisibleRectChanged](#_VisibleRectChanged事件) | 可见视窗Rect变化 |

TreeView没有提供节点被选中, Hover 或者 Click的事件. 因为由使用者提供的Item Factory来创建节点对象, 用户自己的节点对象可以自定义各种消息，因为用户自定义item控件的zorder比较高，所以可以优先接收鼠标和键盘消息

## 成员方法说明

### RenderTreeFromDataTable()方法

原型

void RenderTreeFromDataTable(table dataTable)

简介

渲染通过Table存储的树形数据.

详细说明

当用户不想自己实现DataModel而且数据结构比较简单(数据不会发生变化, 父节点key和value一致)时, 可以用table来存储树形数据.

例如:

{“A”={“C”=”LEAF 1”, “D”=”LEAF 2”}, “B”={“E”=”LEAF 3”}}

表示树

---- (key:””, value:nil)

---- (key:A,value:A)

---- (key:C, value:LEAF 1)

---- (key:D, value:LEAF 2)

----(key:B, value:B)

----(key:E, value:LEAF 3)

函数参数

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 类型 | 默认值 | 传入传出 | 描述 |
| dataTable | table | nil | 传入 | 树形数据表 |

返回值

|  |  |
| --- | --- |
| 类型 | 描述 |

参考

SetTreeModel()

### SetStyle方法

原型

void SetStyle (style)

简介

设置treeview使用样式

详细说明

样式控制treeview的节点, 连接线和展开/收拢按钮的位置, 是一个会被treeview回调的方法列表. TreeView会检查并调用style的如下方法

函数参数

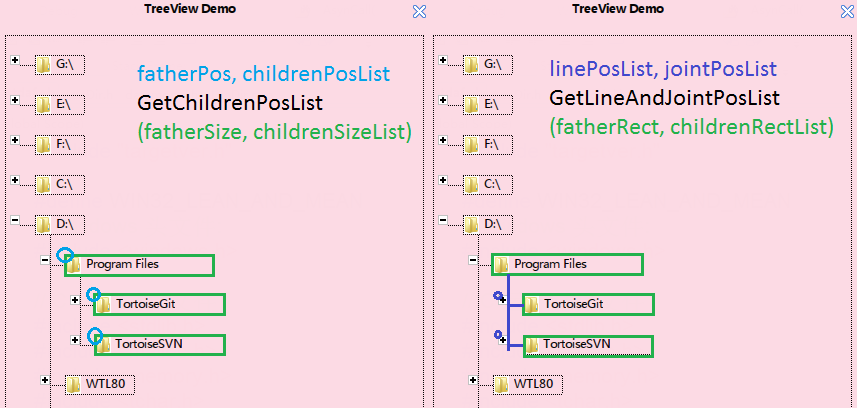
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 类型 | 默认值 | 传入传出 | 描述 |
| style | table | nil | 传入 | 回调方法列表 |

参数style

这是一个table, key是方法名, value是函数, 总览如下:

|  |  |  |  |
| --- | --- | --- | --- |
| 键 | 函数原型 | 必需/可选 | 说明 |
| GetChildrenPosList | fatherPos, childrenPosList GetChildrenPosList(fatherSize, childrenSizeList) | 必需 | 给定父节点与子节点的大小, 返回父节点与子节点在父节点layout中的位置 |
| GetLinesAndJointPosList | linePosList, jointPosList GetLineAndJointPosList(fatherRect, childrenRectList) | 必需 | 给顶贴父节点与子节点的geometry (rect), 返回连接线位置列表和展开/合并按钮位置列表 |
| node\_size | {width=int, height=int} | 可选 | 返回节点大小. 如果节点大小不是统一的可以不指定, 在ItemFactory创建节点对象时再返回大小. |

图示一下, 绿色表示输入, 蓝色表示输出



返回值

|  |  |
| --- | --- |
| 类型 | 描述 |

参考

### SetItemFactory()方法

原型

void SetItemFactory(ItemFactory)

简介

设置创建节点对象的ItemFactory

详细说明

TreeView会检查并调用ItemFactory的” CreateNodeObject”方法.

用户必须提供ItemFactory以创建节点对象, 如果用户不提供, TreeView没有默认行为.

ItemFactory[“CreateNodeObject”]需为方法, 方法接受参数为节点data和节点key, 返回节点对象实例和节点对象大小 (需要返回节点对象大小是因为在绑定到ObjectTree之前UI对象的尺寸不可得), 如果返回对象大小size为nil, 则TreeView会使用SetStyle(style)的参数style的node\_size作为默认尺寸.

如果用户需要每个node object有不一样的尺寸, 那么在itemFactory.CreateNodeObject()被调用时返回; 如果用户只需要统一的尺寸, 那么在itemFactory.CreateNodeObject()不返回size, 转而在style.node\_size指定是个好办法.

函数参数

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 类型 | 默认值 | 传入传出 | 描述 |
| itemFactory | table | nil | 传入 | 回调方法列表 |

参数itemFactory

这是一个table, key是方法名, value是函数, 总览如下:

|  |  |  |  |
| --- | --- | --- | --- |
| 键 | 函数原型 | 必需/可选 | 说明 |
| CreateNodeObject | object, size  CreateNodeObject(data, key) | 必需 | 给定节点key和value, 返回节点对象和节点对象大小 |

返回值

|  |  |
| --- | --- |
| 类型 | 描述 |

参考

### SetTreeModel ()方法

原型

void SetTreeModel (treeModel)

简介

设置treeView显示的树形数据来源.

详细说明

当treeView的一个节点被展开时, treeView会调用treeModel传入的回调方法得到当前节点的所有子节点信息, 并创建节点对象显示. treeModel是一个回调方法table, key是方法名, value是函数.

作为函数参数和返回值的节点Node的数据类型:

Node的类型是table, treeView预设并会使用Node如下key:

Node.Key 节点键值, 类型string

Node.Children 子节点列表, 类型是table

Node.Data 节点数据, 类型自定义

Node.Expand 节点是否展开, 类型bool

函数参数

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 类型 | 默认值 | 传入传出 | 描述 |
| treeModel | table | nil | 传入 | 回调方法列表 |

参数treeModel

这是一个table, key是方法名, value是函数, 总览如下:

|  |  |  |  |
| --- | --- | --- | --- |
| 键 | 函数原型 | 必需/可选 | 说明 |
| GetChildren() | children GetChildren(fatherNode) | 必需 | 返回fatherNode的所有子节点 |
| HasChildren() | bool HasChildren(fatherNode) | 必需 | 返回fatherNode是否有子节点 (叶子节点false, 其他节点true) |
| GetFather | Node GetFather(node) | 必需 | 返回node节点的父节点 |
| GetChild() | Node GetChild(fatherNode, key) | 必需 | 返回fatherNode键值为key的子节点 |
| GetRoot() | Node GetRoot() | 必需 | 返回tree的根节点 |

返回值

|  |  |
| --- | --- |
| 类型 | 描述 |

参考

### SetVScrollBar方法

原型

void SetVScrollBar (UIObject scrollbar)

简介

设置垂直滚动条对象

详细说明

使用者如果不想使用TreeView默认的scrollbar, 可以在TreeView初始化时调用这个接口来设置滚动条对象.

|  |  |  |  |
| --- | --- | --- | --- |
| 键 | 函数原型 | 必需/可选 | 说明 |
| GetScrollPos | int GetScrollPos() | 必需 | 返回当前滚动条的位置 |
| SetScrollPos | SetScrollPos(int pos) | 必需 | 设置当前滚动条位置为pos |
| GetScrollRange | int min, int max GetScrollRange() | 必需 | 返回滚动条最大最小范围 |
| SetScrollRange | SetScrollRange(int min, int max) | 必需 | 设置滚动条范围 |

函数参数 (SetScrollBarV的函数参数)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 类型 | 默认值 | 传入传出 | 描述 |
| Scrollbar | UIObject | nil | 传入 | 垂直方向滚动条对象. |

返回值

|  |  |
| --- | --- |
| 类型 | 描述 |

参考[SetHScrollBar](#SetScrollBarH)

### SetHScrollBar方法

原型

void SetHScrollBar (UIObject scrollBar)

简介

设置水平滚动条对象

详细说明

见[SetVScrollBar](#SetScrollBarV)的详细说明. 除了方向不一样, 二者的使用方法是一样的.

函数参数

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 类型 | 默认值 | 传入传出 | 描述 |
| Scrollbar | UIObject | nil | 传入 | 水平方向滚动条对象. |

返回值

|  |  |
| --- | --- |
| 类型 | 描述 |

参考[SetVScrollBar](#SetScrollBarV)

## 事件说明

### VerticalScrollPosChanged事件

原型

void VerticalScrollPosChanged(int oldPos, int newPos)

简介

滚动区域垂直方向滚动值发生变化

详细说明

如果用户不想自己设置ScrollBar, 但是又需要监听ItemView的滚动事件, 可以监听这个事件

函数参数

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 类型 | 默认值 | 传入传出 | 描述 |
| oldPos | Int | 无 | 传入 | 之前的滚动位置 |
| newPos | Int | 无 | 传入 | 新的滚动位置 |

返回值

参考 [HorizontalScrollPosChanged](#HorizontalScrollPosChanged)

### HorizontalScrollPosChanged事件

原型

void HorizontalScrollPosChanged(int oldPos, int newPos)

简介

滚动区域水平方向位置发生变化

详细说明

如果用户不想自己设置ScrollBar, 但是又需要监听ItemView的滚动事件, 可以监听这个事件

函数参数

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 类型 | 默认值 | 传入传出 | 描述 |
| oldPos | Int | 无 | 传入 | 之前的滚动位置 |
| newPos | Int | 无 | 传入 | 新的滚动位置 |

返回值

参考 [VerticalScrollPosChanged](#VerticalScrollPosChanged)

### VisibleRectChanged事件

原型

void VisibleRectChanged(left, top, width, height)

简介

视窗可见rect发生变化, 参数表示新rect的左上宽高.

详细说明

水平/垂直方向滚动, 或者视窗大小发生变化时这个事件都会被触发.

函数参数

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 名称 | 类型 | 默认值 | 传入传出 | 描述 |
| left | Int | 无 | 传入 | 左 |
| top | Int | 无 | 传入 | 上 |
| width | Int | 无 | 传入 | 宽 |
| height | Int | 无 | 传入 | 高 |

返回值

参考

## 使用用例

文件:

TreeView.xml和TreeView.lua分别定义了TreeView控件的布局和方法

Scrollpanel.xml和scrollpanel.lua分别定义了scrollpanel控件的布局和方法. scrollpanel控件被treeview控件引用, 作为滚动显示树形数据的容器.

TreeViewStyle.lua定义了三种TreeView自带样式: “WindowsStyle”, “LeftMidStyle”和”MidTopStyle”. 它们所需的默认属性值在TreeView.xml中已定义(是否另外文件TreeViewStyle.xml定义更好?). 使用这三种样式之一时只需要在引用TreeView的control部分指定templateid即可.

joint.xml和joint.lua定义了展开/收起按钮

以上是控件定义

以下是用户代码

MainWnd.xml为Demo主窗口, 引用了TreeView, 并通过templateid指定它的style是windowsStyle

<obj id="tree" class="TreeView" templateid="WindowsStyle" >

<attr>

<left>5</left>

<top>35</top>

<width>father.width-10</width>

<height>(father.height-40)</height>

</attr>

<eventlist>

<event name="OnInitControl" file="MainWnd.xml.lua" func="OnInitTree" />

</eventlist>

</obj>

MainWnd.xml.lua指定了TreeView使用的ItemFactory, TreeModel

TreeModel.lua定义了设置给TreeView的data model. 我们调用GetDirTreeModel返回一个包含了系统路径信息的data model. 子路径信息通过一个注册到lua环境中的c++对象LuaDir.Factory.Object返回. 只有当TreeView调用GetChildren()时, TreeModel才去访问子路径列表.

ItemFactory.lua定义了设置给TreeView的创建对象实例方法CreateNodeObject. 根据当前节点表示文件夹还是具体文件显示不同的图标和文件(夹)名称.

Node.xml定义了一个虚线框框起来的图片对象和文字对象, 这就是CreateNodeObject创建的对象模板

如果用户想要自己实现style, 可以参考TreeViewStyle.lua返回一个table. 需要的关键字和方法在[SetStyle方法](#_SetStyle方法)中有列出. 需要注意的是如果Style需要访问TreeView的attributes, 可以在初始化的时候一次读取存在Style中.