BlackBerry 10 Jam

Developing NFC Applications for BlackBerry 10

John Murray and Martin Woolley

DEV120

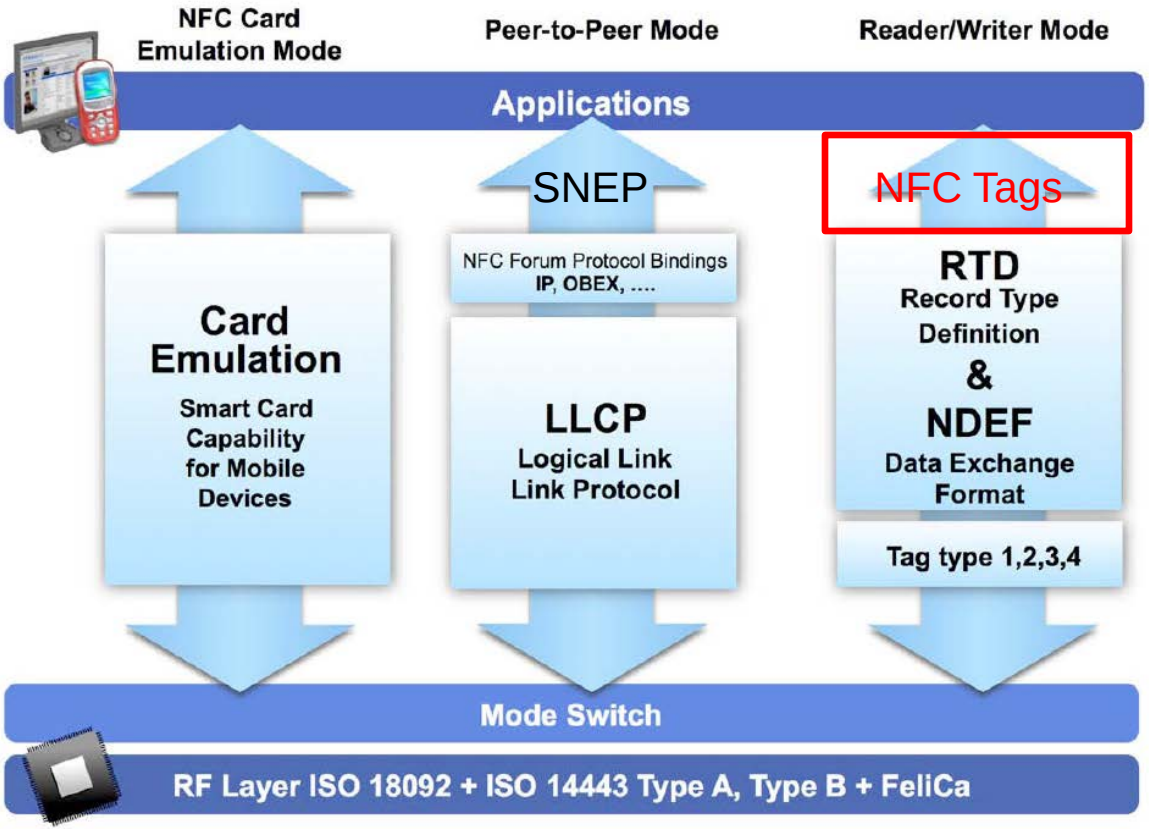
May 1-3, 2012

Session Overview

BlackBerry 10 Jam

- Introductions
- Brief introduction to NFC
- Developing for the BlackBerry® 10 platform
- How to read NFC Tags
- How to write to NFC tags
- Close
- Q&A

NFC Forum Architecture



- We're going to talk about:
  - Reading NFC Tags
  - Writing NFC tags
- And general BlackBerry 10 development using the NDK

BlackBerry 10 Jam

Developing for BlackBerry 10

Martin Woolley

Developing for BlackBerry 10

- NFC can be exploited on BlackBerry today

Java	WebWorks
Tag reading and writing	Tag reading using open source extension
Peer to peer including LLCP and SNEP	
Card Emulation using a choice of 2 Secure Elements	

- How many of you have used these APIs?

Developing for BlackBerry 10

BlackBerry 10 Jam

- Here's what we're going to talk about:
  - Developing NFC applications for the BlackBerry 10 platform
  - The BlackBerry NDK with Cascades and Qt
  - The BlackBerry Platform Services (BPS)

- Our discussion will be based around a real application "Tag Tool"

BlackBerry 10 Jam

Reading NFC Tags

Martin Woolley

About NFC Tags

BlackBerry 10 Jam

- A card that stores data
- They come in many forms
- The NFC Forum define 4 types
- Simple and standardised behaviour
- Data is stored in the **NDEF** format
- It’s the standardisation that makes tags so useful

About NFC Tags

- Bus stops in Caen, France use NFC tags
- They provide easy access to bus arrival times
  - Browser opens at the transport company's web site and the next bus time is displayed
  - No special software from the bus company required
  - No extra infrastructure required at the bus stop

Developing your own tag reader

BlackBerry 10 Jam

- Initialise the BlackBerry Platform Services (BPS) library
- Request NFC events from the BPS
- Register for specific NDEF message types
- Process NFC event objects in an event loop

Initialise BPS

This step requires a single function call to a standard BPS library function:

```
#include <bps/bps.h>


int rc = bps_initialize();
```

Request NFC events from BPS

BlackBerry 10 Jam

This step also requires a single function call only:

```
#include <nfc.h>

rc = nfc_request_events();
```

Specific NDEF message types

BlackBerry 10 Jam

Here we register for three NDEF RTD types:

- **“U” (URI),**
- **“T” (Text) and**
- **“Sp” (Smart Poster).**

```
nfc_register_ndef_reader(NDEF_TNF_WELL_KNOWN, "U”);

nfc_register_ndef_reader(NDEF_TNF_WELL_KNOWN, "T”);

nfc_register_ndef_reader(NDEF_TNF_WELL_KNOWN, "Sp”);
```

The Event Loop

This code should execute in a separate thread from the main UI thread since it involves blocking calls.

```
while (notInterrupted()) {
  bps_event_t *event; int domain; int rc;
  // Blocking call to wait for events with a timeout.
  rc = bps_get_event(&event, BPS_EVENT_TIMEOUT); // BLOCKS
  if (!rc) {
    if (event) {
        handleNfcEvent(event);
.....
```

Handling an NFC Event Object (1)

BlackBerry 10 Jam

```
void NfcManager::handleNfcEvent(bps_event_t *event) {

    // get ndef target. Required to read ndef data.
    nfc_target_t* target;
    nfc_get_target(event, &target)

    // count the number of messages available
    int msg_count = 0;
    nfc_get_ndef_message_count(target, &msg_count);
```

Handling an NFC Event Object (2)

BlackBerry 10 Jam

```
// We only want the first NDEF message for the moment
if(msg_count > 0) {
    nfc_ndef_message_t *message;
    nfc_get_ndef_message(target, 0, &message);

// Check the number of NDEF records in this message
    int rec_count = 0;
    nfc_get_ndef_record_count(message, &rec_count);
```

Handling an NFC Event Object (3)

BlackBerry 10 Jam

```
// We only want the first NDEF record for the moment
if(rec_count > 0) {

// Get the first NDEF message record
   nfc_ndef_record_t *record;
   nfc_get_ndef_record(message, 0, &record);

// Get the record payload
   uchar_t* data;
   int len;
   nfc_get_ndef_record_payload(record, &data, &len);
```

Handling an NFC Event Object (4)

BlackBerry 10 Jam

```
// Obtain other attributes from the record
tnf_type_t *tnf;
nfc_get_ndef_record_tnf(record, &tnf);

char *type;
nfc_get_ndef_record_type(record, &type);

char *uri;
nfc_get_sp_uri(record, &uri)
```

Recap (1)

BlackBerry 10 Jam

```
// initialisation

int rc = bps_initialize();

rc = nfc_request_events();

nfc_register_ndef_reader(NDEF_TNF_WELL_KNOWN, "U");
```

Recap (2)

BlackBerry 10 Jam

```
// in the event loop

rc = bps_get_event(&event, BPS_EVENT_TIMEOUT); // BLOCKS
nfc_get_target(event, &target)
nfc_get_ndef_message(target, 0, &message);
nfc_get_ndef_record(message, 0, &record);

nfc_get_ndef_record_payload(record, &data, &len);
nfc_get_ndef_record_tnf(record, &tnf);
nfc_get_ndef_record_type(record, &type);
nfc_get_sp_uri(record, &uri)
```

BlackBerry 10 Jam

Sample Application Review #1

Martin Woolley

BlackBerry 10 Development

BlackBerry 10 Jam

- There's more to developing an NFC application for the BlackBerry 10 platform than simply calling the NFC APIs
- Imagine what we could do with the NFC "Bus Stop" example?

BlackBerry 10 Development

BlackBerry 10 Jam

Flow

- User Experience
- Simplify consumption of real-time bus arrival data by simply tapping a poster

Share

- Imagine it’s a child taking the bus
- The NFC tag could trigger your app to share the child’s location/timestamp with the family BBM group

Extend

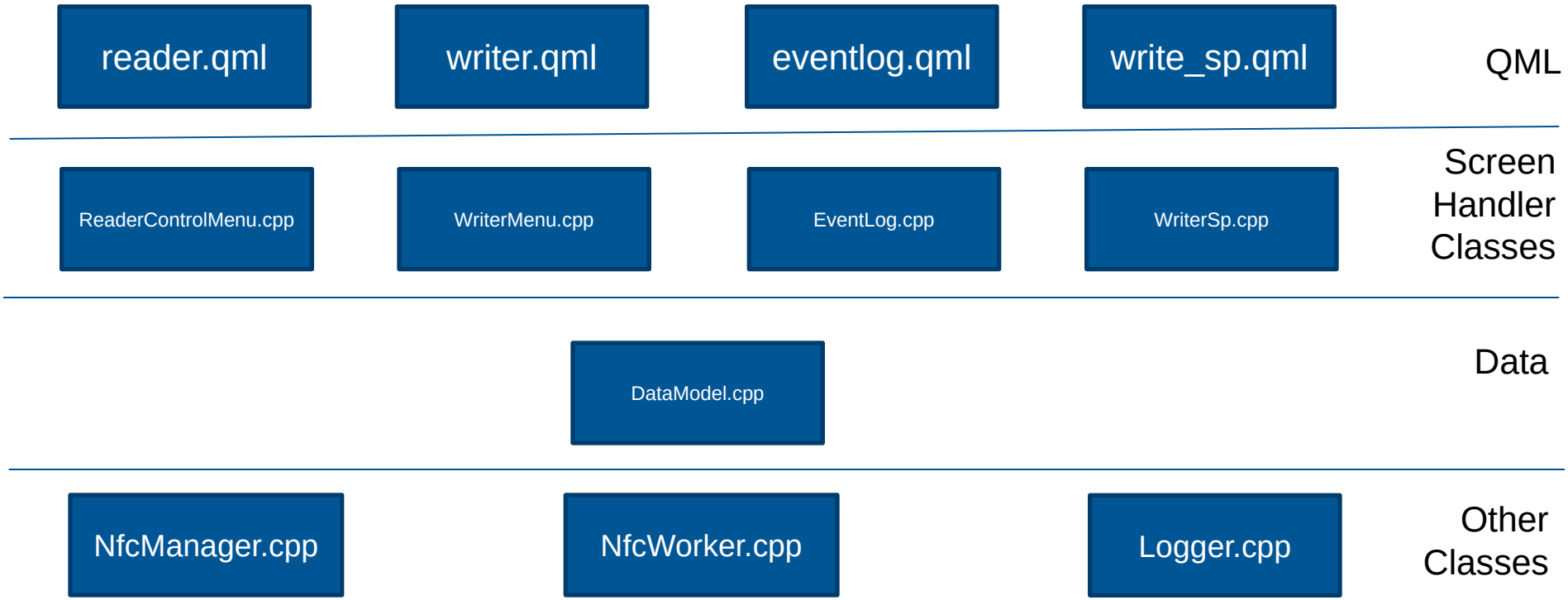
- Calculate expected time of arrival at destination and notify parent via BBM family group

BlackBerry 10 Development

BlackBerry 10 Jam

- John and Martin are developing a sample application called “Tag Tool” which will be released as open source via GitHub
- Let’s look at some other aspects of Tag Tool briefly

Tag Tool Architecture



- Let’s consider the following challenges:

1. How to create the user interface
2. How to integrate the UI with C++ components

User Interface Development

BlackBerry 10 Jam

- For the UI we used Cascades
- This is super-easy and involves writing QML, a declarative language for user interfaces, with JavaScript and attribute bindings
- QML is what gives your app its flow

```
animations: [
                SequentialAnimation {
                    id: "animation"
                    animations: [
                        FadeTransition {
                            duration: 1000
                            fromOpacity: 0.0
                            toOpacity: 1.0
                        }
.......
```

UI / C++ Integration

BlackBerry 10 Jam

- C++ objects can be stored in the context of a QML document and its properties exposed for use from JavaScript in the QML page
- Exposing properties involves use of the Qt Q_PROPERTY macro
- Let’s look at an example involving the version number of our sample application

UI / C++ Integration

BlackBerry 10 Jam

```
// MainMenu.hpp

class MainMenu : public QObject
{
  Q_OBJECT
  Q_PROPERTY(QString appVersion READ appVersion WRITE setAppVersion NOTIFY detectAppVersionChanged)

public:
    MainMenu();
    virtual ~MainMenu();

    QString appVersion() const;
    void setAppVersion(QString appVersion);

private:
    QString _appVersion;
    QmlDocument *_qml;

signals:
    void detectAppVersionChanged();

};
```

UI / C++ Integration

BlackBerry 10 Jam

```
// MainMenu.cpp

MainMenu::MainMenu() : _appVersion(QString(Settings::AppVersion)) {
      _qml = QmlDocument::create("main.qml");
      _qml->setContextProperty("_mainMenu", this);
}

// maps to READ function in Q_PROPERTY
QString MainMenu::appVersion() const {
      return _appVersion;
}

// maps to WRITE function in Q_PROPERTY
void MainMenu::setAppVersion(QString appVersion) {
      _appVersion = appVersion;

// “tell” QML that the value has changed by emitting a signal
      emit detectAppVersionChanged();
}
```

UI / C++ Integration

BlackBerry 10 Jam

```
// main.qml

Label {
            layoutProperties: DockLayoutProperties {
                verticalAlignment: VerticalAlignment.Top
                horizontalAlignment: HorizontalAlignment.Center
            }

// note use of object name as registered in call to setContextProperty in MainMenu.cpp
//    _qml->setContextProperty("_mainMenu", this);

            text: "NFC Tag Tool V" + _mainMenu.appVersion
            textStyle.base: SystemDefaults.TextStyles.BigText
      }
```

BlackBerry 10 Jam

Writing NFC Tags

John Murray

Developing your own tag writer

BlackBerry 10 Jam

- Initialise the BlackBerry Platform Services (BPS) library
- Request NFC events from the BPS
- Register for NFC NDEF target detection events
- Process NFC event objects in an event loop

Register for Tag Detection

BlackBerry 10 Jam

Here we register for NFC tag detection.

In particular we refine this request to register for **NDEF_TAG** target detection events only.

It's also possible to request notification of target types **ISO_14443_3** and **ISO_14443_4**

```
nfc_register_tag_readerwriter(NDEF_TAG);
```

Handling an NFC Event to Write(1)

BlackBerry 10 Jam

```
void NfcManager::handleNfcEvent(bps_event_t *event) {

    // Get Target to write to from the BPS event
    nfc_target_t *target;
    nfc_get_target(event, &target);

    // Construct a URI("U") type NDEF Record
    // from a uri of length uri_len (containing prefix)
    nfc_ndef_record_t* record;
    nfc_create_ndef_record(NDEF_TNF_WELL_KNOWN,
                           "U", uri, uri_len, 0, &record );
...
```

Handling an NFC Event to Write(2)

BlackBerry 10 Jam

```
...
    // Create an empty NDEF Message
    nfc_ndef_message_t* message;
    nfc_create_ndef_message(&message));

    // Add the NDEF record to the NDEF message
    nfc_add_ndef_record(message, record));

    // Write the NDEF Message to the target
    nfc_write_ndef_message_to_tag(target, message, false);
...
```

BlackBerry 10 Jam

Sample Code Review #2

John Murray

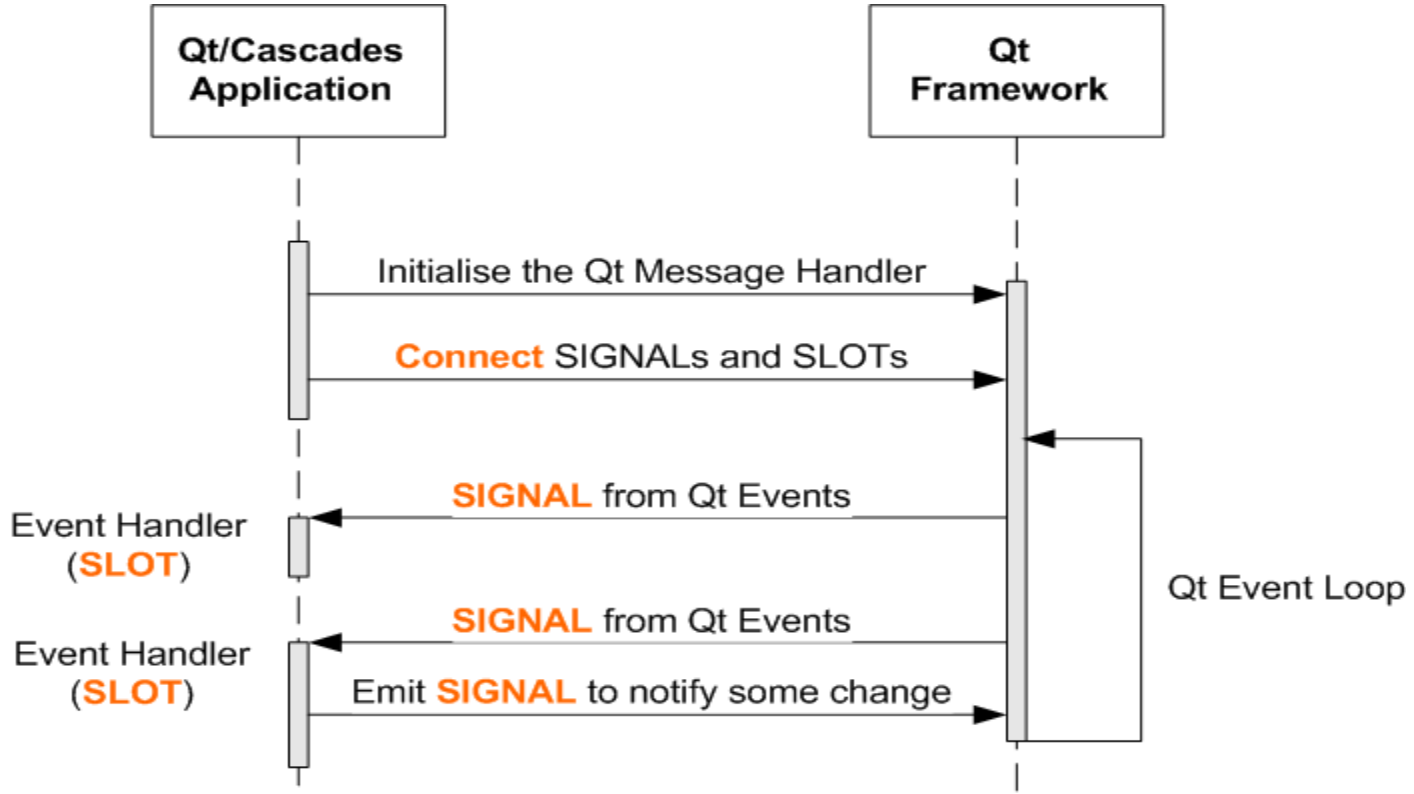
- Let's consider the following challenges:
  - **How to navigate within the UI and pass control around**
  - How to ensure your UI does not block when executing NFC operations

Navigation and Control

BlackBerry 10 Jam

- Qt includes a powerful capability known as “signals and slots”.
- Objects can emit signals
- Slots are functions which are called when a connected signal is emitted elsewhere
- Signals and slots can be linked by connecting them in a potentially many:many configuration
- If you prefer to think of things visually …

Navigation and Control



Navigation and Control

BlackBerry 10 Jam

- Example: clicking the Write Tag button must trigger the tag writing process
- Our QML contains the Write Tag button

```
Button {
        id: btn_start_write_operation
        objectName: "btn_start_write_operation"
        text: "Write Tag"
        imageSource: "images/write.png“
}
```

Navigation and Control

BlackBerry 10 Jam

- Our C++ header file defines a slot called “startWriteProcess”

```
public slots:
    void show();
    void onMainMenuTriggered();
    void onBackTriggered();
    void startWriteProcess();
    void onUriChanged(QString uri);
```

- The QML Button object has a signal called “clicked” defined as standard

Navigation and Control

BlackBerry 10 Jam

- Our C++ object connects:
  - the “clicked()” **signal** of the Button
  - to the “startWriteProcess()” **slot** which it implements

```
Button* btnStartWrite =
    _root->findChild<Button*>(
            “btn_start_write_operation”
);

QObject::connect(
          btnStartWrite, SIGNAL(clicked()),
          this, SLOT(startWriteProcess())
);
```

Navigation and Control

BlackBerry 10 Jam

- “startWriteProcess()” kicks off the NFC operation and then calls a function to cause the next screen to be displayed

```
void WriteURI::startWriteProcess() {
   NfcManager* nfc = NfcManager::getInstance();
   nfc->writeUri(&_uri);
   _eventLog->show();
}
```

Navigation and Control

BlackBerry 10 Jam

- Showing the next screen

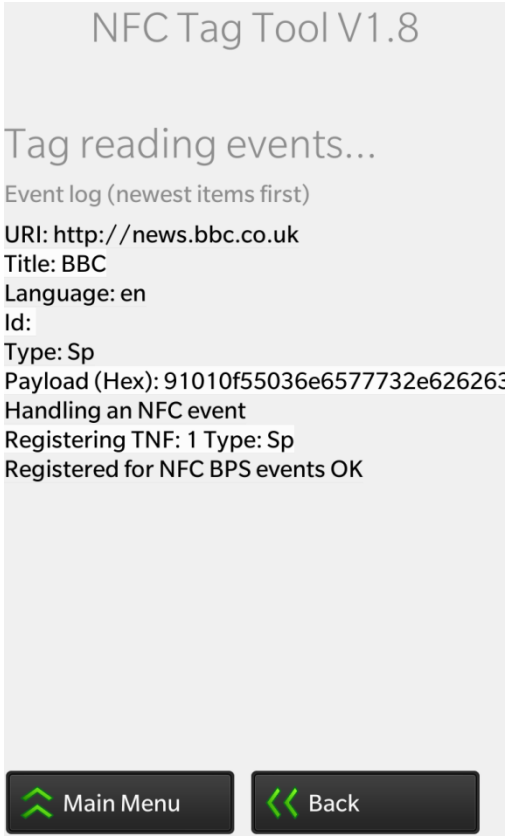
```
// elsewhere we create the QmlDocument
qml = QmlDocument::create("eventlog.qml");

// our show() function actually displays it

void EventLog::show() {
   _root = qml->createRootNode<AbstractPane>();
   Application::setScene(_root);
   ....
}
```

- Let's consider the following challenge:
  - **How to ensure your UI does not block when executing NFC operations**

Worker Threads

BlackBerry 10 Jam

- Reading events from the BPS event queue will block.
- If you perform BPS operations in the main UI thread then this will impact the user experience.
- So, we need to run NFC operations in a different thread to the main UI thread



Useful Qt Threading Techniques

BlackBerry 10 Jam

- The Qt Framework has a rich set of classes that can be used to run NFC operations in a separate thread such as:
  - Qthread, QtConcurrent
- Many of the Qt examples on the web focus on sub-classing QThread which can result in more convoluted code.
- There is, in fact, a very simple pattern for using QThread without sub-classing. There is a very good tutorial covering this pattern (see link below) which we use in our own code sample.
  - http://mayaposch.wordpress.com/2011/11/01/how-to-really-truly-use-qthreads-the-full-explanation/

BlackBerry 10 Jam

Closing Remarks

Martin Woolley

For More Information…

BlackBerry 10 Jam

- Articles and open source code:
  - http://supportforums.blackberry.com/t5/Java-Development/NFC-Article-and-Code-Index/ta-p/1538775
- Twitter
  - Martin - @mdwrim
  - John - @jcmrim
  - @blackberrydev
- Come and talk to us here at the event!
  - Ask the Experts
- BlackBerry Developer Blog - http://devblog.blackberry.com

BlackBerry 10 Jam

THANK YOU

John Murray and Martin Woolley

DEV120

May 1-3, 2012