如何 通过QGalleryQueryModel 制作一个简单的图片浏览器

介绍

QML 提供了一个 plugin DocumentGalleryMode ，通过使用这个plugin,我们可以很简单的实现一个图片浏览器，但是随着图片的增加，我们会发现加载速度十分缓慢，用户体验很差，这是由于QML 内部实现采取高质量的平滑缩放耗时很长，所以我们不的不重新实现自己的model ,并在线程中通过QImageReader 实现大量图片的快速缩放。接下来我们通过代码片段来看下具体的实现过程

代码实现

ThumbnailModel.h

```
class ThumbnailModel : public QGalleryQueryModel
{
    Q_OBJECT
public:
    static const QSize thumbnailSize;

    ThumbnailModel(QAbstractGallery *gallery, QObject *parent = 0);
    ~ThumbnailModel();

    QVariant data(const QModelIndex &index, int role = Qt::DisplayRole) const;

protected:
    virtual QUrl imageUrl(const QModelIndex &index) const;

private slots:
    void thumbnailLoaded();

private:
    mutable ThumbnailCache cache;

};
<code>
首先我们必须从 QGalleryQueryModel 派生一个类, 并且重写data()方法
===ThumbnailModel.cpp===
<code>
QVariant ThumbnailModel::data(const QModelIndex &index, int role) const
{
    qDebug()<<"enter data";
    return role == Qt::DecorationRole && index.isValid()
            ? cache.thumbnail(imageUrl(index))
            : QGalleryQueryModel::data(index, role);
}
```

```
void ThumbnailModel::thumbnailLoaded()
{
    emit dataChanged(index(0, 0), index(rowCount() - 1, columnCount() -1));
}
```

在data 函数中我们返回已经缩放好的图片， 以便VIEW绘制。在thumbnailLoaded()函数中我们需要发送一个信号通知VIEW MODEL中的数据有所更新，以便VIEW进行更新。

thumbnailcache.h

```
class ThumbnailCache : public QThread
{
    Q_OBJECT
public:
    ThumbnailCache(QObject *parent = 0);
    ~ThumbnailCache();

    QString thumbnail(const QUrl &url);

    bool event(QEvent *event);

signals:
    void thumbnailReady();

protected:
    void run();

private:
    QImage loadImage(const QUrl &url) const;
    QString thumbnailPath(const QUrl &url) const;

    QMutex mutex;
    QWaitCondition waitCondition;
    QCache<QUrl, Thumbnail> cache;
    QQueue<QUrl> pendingUrls;
    bool cancelled;
};
```

我们需要定义一个线程类，在run函数中实现图片的缩放

(c) Copyright Nokia 2013. All rights reserved.

thumbnailcache.cpp

```
QImage ThumbnailCache::loadImage(const QUrl &url) const
{
    const QString fileName = thumbnailPath(url);

    QImageReader reader(fileName);
    reader.setQuality(25);

    if (reader.supportsOption(QImageIOHandler::Size)) {
        QSize size = reader.size();

        if (!reader.supportsOption(QImageIOHandler::ScaledSize)
                && (size.width() > 1280 || size.height() > 1280)) {
            return QImage();
        }

        if (size.width() > ThumbnailModel::thumbnailSize.width()
                || size.height() > ThumbnailModel::thumbnailSize.height()) {
            size.scale(ThumbnailModel::thumbnailSize, Qt::KeepAspectRatio);
        }

        reader.setScaledSize(size);
    } else {
        reader.setScaledSize(ThumbnailModel::thumbnailSize);
    }

    return reader.read();
}
```

我们将在loadImage（）函数中通过QImageReader 对图片进行快速缩放

photodelegate.h

```
#ifndef PHOTODELEGATE_H
#define PHOTODELEGATE_H

#include <QtGui/QAbstractItemDelegate>
#include <qDebug>
class PhotoDelegate : public QAbstractItemDelegate
{
    Q_OBJECT
public:
    PhotoDelegate(QObject *parent = 0);
    ~PhotoDelegate();

    void paint(
            QPainter *painter, const QStyleOptionViewItem &option, const QModelIndex &index) const;

    QSize sizeHint(const QStyleOptionViewItem &option, const QModelIndex &index) const;
};

#endif
```

我们将定义一个类派生自 QAbstractItemDelegate，用于进行图片的绘制

photodelegate.cpp

```
void PhotoDelegate::paint(
        QPainter *painter, const QStyleOptionViewItem &option, const QModelIndex &index) const
{
    qDebug()<<"enter paint";

    QPen oldPen = painter->pen();

    QStyle *style = QApplication::style();

    int margin = style->pixelMetric(QStyle::PM_ButtonMargin);

    QRect rect = option.rect;

    if (option.state & QStyle::State_HasFocus) {
        painter->fillRect(rect, option.palette.highlight());

        painter->setPen(option.palette.color(QPalette::HighlightedText));
    }

    rect.adjust(margin, margin, -margin, -margin);

    QRect decorationRect = rect;
    decorationRect.setRight(decorationRect.left() + option.decorationSize.width());

    QPixmap decoration = qvariant_cast<QPixmap>(index.data(Qt::DecorationRole));
    if (!decoration.isNull())
        style->drawItemPixmap(painter, decorationRect, Qt::AlignCenter, decoration);
    else
        painter->drawRect(decorationRect);

    painter->setPen(oldPen);
}

QSize PhotoDelegate::sizeHint(const QStyleOptionViewItem &option, const QModelIndex &) const
{
    int margin = QApplication::style()->pixelMetric(QStyle::PM_ButtonMargin);

    QSize size = option.decorationSize;

    size.rheight() += 2 * margin;
    size.rwidth() += 2 * margin;

    return size;
}
```

我们在Paint()函数中绘制从MODEL中获得的数据，这样我们就完成了一个简单图片浏览器

(C) Copyright Nokia 2013. All rights reserved.

代码下载

File:Mediabrowser.zip

相关链接

Add categories below. Remove Category:Draft when the page is complete or near complete

© Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.