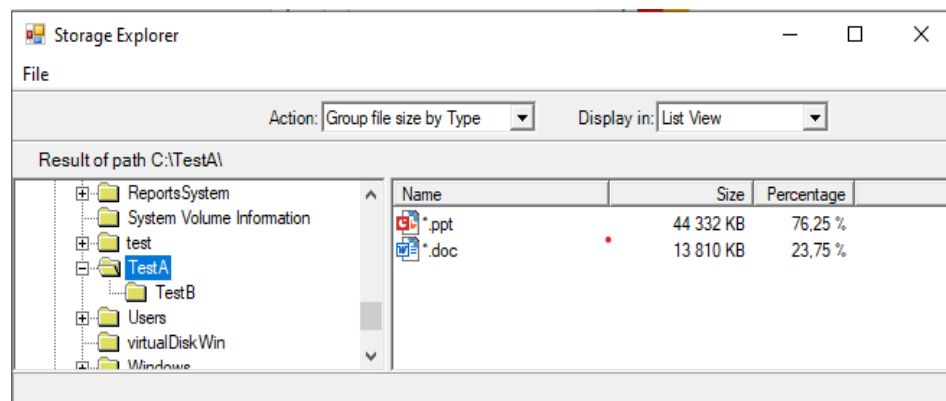


## Тема: Пример создания пользовательской модели данных на основе абстрактного класса

(Повторите тему – Концепция программирования MVC ("модель-вид-контроллер") )

Напомним, что в нашем случае необходимо реализовать возможность отображения данных в виде списка Рис.1.



Рассмотрим простейший пример на основе MVC.

В лекции посвященной теме MVC, рассматривался пример создания пользовательской (*Пример 3. Модель списка строк*) модели (`QAbstractListModel`). В нашем случае данная модель не подходит так как по умолчанию модели, производные от `QAbstractListModel`, содержат только один столбец, поэтому лучше организовать работу с `QAbstractTableModel`.

Так, как нам требуется только отображать данные, то минимальный набор методов необходимых для переопределения будет следующий.

- конструктор модели – с целью обеспечить стандартное создание объектов класса;
- метод `rowCount()` – определение количества строк в модели;
- метод `columnCount()` – определение количества столбцов в модели;
- метод `data()` – позволяет вернуть или изменить элемент данных, соответствующий определенному модельному индексу;
- метод `headerData()` – служит для получения представлениями деревьев и таблиц сведений, которые будут отображаться в их заголовках;

В качестве основных данных, в соответствии с постановкой задачи, это данные, полученные по заданной стратегии, характеризующиеся парой **<Название - значение>**.

В качестве примера предлагается описать данные простым способом, состоящее из трех полей строкового типа. Заметим, что `QString` автоматически конвертируется в `QVariant`, что не требует дополнительной обработки при реализации методов абстрактного класса, взаимодействующих с типом `QVariant`.

```
class SomeData
{
public:
    SomeData(QString nm = "SomeName", QString sz = "SomeSize", QString prc = "SomePercent")
    {
        name = nm;
        size = sz;
        prcent = prc;
    }
    QString name;
    QString size;
    QString prcent;
}
```

```
};
```

Рассмотрим интерфейса класса

```
#ifndef FILEBROWSERDATAMODEL_H
#define FILEBROWSERDATAMODEL_H

#include <QAbstractTableModel>
#include <QList>
#include <QString>

class FileBrowserDataModel : public QAbstractTableModel
{
    Q_OBJECT
public:
    FileBrowserDataModel(QObject *parent = nullptr, QList<SomeData> dt =
    QList<SomeData>());
    //Минимальный и обязательный набор необходимых методов
    //Так как нам требуется только отображать данные, то этого набора достаточно
    int rowCount(const QModelIndex &parent) const;
    int columnCount(const QModelIndex &parent) const;
    QVariant data(const QModelIndex &index, int role) const;
    QVariant headerData(int section, Qt::Orientation orientation, int role) const;

private:
    //Определим перечисление NameColumn для индексации столбцов нашей таблицы.
    enum NameColumn {
        NAME = 0,
        SIZE,
        PERCENT
    };
    //Представим данные модели в виде списка.

    QList<SomeData> dataModel;
};
#endif
```

Рассмотрим реализацию каждого метода.

```
#include "filebrowserdatamodel.h"
//В конструкторе инициализируем исходные данные.
//Причем, обратите внимание, здесь выполняется копирование.
//Если рассматривать наш случай, то в модель придется постоянно передавать данные,
полученные в результате обхода файловой системы по заданной стратегии. Подумайте, как
лучше передавать данные в модель. Может быть потребуется некий метод SetModelData,
который будет устанавливать в модель данные необходимые для отображения????

FileBrowserDataModel::FileBrowserDataModel(QObject *parent, QList<SomeData> dt) :
QAbstractTableModel(parent)
{
    dataModel = dt;
}

//Возвращаем количество строк, в зависимости от количества данных в списке

int FileBrowserDataModel::rowCount(const QModelIndex &parent) const
{
    Q_UNUSED(parent)
    return dataModel.count();
}
```

```

//Возвращаем количество столбцов, оно у нас постоянно
int FileBrowserDataModel::columnCount(const QModelIndex &parent) const
{
    Q_UNUSED(parent)
    return PERCENT + 1;
}

//Возвращаем названия заголовков. Обратите внимание на тип возвращаемого значения.
QVariant FileBrowserDataModel::headerData(int section, Qt::Orientation orientation, int
role) const
{
    if (role != Qt::DisplayRole) {
        return QVariant();
    }

    if (orientation == Qt::Vertical) {
        return section;
    }

    switch (section) {
        case NAME:
            return trUtf8("Название");

        case SIZE:
            return trUtf8("Размер");

        case PERCENT:
            return trUtf8("В процентах");

    }

    return QVariant();
}

//Возвращаем соответствующие данные, относительно модельного индекса и роли.
//В случае, если на вход пришли не корректные данные, что может возникнуть при не
//верной реализации, также в случае, когда приходит роль (int role), которую нам не
//нужно обрабатывать, в частности Qt::DisplayRole и Qt::EditRole , возвращается пустой
//объект QVariant(). Тем самым обеспечивается устойчивая работа представления, при
//отображении модели в случае наличия ошибок.
// Модельный индекс характеризуется номером строки и столбца, следовательно в зависимости
от номера и столбца возвращаются корректные данные, сохраненные в модели данных.

QVariant FileBrowserDataModel::data(const QModelIndex &index, int role) const
{
    if (!index.isValid() ||
        dataModel.count() <= index.row() || (role != Qt::DisplayRole && role !=
Qt::EditRole))
    {
        return QVariant();
    }

    if (index.column() == 0) {
        return dataModel[index.row()].name;
    } else if (index.column() == 1) {
        return dataModel[index.row()].size;
    } else if (index.column() == 2) {
        return dataModel[index.row()].prcent;
    }
}

```

Рассмотрим простейший пример реализации в main.

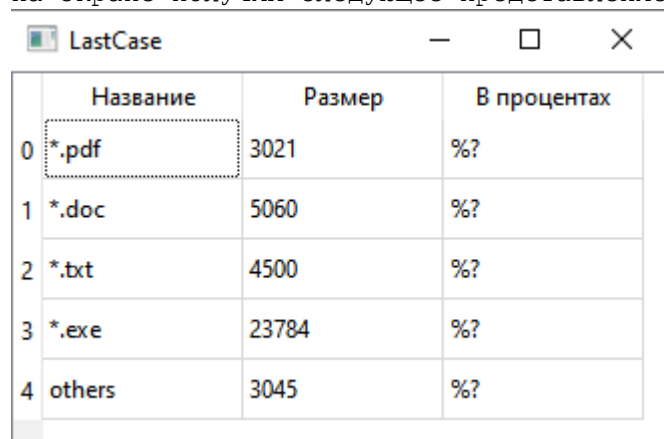
```
#include <QApplication>
#include <QtWidgets>
#include "filebrowserdatamodel.h"

int main(int argc, char *argv[])
{
    QApplication app(argc, argv);
    //Создаем модельные данные и заполняем их
    QList<SomeData> someDataModel;
    someDataModel.append(SomeData("*.pdf", "3021", "%?"));
    someDataModel.append(SomeData("*.doc", "5060", "%?"));
    someDataModel.append(SomeData("*.txt", "4500", "%?"));
    someDataModel.append(SomeData("*.exe", "23784", "%?"));
    someDataModel.append(SomeData("others", "3045", "%?"));

    //Создаем модель
    QAbstractItemModel *tablemodel = new FileBrowserDataModel(nullptr, someDataModel);
    //Создаем представление QTableView

    QTableView *tableView = new QTableView;
    //Устанавливаем модель в представление
    tableView->setModel(tablemodel);
    //Показываем представление
    tableView->show();
    return app.exec();
}
```

На экране получим следующее представление



	Название	Размер	В процентах
0	*.pdf	3021	%?
1	*.doc	5060	%?
2	*.txt	4500	%?
3	*.exe	23784	%?
4	others	3045	%?

**Задание.** Познакомьтесь с предложенным примером. Подумайте над способом представления и передачи новых данных для модели. Выясните в какой момент вызываются следующие методы `rowCount`, `columnCount`, `data`, `headerData`.