  Qt Quick里的ListView，本身是Flickable的派生类，当你用鼠标拖曳或者手指触摸（触摸屏）时，会产生flickStarted和flickEnded两个信号，利用这两个信号，就可以实现下拉刷新数据，当然上拉刷新也是可以的。

    创建一个Qt Quick App项目，添加dynamicModel.h和dynamicModel.cpp两个文件，用于实现DynamicListModel。项目创建过程参考《[Qt Quick 之 Hello World 图文详解](http://blog.csdn.net/foruok/article/details/28850879" \t "_blank)》。

    我们实现的下拉刷新效果有点儿特别，每次刷新后，只保留预定义的一页数据，比如代码中默认的页大小为20。

    版权所有foruok，转载请注明出处：<http://blog.csdn.net/foruok>。

**C++ model实现**

    很简单，直接上代码了。

    dynamic.h：

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/foruok/article/details/39052729)

1. #ifndef DYNAMICMODEL\_H
2. #define DYNAMICMODEL\_H
3. #include <QAbstractListModel>
5. class DynamicListModelPrivate;
6. class DynamicListModel : public QAbstractListModel
7. {
8. Q\_OBJECT
9. Q\_PROPERTY(**int** pageSize READ pageSize WRITE setPageSize NOTIFY pageSizeChanged)
10. Q\_PROPERTY(**int** total READ total WRITE setTotal NOTIFY totalChanged)
11. public:
12. DynamicListModel(QObject \*parent = 0);
13. ~DynamicListModel();
15. **int** rowCount(const QModelIndex &parent) const;
16. QVariant data(const QModelIndex &index, **int** role) const;
17. QHash<**int**, QByteArray> roleNames() const;
19. Q\_INVOKABLE void loadMore(**bool** forward);
21. **int** pageSize();
22. void setPageSize(**int** size);
23. **int** total();
24. void setTotal(**int** total);
26. signals:
27. void pageSizeChanged(**int** size);
28. void totalChanged(**int** total);
30. private:
31. DynamicListModelPrivate \*m\_dptr;
32. };
34. #endif // DYNAMICMODEL\_H

#ifndef DYNAMICMODEL\_H

#define DYNAMICMODEL\_H

#include <QAbstractListModel>

class DynamicListModelPrivate;

class DynamicListModel : public QAbstractListModel

{

Q\_OBJECT

Q\_PROPERTY(int pageSize READ pageSize WRITE setPageSize NOTIFY pageSizeChanged)

Q\_PROPERTY(int total READ total WRITE setTotal NOTIFY totalChanged)

public:

DynamicListModel(QObject \*parent = 0);

~DynamicListModel();

int rowCount(const QModelIndex &parent) const;

QVariant data(const QModelIndex &index, int role) const;

QHash<int, QByteArray> roleNames() const;

Q\_INVOKABLE void loadMore(bool forward);

int pageSize();

void setPageSize(int size);

int total();

void setTotal(int total);

signals:

void pageSizeChanged(int size);

void totalChanged(int total);

private:

DynamicListModelPrivate \*m\_dptr;

};

#endif // DYNAMICMODEL\_H

    dynamicModel.cpp：

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/foruok/article/details/39052729)

1. #include "dynamicModel.h"
2. #include <QDebug>
4. class DynamicListModelPrivate
5. {
6. public:
7. DynamicListModelPrivate(DynamicListModel \*model)
8. : m\_model(model), m\_start(0), m\_end(20)
9. , m\_total(100), m\_pageSize(20)
10. {
11. m\_roleNames.insert(Qt::UserRole, "content");
12. }
14. void pageDown()
15. {
16. if(m\_end < m\_total)
17. {
18. m\_start += m\_pageSize;
19. m\_end += m\_pageSize;
20. if(m\_end > m\_total)
21. {
22. m\_end = m\_total;
23. m\_start = m\_end - m\_pageSize;
24. }
25. }
26. }
28. void pageUp()
29. {
30. if(m\_start > 0)
31. {
32. m\_start -= m\_pageSize;
33. if(m\_start < 0) m\_start = 0;
34. m\_end = m\_start + m\_pageSize;
35. }
36. }
38. void adjustPageRange()
39. {
40. if(m\_end - m\_start < m\_pageSize)
41. {
42. m\_end = m\_start + m\_pageSize;
43. if(m\_end > m\_total)
44. {
45. m\_end = m\_total;
46. m\_start = m\_end - m\_pageSize;
47. }
48. }
49. }
51. DynamicListModel \*m\_model;
52. **int** m\_start;
53. **int** m\_end;
54. **int** m\_total;
55. **int** m\_pageSize;
56. QHash<**int**, QByteArray> m\_roleNames;
57. };
59. DynamicListModel::DynamicListModel(QObject \*parent)
60. : QAbstractListModel(parent),
61. m\_dptr(new DynamicListModelPrivate(this))
62. {
64. }
66. DynamicListModel::~DynamicListModel()
67. {
68. delete m\_dptr;
69. }
71. **int** DynamicListModel::rowCount(const QModelIndex &parent) const
72. {
73. return m\_dptr->m\_end - m\_dptr->m\_start;
74. }
76. QVariant DynamicListModel::data(const QModelIndex &index, **int** role) const
77. {
78. **int** row = index.row();
79. //qDebug() << "index.row - " << row << " start - " << m\_dptr->m\_start;
80. return QString::number(row + m\_dptr->m\_start);
81. }
83. QHash<**int**, QByteArray> DynamicListModel::roleNames() const
84. {
85. return m\_dptr->m\_roleNames;
86. }
88. void DynamicListModel::loadMore(**bool** forward)
89. {
90. beginResetModel();
91. if(forward)m\_dptr->pageDown();
92. else m\_dptr->pageUp();
93. endResetModel();
94. }
96. **int** DynamicListModel::pageSize()
97. {
98. return m\_dptr->m\_pageSize;
99. }
101. void DynamicListModel::setPageSize(**int** size)
102. {
103. m\_dptr->m\_pageSize = size;
104. m\_dptr->adjustPageRange();
105. emit pageSizeChanged(size);
106. }
108. **int** DynamicListModel::total()
109. {
110. return m\_dptr->m\_total;
111. }
113. void DynamicListModel::setTotal(**int** total)
114. {
115. m\_dptr->m\_total = total;
116. m\_dptr->adjustPageRange();
117. emit totalChanged(total);
118. }

#include "dynamicModel.h"

#include <QDebug>

class DynamicListModelPrivate

{

public:

DynamicListModelPrivate(DynamicListModel \*model)

: m\_model(model), m\_start(0), m\_end(20)

, m\_total(100), m\_pageSize(20)

{

m\_roleNames.insert(Qt::UserRole, "content");

}

void pageDown()

{

if(m\_end < m\_total)

{

m\_start += m\_pageSize;

m\_end += m\_pageSize;

if(m\_end > m\_total)

{

m\_end = m\_total;

m\_start = m\_end - m\_pageSize;

}

}

}

void pageUp()

{

if(m\_start > 0)

{

m\_start -= m\_pageSize;

if(m\_start < 0) m\_start = 0;

m\_end = m\_start + m\_pageSize;

}

}

void adjustPageRange()

{

if(m\_end - m\_start < m\_pageSize)

{

m\_end = m\_start + m\_pageSize;

if(m\_end > m\_total)

{

m\_end = m\_total;

m\_start = m\_end - m\_pageSize;

}

}

}

DynamicListModel \*m\_model;

int m\_start;

int m\_end;

int m\_total;

int m\_pageSize;

QHash<int, QByteArray> m\_roleNames;

};

DynamicListModel::DynamicListModel(QObject \*parent)

: QAbstractListModel(parent),

m\_dptr(new DynamicListModelPrivate(this))

{

}

DynamicListModel::~DynamicListModel()

{

delete m\_dptr;

}

int DynamicListModel::rowCount(const QModelIndex &parent) const

{

return m\_dptr->m\_end - m\_dptr->m\_start;

}

QVariant DynamicListModel::data(const QModelIndex &index, int role) const

{

int row = index.row();

//qDebug() << "index.row - " << row << " start - " << m\_dptr->m\_start;

return QString::number(row + m\_dptr->m\_start);

}

QHash<int, QByteArray> DynamicListModel::roleNames() const

{

return m\_dptr->m\_roleNames;

}

void DynamicListModel::loadMore(bool forward)

{

beginResetModel();

if(forward)m\_dptr->pageDown();

else m\_dptr->pageUp();

endResetModel();

}

int DynamicListModel::pageSize()

{

return m\_dptr->m\_pageSize;

}

void DynamicListModel::setPageSize(int size)

{

m\_dptr->m\_pageSize = size;

m\_dptr->adjustPageRange();

emit pageSizeChanged(size);

}

int DynamicListModel::total()

{

return m\_dptr->m\_total;

}

void DynamicListModel::setTotal(int total)

{

m\_dptr->m\_total = total;

m\_dptr->adjustPageRange();

emit totalChanged(total);

}

    DynamicListModel仅仅是演示用法，使用m\_start、m\_end、m\_total、m\_pageSize四个整型变量来模拟实际的数据。而data()方法，将ListView内的行序号加上m\_start转换为字符串返回，就是我们在ListView界面上看到了文字了。

    loadMore()函数，区分向前还是向后加载数据，它调用DynamicListModel的pageDown()、pageUp()来更新内部的数据状态。在loadMore()一开始，调用beginResetModel()，通知关联到DynamicListModel上的view们刷新自己，当内部数据状态更新结束后，调用endResetModel()来通知view们，这样view们就会刷新，最终在实例化item delegate时调用data()方法来准备数据，此时m\_start已变化，所以界面上看到的数字也跟着变了。

**导出C++ Model**

    这个简单，我们在《[Qt Quick 之 QML 与 C++ 混合编程详解](http://blog.csdn.net/foruok/article/details/32698603" \t "_blank)》一文中已经讲过。直接看main.cpp：

**[cpp]** [view plaincopyprint?](http://blog.csdn.net/foruok/article/details/39052729)

1. #include <QGuiApplication>
2. #include <QQmlApplicationEngine>
3. #include <QQmlContext>
4. #include "dynamicModel.h"
6. **int** main(**int** argc, **char** \*argv[])
7. {
8. QGuiApplication app(argc, argv);
10. QQmlApplicationEngine engine;
11. QQmlContext \*ctx = engine.rootContext();
12. ctx->setContextProperty("dynamicModel", new DynamicListModel());
13. engine.load(QUrl(QStringLiteral("qrc:///main.qml")));
15. return app.exec();
16. }

#include <QGuiApplication>

#include <QQmlApplicationEngine>

#include <QQmlContext>

#include "dynamicModel.h"

int main(int argc, char \*argv[])

{

QGuiApplication app(argc, argv);

QQmlApplicationEngine engine;

QQmlContext \*ctx = engine.rootContext();

ctx->setContextProperty("dynamicModel", new DynamicListModel());

engine.load(QUrl(QStringLiteral("qrc:///main.qml")));

return app.exec();

}

**QML代码介绍**

    是时候看看main.qml了：

**[javascript]** [view plaincopyprint?](http://blog.csdn.net/foruok/article/details/39052729)

1. import QtQuick 2.2
2. import QtQuick.Window 2.1
3. import QtQuick.Controls 1.2
4. import QtQuick.Layouts 1.1
5. import QtQuick.Controls.Styles 1.2
7. Window {
8. width: 320;
9. height: 480;
10. minimumWidth: 300;
11. minimumHeight: 480;
12. visible: true;
13. id: root;
15. Component {
16. id: listDelegate;
17. Text {
18. id: wrapper;
19. width: parent.width;
20. height: 32;
21. font.pointSize: 15;
22. verticalAlignment: Text.AlignVCenter;
23. horizontalAlignment: Text.AlignHCenter;
24. text: content;
25. color: ListView.view.currentIndex == index ? "red" : "blue";
26. MouseArea {
27. anchors.fill: parent;
28. onClicked: {
29. if(wrapper.ListView.view.currentIndex != index){
30. wrapper.ListView.view.currentIndex = index;
31. }
32. }
33. }
34. }
35. }
37. ListView {
38. id: dynamicList;
39. z: 1;
40. anchors.fill: parent;
41. anchors.margins: 10;
43. delegate: listDelegate;
44. model: dynamicModel;
46. focus: true;
47. activeFocusOnTab: true;
48. highlight: Rectangle {
49. color: "steelblue";
50. }
52. property real contentYOnFlickStarted: 0;
53. onFlickStarted: {
54. //console.log("start,origY - ", originY, " contentY - ", contentY);
55. contentYOnFlickStarted = contentY;
56. }
58. onFlickEnded: {
59. //console.log("end,origY - ", originY, " contentY - ", contentY);
60. dynamicModel.loadMore(contentY < contentYOnFlickStarted);
61. }
62. }
63. }

import QtQuick 2.2

import QtQuick.Window 2.1

import QtQuick.Controls 1.2

import QtQuick.Layouts 1.1

import QtQuick.Controls.Styles 1.2

Window {

width: 320;

height: 480;

minimumWidth: 300;

minimumHeight: 480;

visible: true;

id: root;

Component {

id: listDelegate;

Text {

id: wrapper;

width: parent.width;

height: 32;

font.pointSize: 15;

verticalAlignment: Text.AlignVCenter;

horizontalAlignment: Text.AlignHCenter;

text: content;

color: ListView.view.currentIndex == index ? "red" : "blue";

MouseArea {

anchors.fill: parent;

onClicked: {

if(wrapper.ListView.view.currentIndex != index){

wrapper.ListView.view.currentIndex = index;

}

}

}

}

}

ListView {

id: dynamicList;

z: 1;

anchors.fill: parent;

anchors.margins: 10;

delegate: listDelegate;

model: dynamicModel;

focus: true;

activeFocusOnTab: true;

highlight: Rectangle {

color: "steelblue";

}

property real contentYOnFlickStarted: 0;

onFlickStarted: {

//console.log("start,origY - ", originY, " contentY - ", contentY);

contentYOnFlickStarted = contentY;

}

onFlickEnded: {

//console.log("end,origY - ", originY, " contentY - ", contentY);

dynamicModel.loadMore(contentY < contentYOnFlickStarted);

}

}

}

    定义ListView对象时，指定其model为main()函数中导出的dynamicModel，其它的代码不必细说了，咱们单看实现下拉（上拉）刷新的关键代码。

    onFlickStarted信号处理器，在这里我们仅仅是将flick开始时的contentY记录到contentYOnFlickStarted属性中。

    onFlickEnded信号处理器，这里比较flick结束时的contentY和开始时的contentY（即contentYOnFlickStarted），结束时小，说明是下拉，结束时大，说明是上拉。根据比较结果调用loadMore()。

    好啦，就这么简单了。看看效果。

**下拉刷新效果**

    图1是初始效果：

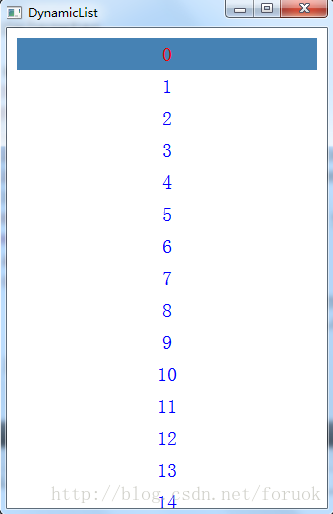


             图1动态刷新列表初始效果

    图2是下拉了两次后的效果：

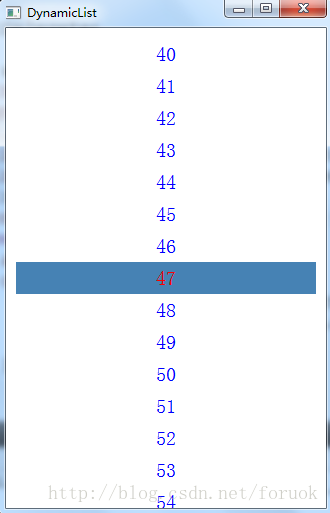


             图2 下拉刷新后的效果

    图3是从图2所示状态上拉后的效果：

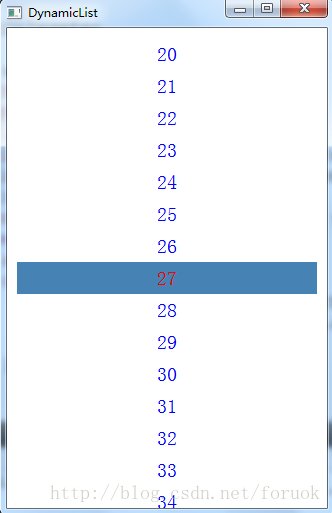


             图3 上拉后的效果

    版权所有foruok，转载请注明出处：<http://blog.csdn.net/foruok>。

    好啦，解说完毕。