Network caching in Qt

Overview

This article demonstrates how to use network caching in Qt.

Description

The `QNetworkDiskCache` class provides a very basic disk cache. It can be used with `QNetworkAccessManager` to store the requested web pages into a specific directory and get the cached content whenever required. Each cached file has a **cache_** prefix and **.cache** extension in the file name.

Network Caching

Required headers

```
#include <QNetworkAccessManager>
#include <QUrl>
#include <QNetworkRequest>
#include <QNetworkReply>
#include <QNetworkDiskCache>
```

Required modules

```
QT += network
```

`NetWorkCacheTest` is our test class, derived from `QWidget`:

```
 class NetWorkCacheTest : public QWidget
  {
  Q_OBJECT

  NetWorkCacheTest( QWidget *parent = 0 );
  // ...

public:
  void sendRequest();

public slots:
  // Slot to receive finished signal from QNetworkAccessManager
  void finishedSlot( QNetworkReply* aReply );

private:

  QNetworkAccessManager*  m_nam;
  QNetworkDiskCache*      m_diskCache;
  QUrl m_url;
  };
```

Constructor implementation

```
NetWorkCacheTest::NetWorkCacheTest( QWidget *parent )
    : QWidget( parent )
  {
  ...
  m_url.setUrl( <nowiki>"http://www.developer.nokia.com/"</nowiki> );
  m_nam = new QNetworkAccessManager( this );

  // Connect the finished signal to receive network replies
  connect( m_nam, SIGNAL( finished( QNetworkReply* ) ), this, SLOT( finishedSlot( QNetworkReply* ) ) );

  // Create an instance of disk cache and set cache directory
  m_diskCache = new QNetworkDiskCache( this );
  QString sysCacheDir( "c:\\data" );
  m_diskCache->setCacheDirectory( sysCacheDir );

  // Assign cache to network manager and send a request
  m_nam->setCache( m_diskCache );
  sendRequest();
  };
```

sendRequest() implementation

```
void NetWorkCacheTest::sendRequest()
  {
  QNetworkRequest request( m_url );

  // Load from cache if available, otherwise load from network, may return stale data
  request.setAttribute( QNetworkRequest::CacheLoadControlAttribute, QNetworkRequest::PreferCache );

  // Data obtained should be saved to cache for future uses
  request.setAttribute( QNetworkRequest::CacheSaveControlAttribute, true );
```


(C) Copyright Nokia 2013. All rights reserved.

```
    // Posts a request to obtain the contents of the target request,
    // finished signal will get triggered when network reply is available
    m_nam->get( QNetworkRequest( request ) );
    }
```

Handling the network reply

```
void NetWorkCacheEx::finishedSlot( QNetworkReply* reply )
  {

  // Get and display the last modified header
  QVariant lmod = reply->header( QNetworkRequest::LastModifiedHeader );
  if ( lmod.isValid() ) {
    QDateTime lmodDate = lmod.toDateTime();
    QString lmodStr = lmodDate.toString( "dd.MM.yyyy" );
    QMessageBox::information( this, "Last Modified", lmodStr, "ok" );
    }

  // Show expiry date
  if ( reply->hasRawHeader( "Expires" ) ) {
    QString expires = reply->rawHeader( "Expires" );
    QMessageBox::information( this, "Expiry date", expires, "ok" );
    }

  // Show Cache-Control header
  if ( reply->hasRawHeader( "Cache-Control" ) ) {
    QString cCntrl = reply->rawHeader( "Cache-Control" );
    QMessageBox::information( this, "Cache-Control", cCntrl, "ok" );
    }

  // Check if the data was obtained from cache or not
  QVariant fromCache = reply->attribute( QNetworkRequest::SourceIsFromCacheAttribute );
  if ( fromCache.toBool() ) {
    // Data is from cache
    }
  else {
    // Data is from network
    }

  if ( reply->error() == QNetworkReply::NoError ) {
    // Display received data in web view (needs webkit module added)
    // QWebView view;
    // Reading bytes form the reply
    // QByteArray bytes = reply->readAll();
    // view.setHtml(bytes);
    // view.show();
    }
  else { // Error handling
    int error = reply->error();
    QMessageBox::information( this, "Error", reply->errorString(), "ok" );
    }
  }
```


(C) Copyright Nokia 2013. All rights reserved.

(C) Copyright Nokia 2013. All rights reserved.