

新浪博客

[\[公告\]](#)

[一去、二三里](#)

[个人中心](#)

[发博文](#)

[消息](#)



心在山水间

一去、二三里的博客

<http://blog.sina.com.cn/liang19890820>

[首页](#) | [博文目录](#) | [图片](#) | [关于我](#)

发博文

页面设置

个人中心

个人资料

[管理]

一去、二三里

Qing

微博

博客等级: 18

博客积分: 78 新

博客访问: 407,756

关注人气: 378

获赠金笔: 12

赠出金笔: 0

荣誉徽章:

相关博文

▪ 流元素（二）
一去、二三里

▪ QTableView添加复选框
一去、二三里

▪ Qt之模型/视图（实时更新数据）
一去、二三里

▪ Qt之QTreeView（二）
一去、二三里

▪ QwtPlot之绘制统计图（三）
一去、二三里

▪ Vector与数组中间的转化
Eamon

▪ errorLNK2019: unresolved externals journey

▪ QwtPlot之绘制统计图（二）
一去、二三里

▪ QCustomPlot配置、使用
一去、二三里

▪ QWT编译、配置、使用（QtCreator）
一去、二三里

更多>>

正文

字体大小: 大 中 小

Qt之QTreeView (一)

 (2014-12-23 16:51:33) [编辑][删除]

+ 转载 ▼

标签: qt qtreeview qtableview qabstractitemmodel 分类: Qt

之前有讲解过QTableView的使用Qt之QTableView, 这节讲解一下也较为常用的另外一个部件QTreeView, 对于多层结构的显示来说, QTreeView无非是最佳的选择。

Qt中有几种纯粹的视图部件: QListView、QTableView、QColumnView、QTreeView, 所有的这些视图都必须提供一个模型(无论是自定义, 还是Qt中已提供的)来与之配合。Qt仍然提供了一些便利的窗口部件(“便利”是因为它们提供了自己内置的模型, 并能直接使用), 如: QListWidget、QTableWidget、和QTreeWidget。还有QComboBox, 既是一个便利的窗口部件也是一个视图部件, 也就是说, 我们既可以直接使用(因为它提供了内置的模型), 也能把它当做一个模型的视图部件(这种情况下, 可以提供一个合适的模型给它)。

之所以说视图部件常用, 是因为在编程的过程当中经常遇到大数据集, 使用视图/模型就显得更有效率。当然, 对于数据集较小(数百或数千个项)的应用程序, 选择便利部件比较合适。

这节讲解中主要包括: QTreeView模型视图的使用、自定义委托、自定义样式等。。。

这里提供三种样式, 先上效果图:

自定义TreeView

样式一 ▼

[-] 技术交流, 就是这么任性!

[-] QQ群

C++联盟 (26188347)

Qt 分享交流 (26197884)

技术博客

新浪博客

CSDN博客

[+] 看到这么好的资料, 我都醉了!

第1209篇 • 图腾

评论 |

北美崔哥：美国正称霸世界，春晚

第1207篇 • 日货

某些国家为何拿高铁来戏弄中国？

肖鹰：柴静的意义——她为何值得

第1202篇 • 冤死

北美崔哥：中国式上访，已正式输

蝗虫之日的启示

关于黄圣依奥斯卡之行遇网络攻击



去阿里山赏樱花
观云海



初春美女穿紧身
裤显腿瘦



光猪跑释放自我
亲近自然



中国历朝末代君
主



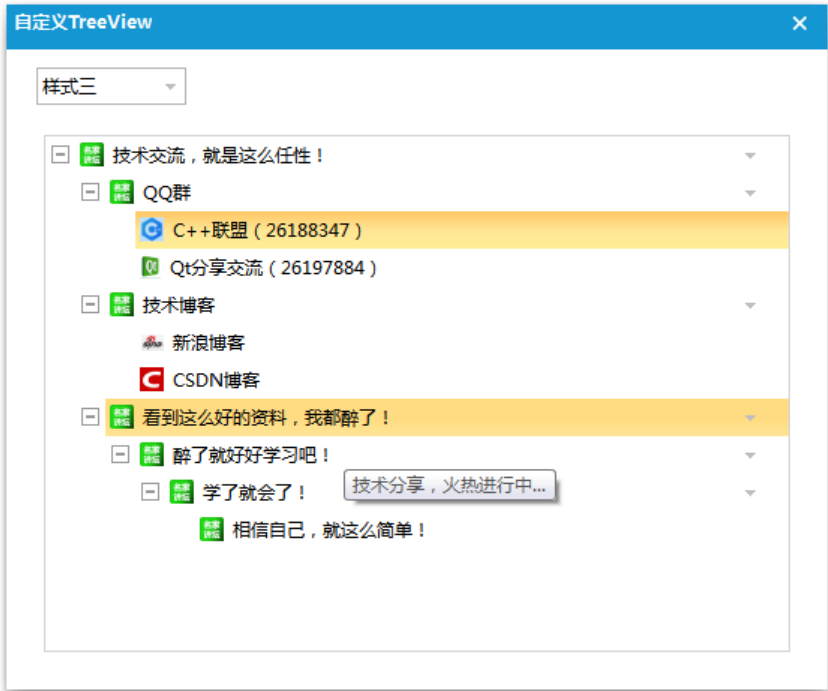
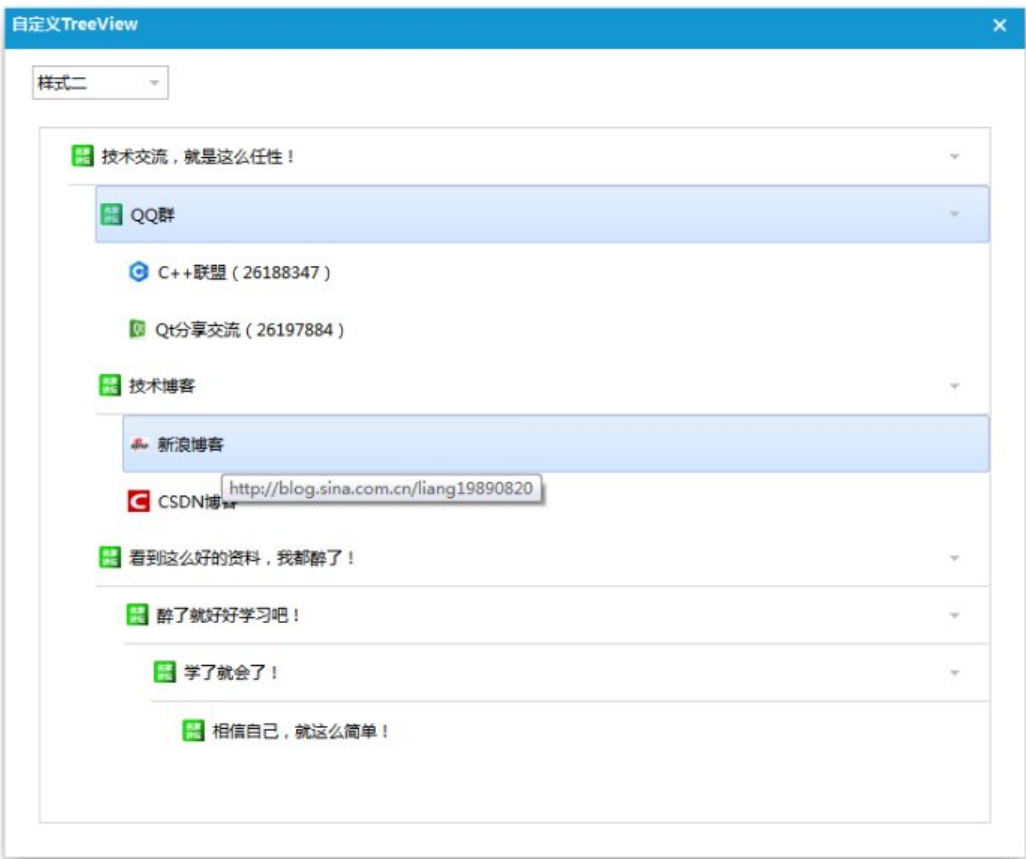
1949年的大上海



太行山村迎城隍

查看更多>>

谁看过这篇博文	
	moonlqer 3月2日
	_wntd_ 2月28日
	大熊 2月27日
	todaysky 2月26日
	右右youyou 2月24日
	王琼夏 2月11日
	守拙——清心 2月11日
	tony2080 2月11日
	tianxin7356 2月6日
	满咲墨染樱 2月6日
	边城菜鸟 2月6日
	Rekols 2月6日



模型/视图篇

(1) QTreeView节点项

```
#include
#include "treeitem.h"

TreeItem::TreeItem(const QList &data, TreeItem *parent)
{
    parentItem = parent;
    itemData = data;
}

TreeItem::~TreeItem()
{
    qDeleteAll(childItems);
}
```

```

void TreeItem::a(TreeItem *item)
{
    childItems.append(item);
}

TreeItem *TreeItem::child(int row)
{
    return childItems.value(row);
}

int TreeItem::childCount() const
{
    return childItems.count();
}

int TreeItem::columnCount() const
{
    return itemData.count();
}

QVariant TreeItem::data(int column) const
{
    return itemData.value(column);
}

TreeItem *TreeItem::parent()
{
    return parentItem;
}

int TreeItem::row() const
{
    if (parentItem)
        return parentItem->childItems.indexOf(const_cast<TreeItem*>(this));

    return 0;
}

```

(2) QAbstractItemModel模型

```

#include "treeitem.h"
#include "treemodel.h"
#include
#include

TreeModel::TreeModel(QObject *parent)
    : QAbstractItemModel(parent)
{
    QList rootData;
    rootData << "Title";
    rootItem = new TreeItem(rootData);
}

TreeModel::~TreeModel()
{
    delete rootItem;
}

void TreeModel::setXML(QString xmlFile)
{
    toolList = Util::parse(xmlFile);
    setupModelData(rootItem);
}

int TreeModel::columnCount(const QModelIndex &parent) const
{
    if (parent.isValid())
        return static_cast<TreeItem*>(parent.internalPointer())->columnCount();
    else
        return rootItem->columnCount();
}

QVariant TreeModel::data(const QModelIndex &index, int role) const
{
    if (!index.isValid())
        return QVariant();
}

```

```
TreeItem *item = static_cast<TreeItem*>(index.internalPointer());

ToolUtil toolUtil;
for (int i = 0; i < toolList.count(); ++i)
{
    toolUtil = toolList.at(i);
    if (toolUtil.id == item->data(0))
    {
        break;
    }
}

if (role == Qt::DisplayRole)
{
    return toolUtil.text;
}
else if (role == Qt::DecorationRole)
{
    return QIcon(Util::exePath() + "\\Resources\\toolicon\\" + toolUtil.toolicon);
}
else if (role == Qt::ToolTipRole)
{
    return toolUtil.tooltip;
}
else
{
    return QVariant();
}
}

Qt::ItemFlags TreeModel::flags(const QModelIndex &index) const
{
    if (!index.isValid())
        return 0;

    return QAbstractItemModel::flags(index);
}

QVariant TreeModel::headerData(int section, Qt::Orientation orientation,
                                int role) const
{
    if (orientation == Qt::Horizontal && role == Qt::DisplayRole)
        return rootItem->data(section);

    return QVariant();
}

QModelIndex TreeModel::index(int row, int column, const QModelIndex &parent)
    const
{
    if (!hasIndex(row, column, parent))
        return QModelIndex();

    TreeItem *parentItem;

    if (!parent.isValid())
        parentItem = rootItem;
    else
        parentItem = static_cast<TreeItem*>(parent.internalPointer());

    TreeItem *childItem = parentItem->child(row);
    if (childItem)
        return createIndex(row, column, childItem);
    else
        return QModelIndex();
}

QModelIndex TreeModel::parent(const QModelIndex &index) const
{
    if (!index.isValid())
        return QModelIndex();

    TreeItem *childItem = static_cast<TreeItem*>(index.internalPointer());
    TreeItem *parentItem = childItem->parent();

    if (parentItem == rootItem)
        return QModelIndex();
}
```

```
        return createIndex(parentItem->row(), 0, parentItem);
    }

int TreeModel::rowCount(const QModelIndex &parent) const
{
    TreeItem *parentItem;
    if (parent.column() > 0)
        return 0;

    if (!parent.isValid())
        parentItem = rootItem;
    else
        parentItem = static_cast<TreeItem*>(parent.internalPointer());

    return parentItem->childCount();
}

TreeItem * TreeModel::item(TreeItem* item, ToolUtil tool)
{
    TreeItem *treeItem = NULL;
    if (item == NULL)
    {
        return treeItem;
    }

    int parentId = tool.parentId;

    if (item->data(0) == parentId)
    {
        treeItem = item;
    }
    else
    {
        for (int j = 0; j < item->childCount(); ++j)
        {
            TreeItem *childItem = item->child(j);
            TreeItem *item2 = this->item(childItem, tool);
            if (item2)
            {
                treeItem = item2;
                break;
            }
        }
    }

    return treeItem;
}

void TreeModel::setupModelData(TreeItem *parent)
{
    QList<TreeItem*> parents;
    parents << parent;

    for (int i = 0; i < toolList.count(); ++i)
    {
        ToolUtil tool = toolList.at(i);
        QList<int> columnData;
        columnData << tool.id;

        for(int j = 0; j < parents.count(); ++j)
        {
            TreeItem* item = this->item(parents.at(j), tool);
            if (item)
            {
                item->a(new TreeItem(columnData, item));
            }
            else
            {
                parents.last()->a(new TreeItem(columnData, parents.last()));
            }
        }
    }
}
```

前几章已经对模型中的方法解释的很清楚了，所以这里就不再过多阐述！

注：
技术在于交流、沟通，转载请注明出处并保持作品的完整性。
作者：`☆奋斗ing♥孩子` 原文：http://blog.sina.com.cn/s/blog_a6fb6cc90102v7q8.html。

2

喜欢

分享：         

阅读 (561) | 评论 (0) | 收藏 (0) | 已有3人转载▼ | 喜欢▼ | 打印

已投稿到： 排行榜

前一篇：QwtPlot之绘制统计图（三）
后一篇：Qt之QTreeView（二）

评论









重要提示：警惕虚假中奖信息

[发评论]

做第一个评论者吧！ 抢沙发>>

发评论

一去、二三里：

☐  分享到微博  ☐ 匿名评论

验证码： 请点击后输入验证码 收听验证码

发评论

以上网友发言只代表其个人观点，不代表新浪网的观点或立场。

＜ 前一篇
QwtPlot之绘制统计图（三）

后一篇 ＞
Qt之QTreeView（二）

