Qt na mobilních zařízeních

Vývoj mobilní Datovky

Karel Slaný • karel.slany@nic.cz • 24. 11. 2017

Stav projektu Datovka v roce 2016

- desktopová Datovka
  - UNIX/Linux, mac OS, Windows
  - Od verze 4.8.1 nepodporujeme Windows XP.
  - Qt 5.3 (5.2+), OpenSSL, libisds
- Datovka pro Android
  - TinyISDS, ISDSCommon, …
  - Java
- iDatovka
  - ObjC
- Byl požadavek na sjednocení funkcionality a vývoje mobilních Datovek.

Qt 5

- Qt 5.0/12-2012 - proti Qt 4 vylepšená podpora QML a JavaScript
- „kompatibilní“ s Qt4
- abstrakce nad platformami
  - X11, Wayland, Windows, Android, Jolla, macOS, iOS, QNX/Blackberry10
- „plná“ podpora pro Android, iOS, WinRT
  - iOS a Android od 5.2/12-2013 (5.1/07-2013)
  - WinRT od (5.3/05-2014)
- Desktopová Datovka 4 je v Qt, chtěli jsme využít již existující kód.

Aplikace v Qt

- *.pro
  - qmake: *.pro → Makefile
- *.ui
  - uic: *.ui → ui_*.h
- *.h + *.cpp
  - moc: *.h → moc_*.cpp
- *.qml (deklarované v *.res)
  - rcc -name qml res/qml.qrc -o qrc_qml.cpp

```
# project.pro

QT += core gui network sql printsupport widgets
TEMPLATE = app
APP_NAME = datovka
VERSION = 4.7.1
DEFINES += DEBUG=1
QMAKE_CXXFLAGS = -g -O0 -std=c++11
INCLUDEPATH += src
LIBS = -lisds -lcrypto

SOURCES += ...
HEADERS += …
FORMS += …
RESOURCES += …
TRANSLATIONS += ...
```

```
/* qrc_qml.cpp */

static const unsigned char qt_resource_data[] = {
  /* ... */
  // qml/main.qml
  0x0,0x0,0xc,0xa,
  0x0,
  0x0,0x2b,0x77,0x78,0x9c,0xed,0x1a,0x6b,0x73,
  0xdb,0xb8,0xf1,0xbb,0x7f,0x5,0x4e,0xbd,0x99,
  0x93,0xf2,0Xa0,...
  /* ... */
};
```

Vývoj mobilní aplikace v Qt

- Předpokládá se použití Qt Quick.
- Kromě Qt je potřeba (dle cílové platformy):
  - Android SDK a NDK, Java SE JDK, Apache Ant
  - XCode + Command Line Tools
  - Visual Studio 2013 (2015 pro Windows 10)
- Výsledná aplikace je v nativním binárním kódu pro cílovou platformu.

QML / Qt Quick

- deklarativní jazyk
- syntaxe podobná JSON
- navrženo pro dotykové ovládání, animace, styly, …
- podpora JavaScript
- strom objektů reprezentující elementy
  - Dají se sestavovat vlastní komponenty.
- Lze integrovat vlastní elementy napsané v C++.

Aplikace v Qt Quick

- smyčka aplikace
- QML engine
- nahraje se QML
- Lze psát programy „pouze“ v QML.

```
/* qml/main.qml */

import QtQuick 2.0
Item {
  id: item
  width: 100; height: 100
  signal qmlSignal(string msg)
  MouseArea {
    anchors.fill: parent
    onClicked: item.qmlSignal("Hello from QML")
  }
}
```

```
/* main.cpp */

#include <Qapplication>
#include <QtQuick>

int main(int argc, char *argv[]) {
  QGuiApplication app(argc, argv);

  /*
  QQuickView view(QUrl::fromLocalFile("qml/main.qml"));
  view.show();
   */

  QQmlApplicationEngine engine;
  engine.load(
    QUrl(QStringLiteral("qrc:/qml/main.qml")));

  return app.exec();
}
```

QML a C++ objekty

- QQuickItem, QObject

```
/* QmlObj.qml */
import QtQuick 2.7

Item {
  id: root

  property int val : 0

  signal signalValue(int sigVal)

  function doSomething(val) {
    console.log("Value " + val)
  }

  onSignalValue: {
    doSomething(sigVal)
  }
}
```

```
/* CppObj.h */
#pragma once
#include <QObject>

class CppObj : public QObject {
  Q_OBJECT
public:
  explicit CppObj(QObject *parent = Q_NULLPTR);
  Q_INVOKABLE void setVal(int val);
signals:
  void signalValue(int val);
public slots:
  void doSomething(int val);
private:
  int m_val;
};
```

```
/* CppObj.cpp */
#include <QDebug>
#include "CppObj.h"

CppObj::CppObj(QObject *parent) : QObject(parent), m_val(0) {
  connect(this, SIGNAL(signalValue(int)),
    this, SLOT(doSomething(int)));
}

void CppObj::setVal(int val) { m_val = val; }

void CppObj::doSomething(int val) {
  qDebug() << "Value" << val;
}
```

Dědičnost v QML

- Jeden soubor definuje strukturu jedné komponenty.
- Jméno souboru je důležité.

```
/* main.qml */

import "components"
// import cz.nic.mobileDatovka 1.0

Item {
  id: item01

  SpinBoxZeroMax { id: spibBox }
}
```

```
/* components/SpinBoxZeroMax.qml */
import QtQuick.Controls 2.0

SpinBox {
  property var items: [1, qsTr("max")]
  property int dfltIdx: 0

  from: 0
  to: items.length - 1 /* Last is infinite. */

  textFromValue: function(idx) { /* text at idx */ }

  valueFromText: function(text) { /* idx with text */ }

  function val() { /* value at current index */ }

  function setVal(v) { /* set index nearest to v */ }
}
```

```
/* main.cpp */

QmlRegisterType(Qurl("qrc:/components/SpinBoxZeroMax"), "cz.nic.mobileDatovka",
  1, 0, "SpinBoxZeroMax");
```

C++ do QML

- přístup k objektům
- přístup k třídám
- parametry přes Qvariant nebo QString

```
/* main.qml */
import Qt.Quick 2.7
import cz.nic.mobileDatovka 1.0

CppObj {
    id: object
    val: 0
}

Item {
    id: obj
    property int val: objCpp.val
}
```

```
/* CppObj.h */
#pragma once
#include <QObject>

class CppObj : public QQuickItem {
  Q_OBJECT
  Q_PROPERTY(QString val READ val WRITE setVal
    NOTIFY valChanged)

public:
  CppObj(QObject *parent = Q_NULLPTR)
    : QObject(parent), m_val(0) { }

  int val(void) const { return m_val; }
  void setVal(const QVariant &v) {
    bool ok;
    int i = v.toInt(&ok);
    if (ok) { m_val = i; emit valChanged(); }
  }

signals:
  void valChanged(void);

private:
  int m_val;
};
Q_DECLARE_METATYPE(CppObj)
```

```
/* main.cpp */
#include "CppObj.h"

QQmlApplicationEngine engine;
QQmlContext *ctx = engine.rootContext();

CppObj obj;

ctx->setContextProperty("objCpp", &obj);
qmlRegisterType<CppObj>("cz.nic.mobileDatovka", 1, 0, "CppObj");
```

QML do C++

- příklad: blokující dialog
  - vyhledání objektu
  - ne vše je QQuickItem
  - pozor na vlákna

```
/* .cpp */
QEventLoop loop; /* Must lie in same thread as engine. */
QQmlApplicationEngine engine;

/* ... */

QList<QObject *> objList = engine.rootObjects()
QObject *root = objList.first(); /* QQuickItem */
QObject *dlg = = root->findChild<QObject *>("dialogue",
    Qt::FindChildrenRecursively); /* not a QQuickItem. */

dlg->setProperty("title", QVariant("Dialogue message"));

QObject::connect(dlg, SIGNAL(dialogueClosed()), &loop, SLOT(quit()));
/* Show dialogue. */
QMetaObject::invokeMethod(dlg, "open");

loop.exec(); /* Wait for dialogue to close. */
```

```
/* dialogue.qml */
import QtQuick 2.7
import QtQuick.Controls 2.0
import QtQuick.Dialogs 1.2

/*
 * objectName properties are used to
 * navigate through the structure from
 * within C++ code.
 */

Item {
  id: root
  objectName: "root"

  signal dialogueClosed()

  Dialog {
    id: dialogue
    objectName: "dialogue"

    /* Set from C++. */
    title: ""

    modality: Qt.ApplicationModal

    StandardButtons: StandardButton.Ok |
      StandardButton.Cancel

    onAccepted: {
      root.dialogueClosed()
    }
    onRejected: {
      root.dialogueClosed()
    }
  }
}
```

Vykonávání QML

- Struktura stromu objektů a jejich závislostí (signály, sloty) je vytvořena při inicializaci - QQmlApplicationEngine::load().
  - Chyby často způsobí pád aplikace.
- JavaScript je interpretován, když je potřeba.
  - Chyby se projeví za běhu aplikace, někdy způsobí pád, někdy zamrznutí.
- Import statement slouží spíše k deklaraci požadované funkcionality.
- Pokud QML funguje na jedné platformě, není jisté, že na ostatních platformách bude program fungovat také.

QML – best practice

- 3 programové vrstvy
  - prezentace/zobrazování dat (QML)
  - konverze dat pro prezentaci, zpracování uživatelských vstupů pro jádro (C++)
  - jádro aplikace pro komunikaci se serverem a správu uložených dat
- Udržovat QML v přehledné a krátké formě.
- Minimalizovat použití JS v QML, funkce psát raději v C++ a exportovat do QML.

Obcházení „nedostatků“ Qt

- AndroidExtras
  - soubor tříd a objektů pro komunikaci s Androidem
  - Lze volat kód v Javě.
  - požíváme s Android Intent (zpracování událostí)
- MacExtras
  - Zpřístupňuje některé funkce OS X
    - specifické datové typy, grafika a GUI
    - komunikace se „správcem oken“
  - málo funkcí (protože jich není zapotřebí?)
    - Lze psát funkce v ObjC, které mají rozhraní v C.
      - SWIFT je „kompatibilní“ s ObjC.
    - Lze psát funkce v C++, které mají rozhraní v C.

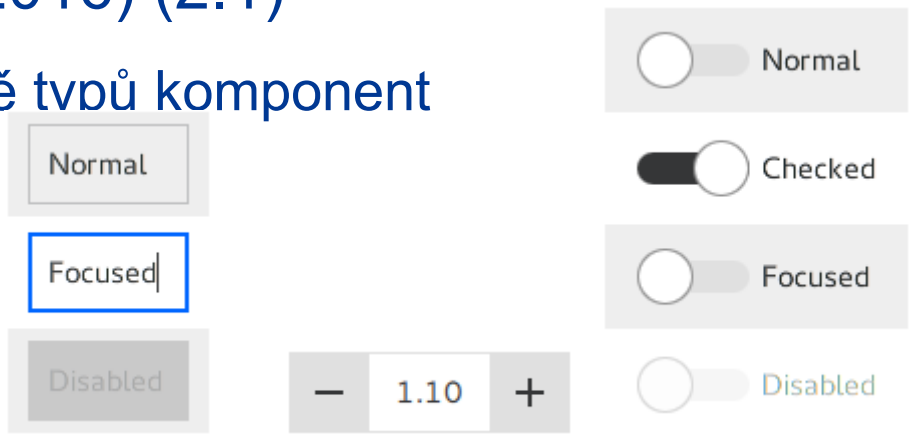
Dokumentace Qt

- Rozhraní C++ má vynikající dokumentaci (Qt Assistant).
- Dokumentace QML se nese v podobném duchu, ale …
- Mnohé komponenty jsou potomky jiných.
  - Často chybí informace, co je rodič komponenty.
  - Kompletní funkcionalitu lze odhadnout až ze zdrojových souborů.
  - Komponenty se v různých verzích (import ...) liší v některých atributech. V mnoha případech není uvedeno, kdy byl který atribut přidán.
- Pomáhá použití Qt Creatoru.

Vzhled ovládacích prvků

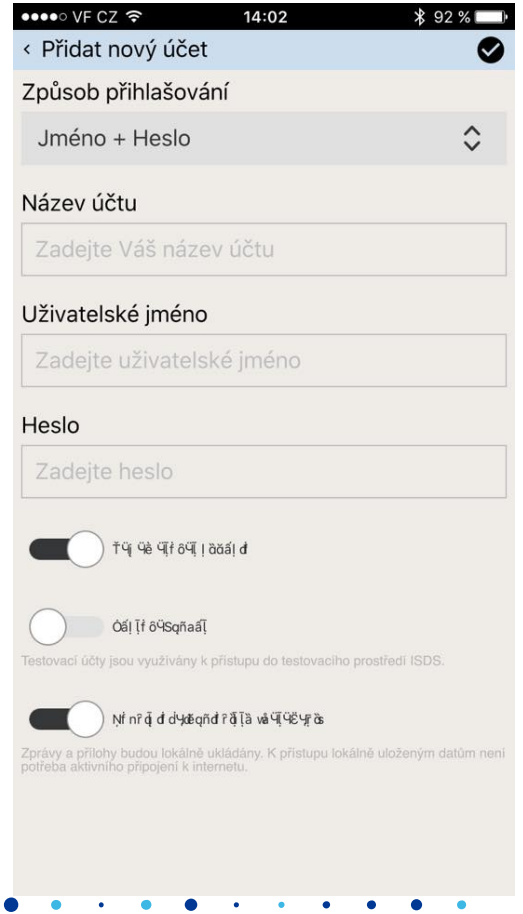
- QuickControls 1.4
  - starší, více komponent, „desktopový“ vzhled

- QuickControls 2.0 (Qt 5.7/06-2016) (2.1)
  - moderní „mobilní vzhled“, méně typů komponent
    - chybí např. TableView



Bolístky

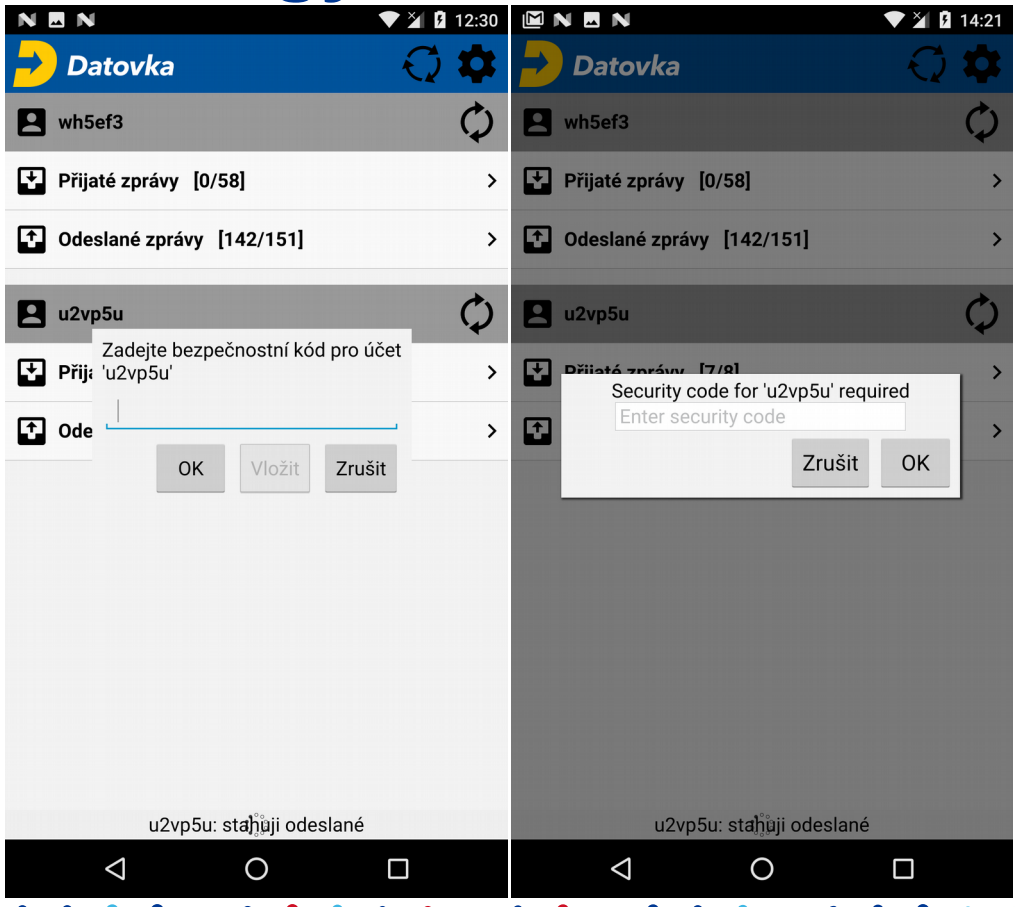
- někdy rozsypané fonty na iOS
  - řešení: zdola omezit velikost fontů



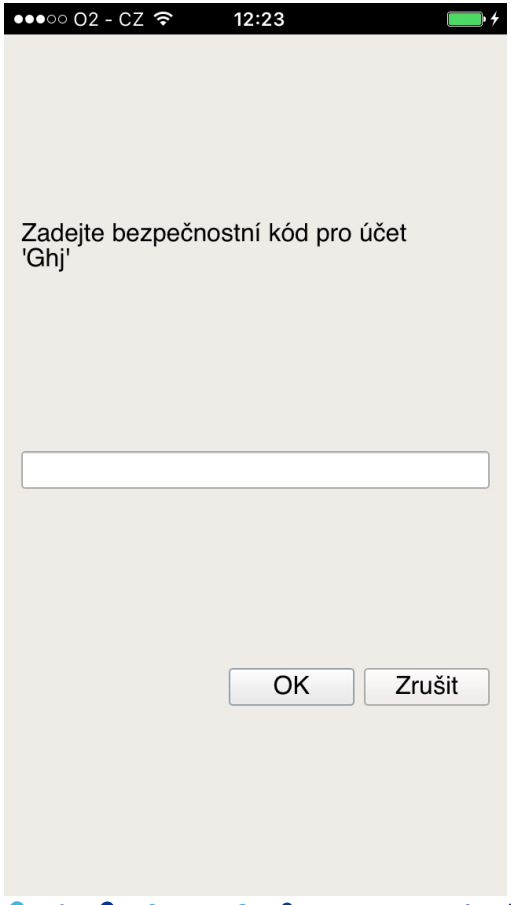
Další bolest

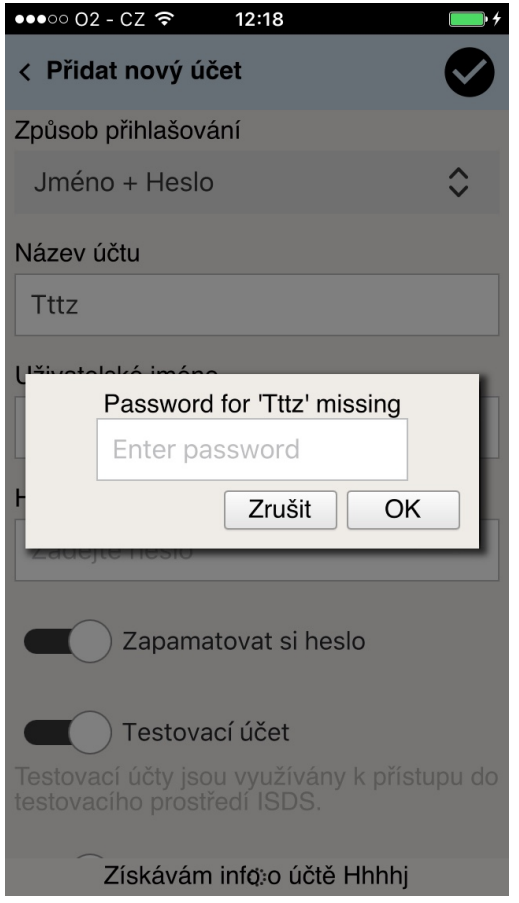
- Občas potřebujeme QML dialogy, které blokují vykonávání C++ kódu.
- Dialogy v QML jsou sice pěkné a blokující ale…
  - Na Androidu po zobrazení ukradnou funkci tlačítka zpět.
  - Na iOS se po zobrazení zablokuje smyčka událostí a aplikace zamrzne.
  - Na desktopu fungují bez problémů. :)

Dialogy


12:30
Datovka
wh5ef3
Přijaté zprávy [0/58]
Odeslané zprávy [142/151]
u2vp5u
Zadejte bezpečnostní kód pro účet 'u2vp5u'
OK
Vložit
Zrušit
u2vp5u: stahuji odeslané

14:21
Datovka
wh5ef3
Přijaté zprávy [0/58]
Odeslané zprávy [142/151]
u2vp5u
Security code for 'u2vp5u' required
Enter security code
Zrušit
OK
u2vp5u: stahuji odeslané


O2 - CZ
12:23
Zadejte bezpečnostní kód pro účet 'Ghj'
OK
Zrušit


O2 - CZ
12:18
Přidat nový účet
Způsob přihlašování
Jméno + Heslo
Název účtu
Tttz
Password for 'Tttz' missing
Enter password
Zrušit
OK
Zapamatovat si heslo
Testovací účet
Testovací účty jsou využívány k přístupu do testovacího prostředí ISDS.
Získávám info o účtě Hhhhj

Chyby v Qt

- It‘s not a bug - it‘s a feature.
  - Vývojáři Qt raději implementují novou funkcionalitu než opravují staré chyby.
    - otevírání emailů na iOS – workaround since Qt 5.2
- The fix would break backward compatibility.
  - Problém napojení a zpracování signálů mezi C++ a QML na iOS.
    - A conceptual change needed, fix likely in Qt 6.
- V Qt 5.8 jde na Androidu konečně vkládat text ze schránky.

Získané poznatky

- Jsme relativně spokojeni s použitelností Qt pro vývoj mobilní aplikace.
- Funguje-li něco na jedné platformě, ještě to nemusí fungovat všude.
- Změna způsobu ovládání a vzhledu vyvolala nečekanou vlnu negativních reakcí.

Děkuji za pozornost

Karel Slaný • karel.slany@nic.cz