嵌入式计算机系统

Lecture #7

MeeGo Communications

内容来自于meego.com以及MeeGo相关公开教程

Communication API

- Communications services consists of APIs related to social and human interaction, connectivity, and networking
- Communications API can be grouped into four parts according to services:
  - Qt WebKit
  - Messaging
  - Qt Network
  - Qt DBus

Qt WebKit

- QtWebKit provides a Web browser engine that makes it easy to
  - embed content from www into Qt application
  - enhance web content with native controls
- To include the definitions of QtWebKit, use :

```
#include <QtWebKit>
```

- To link against the module, add this line to qmake .pro file:

```
QT += webkit
```

Qt WebKit Classes

- QWebView
  - Widget that is used to view and edit web documents

```
QWebView *view = new QWebView(parent);
view->load(QUrl("http://www.sjtu.edu.cn"));
view->show();
```

Qt WebKit Classes

- QWebPage
  - Object to view and edit web documents

```
m_page.mainFrame()->load(url);

m_page.mainFrame()->setScrollBarPolicy(Qt::Vertical,
Qt::ScrollBarAlwaysOff);

m_page.mainFrame()->setScrollBarPolicy(Qt::Horizontal,
Qt::ScrollBarAlwaysOff);

m_page.setViewportSize(QSize(1024, 768));
```

Qt WebKit Classes

- QWebFrame
  - Represents a frame in a web page
  - Each QWebPage object contains at least one frame, the main frame, obtained using QWebPage::mainFrame().

```
m_page.mainFrame()->load(url);

m_page.mainFrame()->setScrollBarPolicy(Qt::Vertical,
Qt::ScrollBarAlwaysOff);

m_page.mainFrame()->setScrollBarPolicy(Qt::Horizontal,
Qt::ScrollBarAlwaysOff);

m_page.setViewportSize(QSize(1024, 768));
```

Qt WebKit Classes

- QWebElement
  - Convenient access to DOM elements in a QWebFrame
  - The root of the tree is called the document element and can be accessed using QWebFrame::documentElement().

```
frame->setHtml("<html><body><p>First Paragraph</p><p>Second
Paragraph</p></body></html>");

QWebElement doc = frame->documentElement();

QWebElement body = doc.firstChild();

QWebElement firstParagraph = body.firstChild();

QWebElement secondParagraph = firstParagraph.nextSibling();
```

Examples

- previewer
- fancybrowser

Qt Messaging

- The QtMessaging module enables access to messaging services to
  - search and sort messages
  - send messages
  - retrieve message data
  - launch the preferred messaging client.

Qt Messaging Classes

- Qmessage
  - The QMessage class provides a convenient interface for working with messages.
  - QMessage supports a number of types including internet email messages, and the telephony types SMS and MMS.

Qt Messaging Classes

- QMessageAccount
  - The QMessageAccount class represents a messaging account in the messaging store.

  - The QMessageAccount class is used for accessing properties of the account related to dealing with the account's folders and messages, rather than for modifying the account itself.

Qt Messaging Classes

- QMessageAddress
  - The QMessageAddress class provides an interface for a message address.
  - A message address consists of an addressee string and a type.
    - Systme
    - Phone
    - Email
    - InstantMessage

Qt Messaging Classes

- QMessageManager
  - The QMessageManager class represents the main interface for storage and retrieval of messages, folders and accounts in the system message store.
  - QMessageManager provides the interface for adding, updating and deleting messages in the system's message store.

Qt Messaging Classes

- QMessageService
  - The QMessageService class provides the interface for requesting messaging service operations.
  - QMessageService provides the mechanisms for messaging clients to request services, and to receive information in response.
  - All requestable service operations present the same interface for communicating status, and progress information.

An example：write message

- This example demonstrates using the Qt Mobility Messaging API to create and send a simple message.

Qt Network

- The QtNetwork module provides classes to make network programming easier and portable.
  - Classes for networking programming
  - Opening, maintaining and closing of network session using various protocols
  - Servers for accepting connections

QtNetwork

- Some important classes included in QtNetwork module
  - QNetworkAccessManager
  - QNetworkRequest
  - QNetworkReply
  - QTcpServer
  - QTcpSocket
  - QFtp
- Steps to use this module
  - #include <QtNetwork>
  - Add QT += network to .pro file

QNetworkAccessManager

- Send network request and receive replies
- Holds common configuration and settings for the request
- Contains the proxy and cache configuration
- Reply signals to monitor the progress of a network operation

QNetworkAccessManager

- example of download using QNetworkAccessManager

```
QNetworkAccessManager *manager = new QNetworkAccessManager(this);

connect(manager,SIGNAL(finished(QNetworkReply*)),

            this, SLOT(replyFinished(QNetworkReply*)));

Manager->get(QNetworkRequest(Qurl(“http://qt.nokia.com”)));
```

QNetworkRequest

- Hold the information necessary to send a request over the network
- Contains a URL and some ancillary information that can be used to modify the request

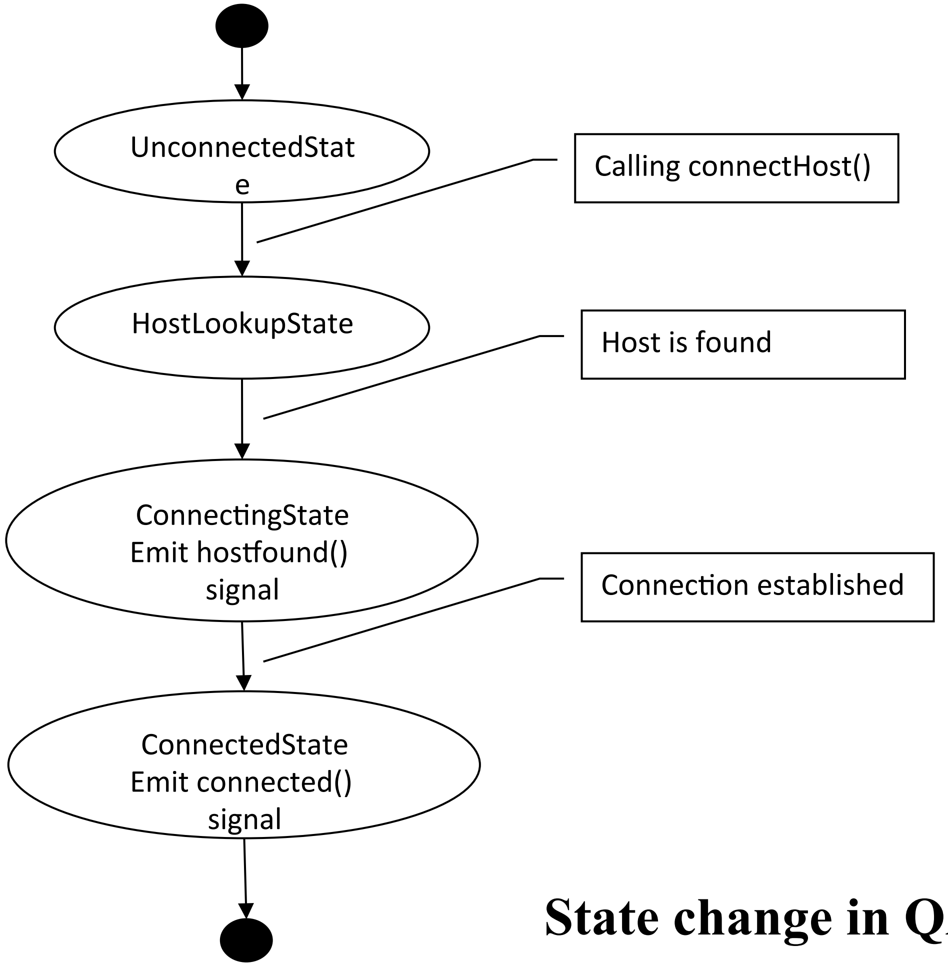
QNetworkReply

- Contain the data and headers for a request sent with QNetworkAccessManager
- QNetworkReply is a sequential-access QIODevice, whenever more data is received from the network, the readyRead() signal is emitted.
- The downloadProcess() signal is also emitted when data is received

QAbstractSocket

- The QAbstractSocket class provides the base functionality common to all socket types
- QAbstractSocket is the base class for QTcpSocket and QUdpSocket and contain all common functionality of these two classes
- There are two way to create socket:
  - Instantiate QTcpSocket or QUdpSocket
  - Create a native socket descriptor, instantiate QAbstractSocket, and call setSocketDescriptor() to wrap the native socket

QAbstractSocket



State change in QAbstractSocket

QAbstractSocket

- In QAbstractSocket read and write data by calling read() and write()
- The readyReady() signal is emitted every time a new chunk of data has arrived. bytesAvailable() returns the number of bytes that are available for reading
- The bytesWritten() signal is emitted when the data has written to the socket

QTcpServer

- The QTcpSerer class provides a TCP-based server
- Call listen() to have the server listen for incoming connections. The newConnection() signal is emitted each time a client connects to the server.
- If port is 0, a port is chosen automatically. If address is QHostAddress::Any, the server will listen on all network interfaces.

Examples

- network-chat
- broadcast-sender/receiver

QtDBus

- What is D-Bus
- D-Bus is a system for interprocess communication(IPC)
  - Low latency: it is designed to avoid round trips and allow asynchronous operation
  - Low head: it use a binary protocol and does not have to convert to and from a text format such as XML
  - Easy to use: it works in terms of message rather than byte stream and automatically handles lots of the hard IPC issues

QtDBus

- Three layers in D-Bus
  - Library libdbus that allows two application to connect to each other and exchange messages
  - Message bus daemon executable built on libdbus can route messages from one application to other ones
  - Wrapper libraries or bindings on particular application framework such as libdbus-glib and libdbus-qt

QtDBus

- Concepts in D-Bus
- Services Names
  - Services name is how that application choose to be known by other application on the same bus
  - The format of a D-Bus service name is dot-separate sequence of letters and digits. The example of a service nam is:

    org.freedesktop.DBus

QtDBus

- Concepts in D-Bus
- Object Paths
  - An object path is that higer-level bindings can name native object instances and allow remote application to refer to them
  - The format of the object path looks like filesystem path

    /com/mycompany/test

QtDBus

- Concepts in D-Bus
- Interface
  - Interfaces are similar to C++ abstract classes and Java’s interface keyword and declare the contracts that is established between caller and callee
  - DBus identifies interfaces with a simple namespaced string something like

    org.freedesktop.Introspectable

QtDBus

- Concepts in D-Bus
- Messages
- D-Bus works by sending messages between processes. There are four message types:
  - Method call message ask to invoke a method on an object
  - Method return message return the results of invoking a method
  - Error message return an exception caused by invoking a method
  - Signal message are notifications that a given signal has been emitted

QtDBus

- Some important classes included in QtDBus module
  - QDBusMessage
  - QDBusConnection
  - QDBusInterface
  - QDBusObjectPath
  - QDBusAbstractAdaptor
- Steps to use this module
  - #include <QtDBus>
  - Add QT += dbus to .pro file

QDBusConnection

- The QDBusConnection class represents a connection to the D-Bus deamon which is used to get access to remote objects, interfaces; connect remote signals to object's slots; register object, etc
- D-Bus connections are created using the connectToBus() function which opens a connection to the server daemon
- The sessionBus() and systemBus() return open connections to the session server daemon and the system server daemon

QDBusMessage

- The QDBusMessage represents one message sent or received over the D-Bus bus
- Four different types of message in class QDBusMessage:
  - QDBusMessage::MethodCallMessage
  - QDBusMessage::SignalMessage
  - QDBusMessage::ReplyMessage
  - QDBusMessage::ErrorMessage
  - QDBusMessage::InvalidMessage

  Objects of this type are created with static createError(), createMethodCall() and createSignal() function