Using QDeclarativeImageProvider for speeding up Symbian QML app installation

This code example shows how you can extract image files stored in a zip archive and display them in QML, using QDeclarativeImageProvider and QuaZip . The technique can be used to significantly reduce the installation time of Symbian applications that have many files, and is particularly useful where it is not possible to store the files in the application's resources.

25 Dec 2011

Introduction

Apps that reference a lot of files can take an unacceptable amount of time to install. Since the time taken to install a Symbian application is more dependent on the *number of files* than the total file size, these apps may benefit from reducing the number of files in the `sis` package.

As an example of the benefits, consider a 15 Mb Symbian installation file containing more than 500 images. If the files are referenced separately it takes about 8 minutes to install to the embedded memory card of a Nokia N8-00 (and 3.5 minutes to uninstall). If the same 500 image files are instead first packaged in two zip files, the installation time reduces to 30 seconds!

Where the total size of the images is relatively small, you can use the Qt Resource system to embed all the images in the application executable. Unfortunately this is memory inefficient because all the files must be loaded into process memory on application-start; if there are too many files your application may well run out of memory.

If the total size of images is large, the solution is to put them into a zip and to load them when needed. To do this we implement a `QDeclarativeImageProvider`-derived class, which uses QuaZip to read the images from the zip, to make them available to QML. The following section shows the most important elements of the solution, and a full sample project is attached.

Solution

From Qt 4.7 there is a useful class called QDeclarativeImageProvider that provides an interface for supporting pixmaps and threaded image requests in QML. To provide images to QML we derive from `QDeclarativeImageProvider`, create an object from the derived class and attach it to QML engine with certain key (QString) value. Every time QML engine tries to resolve image an image source (`image://you_key/somefile_id.jpg`) it will invoke the object's method with `somefile_id.jpg` as parameter and will expect a QImage or QPixmap in return.

Declaring our `ImageReader` class:

```
#include <QDeclarativeImageProvider>
#include "quazip/quazip.h"
class QZIPDeclarativeImageProvider : public QDeclarativeImageProvider
{
public:
    QZIPDeclarativeImageProvider(const QString &filenameZIP);
    ~QZIPDeclarativeImageProvider();
    QImage requestImage(const QString &id, QSize *size, const QSize& requestedSize);
private:
 QuaZip* m_zip;
};
```

Its implementation:

```
#include "quazip/quazipfile.h"
#include <QDir.h>
QZIPDeclarativeImageProvider::QZIPDeclarativeImageProvider(const QString &filenameZIP):
QDeclarativeImageProvider(QDeclarativeImageProvider::Image)
{
    m_zip =  new QuaZip(QDir::currentPath().append("/").append(filenameZIP));
    if (m_zip) m_zip->open(QuaZip::mdUnzip);
```


(C) Copyright Nokia 2014. All rights reserved.

```
}

QZIPDeclarativeImageProvider::~QZIPDeclarativeImageProvider()
{
    if (m_zip)
    {
        m_zip->close();
        delete m_zip;
    }
}

QImage QZIPDeclarativeImageProvider::requestImage(const QString &id, QSize *size, const
QSize& requestedSize)
{
if (m_zip && m_zip->isOpen())
if ( m_zip->setCurrentFile(id, QuaZip::csInsensitive) )
{
    QuaZipFile file(m_zip);
    file.open(QIODevice::ReadOnly);
    QByteArray data = file.read(file.csize());
    QImage img = QImage::fromData(data);
    file.close();
    return img;
}
return QImage();
}
```

To make our image provider available to QML it needs to be attached to the QML declarative engine (before opening any qml files) as shown below:

```
QZIPDeclarativeImageProvider zipreader (QString("images.zip"));
viewer->engine()->addImageProvider(QString("myzip"), &zipreader);
```

Here `images.zip` is our data file and `myzip` is the keyword used used to reference *this* `QDeclarativeImageProvider` from QML.

Now we can use any image from that zip in our qml file as shown below:

```
Image {
        id: image1
        x: 58
        y: 51
        width: 100
        height: 100
        source: "image://myzip/01_sdk_download_v4.jpg"
    }
```

File `01_sdk_download_v4.jpg` will be retrieved from the `images.zip` when necessary.

References

- Using QDeclarativeImageProvider for speeding up Symbian QML app installation (Alexander Trufanov's Blog)

(C) Copyright Nokia 2014. All rights reserved.

(C) Copyright Nokia 2014. All rights reserved.