Vývoj multiplatformní aplikace v Qt

z pohledu vývoje Datovky

Karel Slaný • karel.slany@nic.cz • 13. 11. 2015

CZ.NIC | SPRÁVCE DOMÉNY CZ

Obsah

- Co je Qt
- Nástroje Qt
- Koncepty Qt
- Problémy při vývoji Datovky
- Balíčkování aplikace

Datovka

- Desktopová aplikace pro příjem a odesílání datových zpráv.
- Dovoluje spravovat více účtů současně.
- Podobnost s poštovním klientem
- Zprávy se ukládají do lokální databáze.
  - Částečně nahrazuje datový trezor.
- Export korespondence za vybrané časové období.
- Podpora všech přihlašovacích metod.
- Multiplatformí
  - Windows, UNIX-like (Linux, FreeBSD, ...), OS X

Qt

- Sada knihoven a nástrojů pro vývoj multiplatformních grafických aplikací v C++.
  - Windows, X11(Linux, UNIX-like), OS X
- Přenositelné datové typy
- Kontejnerové třídy
- Abstrakce nad službami OS
  - Síť, souborový systém, vlákna, IPC
- Podpora databázových systémů
  - Přímé zobrazení výsledku SQL dotazu v tabulce
- Přehledná a dostupná dokumentace

Projekt v Qt

- Projektový soubor
  - Výčet zdrojových souborů
  - Nastavení parametrů překladu
  - Dovoluje podmíněný překlad
  - Základní podpora skriptování
- qmake
  - Generuje Makefile.

```
TEMPLATE = app
APP_NAME = datovka

VERSION = 4.4.1

macx {
 ICON = res/datovka.icns

 QMAKE_CXXFLAGS += -mmacosx-version-min=10.7
 CONFIG += c++11
 isEmpty(SDK_VER) {
  SDK_VER = 10.7
 }
 QMAKE_MAC_SDK = macosx$${SDK_VER}
 QMAKE_MACOSX_DEPLOYMENT_TARGET = 10.6
}

SOURCES += ...

HEADERS += ...

FORMS += ...
```

Návrh UI

- Qt Designer
  - UI se sestavuje přetahováním komponent
  - Výsledkem je XML soubor s popisem
- User Interface Compiler (uic)
  - Vstup *.ui soubor
  - Výstup ui_*.h soubor
    - Třída jejíž konstruktor nastavuje UI

```
<item>
 <widget class="QLineEdit"
  name="filterLineEdit">
  <property name="focusPolicy">
   <enum>Qt::ClickFocus</enum>
  </property>
 </widget>
</item>
```

```
QLineEdit *filterLineEdit;

FilterLineEdit =
  new QLineEdit(Contacts);
filterLineEdit->setObjectName(
  QStringLiteral("filterLineEdit"));
filterLineEdit->setFocusPolicy(
  Qt::ClickFocus);

horizontalLayout->addWidget(
  filterLineEdit);
```

Lokalizace aplikace

- Metoda QObject::tr()
- Nástroj lupdate
  - .ui – XML soubor s řetězci na překlad
- Qt Linguist
  - .ui – XML soubor s překlady
- Nástroj lrelease
  - .qm – binární soubor obsahující lokalizaci
- Třída QTranslator
  - Nahrává se do QApplication

```
fileName = QfileDialog::getSaveFileName(this,
  tr("Save attachments"),
  saveAttachPath + QDir::separator() + fileName);
```

```
<message>
 <location filename="gui/datovka.cpp" line="1866"/>
 <source>Save attachments</source>
 <translation>Uložit všechny přílohy</translation>
</message>
```

```
static QTranslator appTranslator;
appTranslator.load(localisationFile,
  LocalisationDir);

app.installTranslator(&appTranslator);
```

Zdrojové kódy C++

- Rozšiřuje klíčová slova C++.
  - Prý to není C++, znečišťuje namespace.
- Základní schopnost introspekce
- Rozšiřuje klíčová slova C++
- Meta-Object Compiler (moc)
  - Skenuje hlavičkové soubory *.h
  - moc_*.cpp – metaobjektový kód

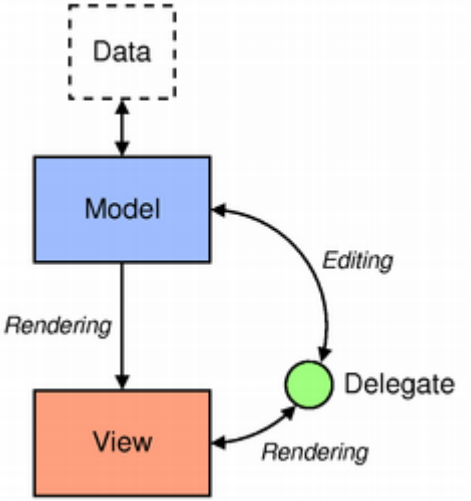
```
class DlgViewZfo :
  public QDialog, public Ui::ViewZfo {

  Q_OBJECT

slots:
  void attachClicked(QPoint &point);

}
```

Signály a sloty

- Základní koncept zpracování událostí
- Signál je vyvolán z kódu.
  - Klíčové slovo emit
- Zpracování připojeným slotem
  - Nutné spojit pomocí connect()
- Zpracování probíhá ve smyčce obsluhy
- Lze předávat vlastní datové typy
  - Definovaný kopírovací konstruktor
  - Typ musí být zaregistrován

```
class SendJobInfo {
 public:
  qint64 timestamp;
  qint32 recipientNum;
};

Q_DECLARE_METATYPE(SendJobInfo)


class Worker {
 ...
signals:
 void sendSig(SendJobInfo);
};


connect(m_worker, SIGNAL(sendSig(SendJobInfo)),
  this, process(SendJobInfo));


qRegisterMetaType<SendJobInfo>("SendJobInfo");
```

Koncept model view



- Data nejsou vkládána přímo do widgetů.
- Data jsou organizována do struktur modelů.
- O zobrazení se starají view widgety.
- Model a view komunikují zasíláním signálů.
  - Při změně dat modelu se view automaticky postará o překreslení.
- O formátování dat se stará model.
  - Lze změnit definicí vlastní funkce metody data()

Kde jsme v Datovce s Qt nevystačili

- Parsování a kontrola certifikátů
  - QSslCertificate neposkytuje dostatečnou funkcionalitu.
- CMS
  - Formát podepsaných datových zpráv (.zfo)
  - Načítání dat z kontejneru
  - Verifikace dat
- SOAP
  - libisds (libcurl, libxml2)

Co nefungovalo podle očekávání

- Horní aplikační menu na OS X
- Pojmenovávání souborů a omezení souborových systémů
- Zvětšování/zmenšování oken
- Práce nad několika databázemi současně
- Navigace s tabulkách
- Příkazová řádka na Windows

Static initialisation order fiasco

- Problém C++, nesouvisí přímo s Qt.
- Pořadí inicializace statických instancí není jednoznačně definováno.
- Může dojít ke špatnému pořadí inicializace, když se nachází ve více překladových jednotkách.
- Dá se obejít metodou „vytvoř při prvním použití“.
  - Statický objekt je zabalen do funkce.

Balíčkování Datovky

- UNIX-like
  - Archivy zdrojových souborů
  - Distribuční balíčky
- Windows
  - Nástroj Qt binarycreator
  - Používáme NSIS
- OS X
  - Balík aplikace pro OX má definovanou strukturu.
  - Open Scripting Architecture (OSA) skript pro výrobu .dmg

Děkuji za pozornost

Karel Slaný • karelslany@nic.cz