Tłumaczenie graficznego interfejsu użytkownika

Bogdan Kreczmer

bogdan.kreczmer@pwr.wroc.pl

Katedra Cybernetyki i Robotyki
Wydziału Elektroniki, Fotoniki i Mikrosystemów
Politechnika Wrocławska

Kurs: Wizualizacja danych sensorycznych

Niniejsza prezentacja została wykonana przy użyciu systemu składu LaTeX *oraz stylu* `beamer`, *którego autorem jest Till Tantau.*

Strona domowa projektu `Beamer`:
`http://latex-beamer.sourceforge.net`

Outline

Korzystanie z metod translacji

Źle:

```
const char *wNapis = "To należy przetłumaczyć";
QString TlumaczonyNapis = wNapis;
```

Dobrze:

```
QString TlumaczonyNapis = tr("To należy przetłumaczyć");
```

Źle:

```
QString NazwaPliku;
...
statusBar( )->message( tr("Nie znaleziono pliku:" + NazwaPliku) );
```

Dobrze:

```
statusBar( )->message( tr("Nie znaleziono pliku: %1").arg(NazwaPliku) );
```

Korzystanie z metod translacji

Nie zaleca się podstawiania do zmiennych napisów, które mają być tlumaczone. Jeżeli jest to konieczne, należy to uczynić z zastosowaniem makr `QT_TR_NOOP` lub `QT_TRANSLATION_NOOP`, np.

```
const char* const TabNapisow[ ] =
        QT_TR_NOOP("Napis tłumaczony 1"),
        QT_TR_NOOP("Napis tłumaczony 2")
    };
```

Tłumaczenie – przykład praktycznej realizacji

0. Jeżeli w aplikacji są napisy po polsku w kodowaniu UTF-8, to wystarczy dodać do projektu wpis:

   ```
   TRANSLATIONS = skromny_edytor_en.ts
   ```

   Jeżeli kodowanie jest inne np. `Latin2 (ISO 8859-2)`, to niezbędny jest dodatkowy wpis:

   ```
   DEFAULTCODEC = ISO-8859-2
   ```

   W przypadku realizacji tłumaczeń angielskich napisów w aplikacji na polski nie jest potrzebna dodatkowa informacja o kodowaniu i wystarczy wpis:

   ```
   TRANSLATIONS = skromny_edytor_pl.ts
   ```

1 `lupdate -verbose skromny_edytor.pro` $\longrightarrow$ `skromny_edytor_pl.ts`

2 `linguist skromny_edytor_pl.ts` $\longrightarrow$ `skromny_edytor_pl.ts`

3 `lrelease -verbose skromny_edytor.pro` $\longrightarrow$ `skromny_edytor_pl.qm`

Krok 0 – przykładowy plik projektu

```
#############################################################################
# Automatically generated by qmake (3.0) pon. cze 3 11:48:15 2019
#############################################################################

OBJECTS_DIR=./obj
INCLUDEPATH=inc
INCLUDEPATH+=out/ui
MOC_DIR=out/moc
UI_DIR=out/ui
RCC_DIR=out/rcc
TRANSLATIONS=interapp_pl.ts
TRANSLATIONS+=interapp_ge.ts
TRANSLATIONS+=interapp_sp.ts
QT+=widgets
TEMPLATE = app
TARGET = interapp
INCLUDEPATH += ../inc

# Input
HEADERS += ../inc/OknoGlowneApp.hh
FORMS += ../ui/oknoglowne.ui
SOURCES += ../src/OknoGlowneApp.cpp ../src/start.cpp
RESOURCES += ../res/przyklad.qrc
```

Krok 1 – lupdate

```
jk@callisto:popolsku-1-najprostsze$ lupdate -verbose pro/interapp.pro
Updating 'pro/interapp_pl.ts'...
    Found 12 source text(s) (12 new and 0 already existing)
Updating 'pro/interapp_ge.ts'...
    Found 12 source text(s) (12 new and 0 already existing)
Updating 'pro/interapp_sp.ts'...
    Found 12 source text(s) (12 new and 0 already existing)
jk@callisto:popolsku-1-najprostsze$ _
```

```
jk@callisto:popolsku-1-najprostsze$ lupdate -verbose pro/interapp.pro
Updating 'pro/interapp_pl.ts'...
    Found 12 source text(s) (0 new and 12 already existing)
Updating 'pro/interapp_ge.ts'...
    Found 12 source text(s) (0 new and 12 already existing)
Updating 'pro/interapp_sp.ts'...
    Found 12 source text(s) (0 new and 12 already existing)
jk@callisto:popolsku-1-najprostsze$ _
```

Co wygenerował lupdate – interapp_pl.ts

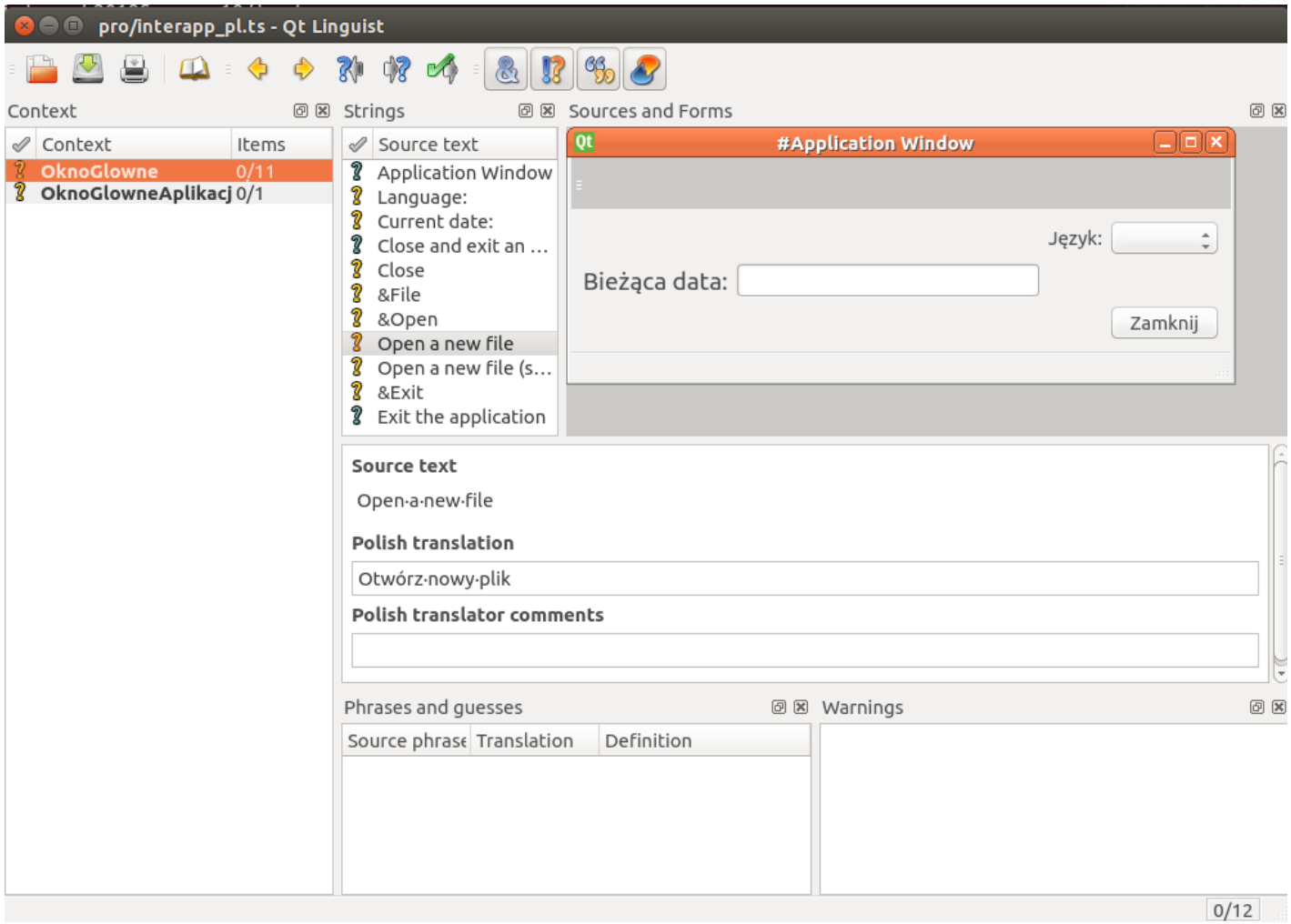
```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE TS>
<TS version="2.1" language="pl_PL">
<context>
    <name>OknoGlowne</name>
    <message>
        <location filename="../ui/oknoglowne.ui" line="14"/>
        <source>Application Window</source>
        <translation type="unfinished"></translation>
    </message>

     ...

    <message>
        <location filename="../ui/oknoglowne.ui" line="145"/>
        <source>Exit the application</source>
        <translation type="unfinished"></translation>
    </message>
</context>
<context>
    <name>OknoGlowneAplikacji</name>
    <message>
        <location filename="../src/OknoGlowneApp.cpp" line="46"/>
        <source>Otworz plik</source>
        <translation type="unfinished"></translation>
    </message>
</context>
</TS>
```

Krok 2 – linguist

```
jk@callisto:popolsku-1-najprostsze$ linguist pro/interapp_pl.ts
```



Krok 3 – lrelease

```
jk@callisto:popolsku-1-najprostsze$ lrelease -verbose pro/interapp.pro
Updating '/home/jk/.../popolsku-1-najprostsze/pro/interapp_pl.qm'...
    Generated 8 translation(s) (0 finished and 8 unfinished)
    Ignored 4 untranslated source text(s)
Updating '/home/jk/.../popolsku-1-najprostsze/pro/interapp_ge.qm'...
    Generated 0 translation(s) (0 finished and 0 unfinished)
    Ignored 12 untranslated source text(s)
Updating '/home/jk/.../popolsku-1-najprostsze/pro/interapp_sp.qm'...
    Generated 0 translation(s) (0 finished and 0 unfinished)
    Ignored 12 untranslated source text(s)
jk@callisto:popolsku-1-najprostsze$ _
```

Najprostszy przykład ładowania tłumaczeń

```
int main(int argc, char* argv[ ] )
{
  QApplication   App(argc,argv);
  ...
  QTranslator   TlumaczPL;
  if  ( TlumaczPL.load("skromny_edytor_pl.qm","." ) ) {
      App.installTranslator(&TlumaczPL);
  } else  {
      cerr << "Plik 'skromny_edytor_pl.qm' nie zostal zaladowany" << endl;
  }
  ...
  return  App.exec( );
}
```

Można zainstalować wiele plików tłumaczeń. Tłumaczenia są wyszukiwane w odwrotnej kolejności, w jakiej zostały zainstalowane. Tak więc ostatnio zainstalowany plik tłumaczenia jest przeszukiwany jako pierwszy, a pierwszy zainstalowany plik tłumaczenia jest przeszukiwany jako ostatni. Wyszukiwanie zatrzymuje się, gdy tylko zostanie znalezione tłumaczenie zawierające pasujący ciąg. Tym ciągiem jest napis wpisany w wywołaniu metody `tr()`.

Przykład okienka aplikacji

```
#include <QMainWindow>
#include "ui_oknoglowne.h"

class OknoGlowneAplikacji: public QMainWindow, public Ui::OknoGlowne {
    Q_OBJECT
  public:
    OknoGlowneAplikacji();
    virtual void changeEvent(QEvent *event) override;

  public slots:
    void on_action_Open_triggered(bool checked);
    void on_action_Exit_triggered(bool checked);
    void on__wPrzycisk_Koniec_clicked();
    void on__wWyborJezyka_currentIndexChanged(int);
};
```

Konstrukcja nazw slotów umożliwia ich automatyczne łączenie z odpowiednimi sygnałami.

Ładowanie tłumaczenia

```
void OknoGlowneAplikacji::on__wWyborJezyka_currentIndexChanged(int Idx)
{

  static QTranslator *wTlumaczPL = new QTranslator();


    switch (Idx) {
        case 0: {
            qApp->removeTranslator(wTlumaczPL);
            _wNapis_Czas->setText(QDate::currentDate().toString());
            break;
        }
        case 1: {
            if (wTlumaczPL->load("pro/interapp_pl.qm",".") ) {
                qApp->installTranslator(wTlumaczPL);
                QLocale  Locale4PL(QLocale::Polish);
                _wNapis_Czas->setText(Locale4PL.toString(QDate::currentDate()));
            } else cerr << "!!! Plik \"interapp_pl.qm\""
                           " nie zostal zaladowany." << endl;
            break;
        }
    }
}
```

Przykład samodzielnej obsługi zmiany tłumaczeń. W tym przypadku zmiana ustawień regionalnych (ang. locale) jest niezbędne ze względu na konieczność dostosowania formatu wyświetlanej daty.

Ładowanie tłumaczenia

```
class OknoGlowneAplikacji: public QMainWindow, public Ui::OknoGlowne {
    Q_OBJECT
  public:
    OknoGlowneAplikacji();
    virtual void changeEvent(QEvent *event) override;

  public slots:
    ...
    void on__wWyborJezyka_currentIndexChanged(int);
};
```

```
void OknoGlowneAplikacji::changeEvent(QEvent *event)
{
    if ( event->type() == QEvent::LanguageChange ) {
        cout << "Zmiana" << endl;
        retranslateUi(this);
        return;
    }
    QMainWindow::changeEvent(event);
}
```

Zainstalowanie lub usunięcie `QTranslator` lub zmiana zainstalowanego `QTranslator` generuje zdarzenie `LanguageChange` dla instancji `QCoreApplication`. Instancja `QApplication` będzie propagować zdarzenie do wszystkich widżetów najwyższego poziomu, gdzie ponowna implementacja `changeEvent` może ponownie przetłumaczyć interfejs użytkownika, przekazując widoczne dla użytkownika ciągi znaków za pomocą funkcji `tr()` do odpowiednich ustawiaczy właściwości. Klasy interfejsu użytkownika generowane przez `Qt Designer` udostępniają funkcję `retranslateUi()`, którą można wywołać.

Ładowanie tłumaczenia

```
class OknoGlowneAplikacji: public QMainWindow, public Ui::OknoGlowne {
    Q_OBJECT
  public:
    OknoGlowneAplikacji();
    virtual void changeEvent(QEvent *event) override;

  public slots:
    ...
    void on__wWyborJezyka_currentIndexChanged(int);
};
```

```
void OknoGlowneAplikacji::changeEvent(QEvent *event)
{
    if ( event->type() == QEvent::LanguageChange ) {
        cout << "Zmiana" << endl;
        retranslateUi(this);
        return;
    }
    QMainWindow::changeEvent(event);
}
```

Zainstalowanie lub usunięcie `QTranslator` lub zmiana zainstalowanego `QTranslator` generuje zdarzenie `LanguageChange` dla instancji `QCoreApplication`. Instancja `QApplication` będzie propagować zdarzenie do wszystkich widżetów najwyższego poziomu, gdzie ponowna implementacja `changeEvent` może ponownie przetłumaczyć interfejs użytkownika, przekazując widoczne dla użytkownika ciągi znaków za pomocą funkcji `tr()` do odpowiednich ustawiaczy właściwości. Klasy interfejsu użytkownika generowane przez `Qt Designer` udostępniają funkcję `retranslateUi()`, którą można wywołać.

Metoda *tłumacząca*

```
void Ui_OknoGlowne::retranslateUi(QMainWindow *OknoGlowne)
 {
    OknoGlowne->setWindowTitle(
                    QApplication::translate("OknoGlowne","Application Window",nullptr));
    action_Open->setText(
                        QApplication::translate("OknoGlowne","&Open",nullptr));
#ifndef QT_NO_TOOLTIP
    action_Open->setToolTip(
                    QApplication::translate("OknoGlowne","Open a new file",nullptr));
#endif // QT_NO_TOOLTIP
    action_Exit->setText(QApplication::translate("OknoGlowne","&Exit",nullptr));
     ...
    _pLabel_Flag->setText(QString());
    _wNapis_Jezyk->setText(QApplication::translate("OknoGlowne","Language:",nullptr));
    _wEtykieta_Data->setText(QApplication::translate("OknoGlowne","Current date:",nullptr));
#ifndef QT_NO_STATUSTIP
    _wPrzycisk_Koniec->setStatusTip(
          QApplication::translate("OknoGlowne","Close and exit an application",nullptr));
#endif // QT_NO_STATUSTIP
    _wPrzycisk_Koniec->setText(QApplication::translate("OknoGlowne","Close",nullptr));
    menu_File->setTitle(QApplication::translate("OknoGlowne", "&File", nullptr));
 } // retranslateUi
```

Przykład metody *tłumaczącej* wygnerowanej przez Qt Designer.

Koniec prezentacji
Dziękuję za uwagę