Quick guide to add multilanguage support to your QML application

This article explains how to add multilingual support in QML application.

13 May 2012

Introduction

Add multilingual support to your application is important as adds more quality, but can be a key factor for its spread in the Store.

Prepare your app for Internationalization

In order to allow Qt Linguist to translate your phrases you have to use **tr** and **qsTr** functions.

Qt C++

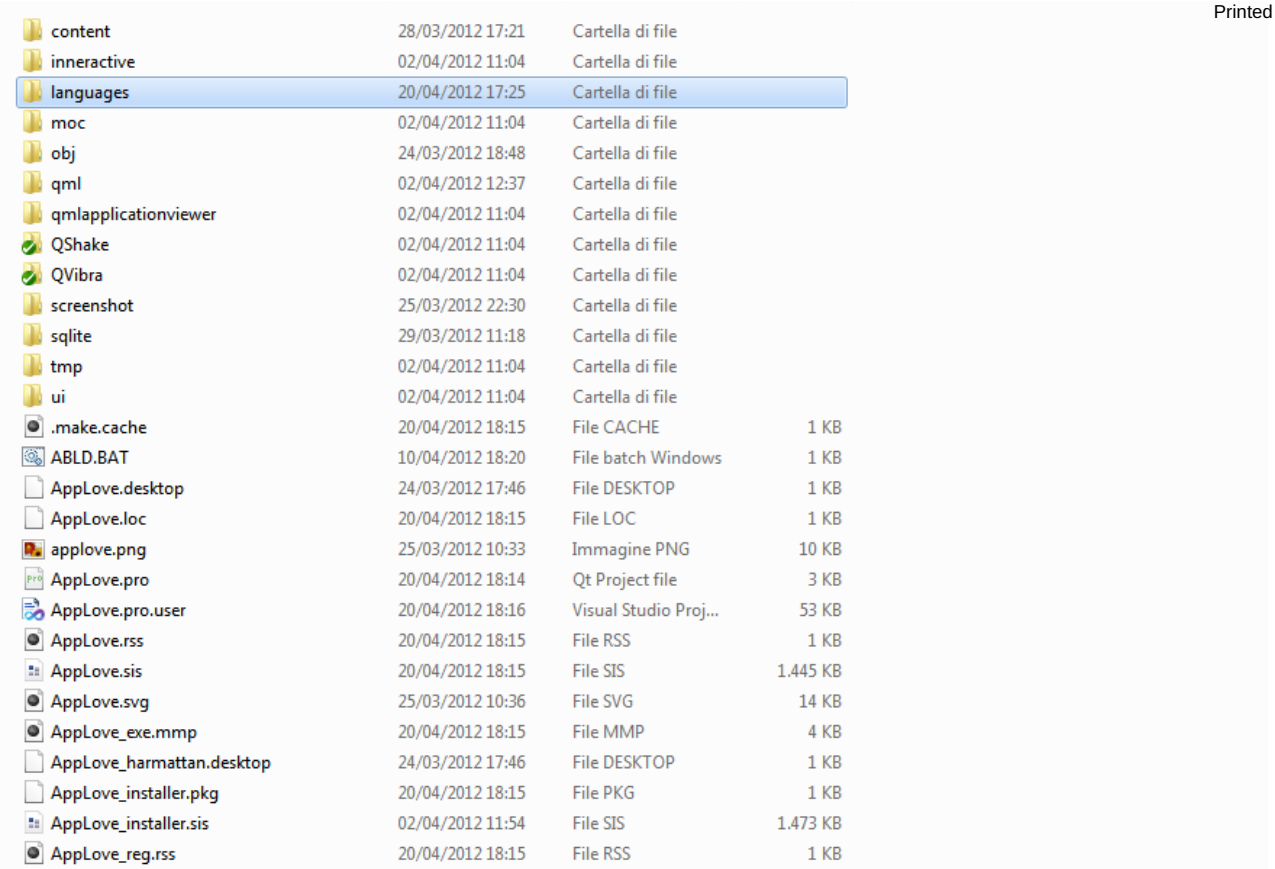
```
QMessageBox message;
message.setText(tr("No products available. Please try again later!"));
message.exec();
```

QML

```
Menu {
        id: colorMenu
        visualParent: window

        MenuLayout {

            MenuItem {
                text: qsTr("Restore purchased products");
                onClicked: {
                    iap_manager.restoreProducts();
                }
            }

            MenuItem {
                text: qsTr("Credits");
                onClicked: {
                    creditsDialog.open()
                }
            }
        }
}
```

On your directory project create the directory languages where we will put all languages files.

(C) Copyright Nokia 2013. All rights reserved.



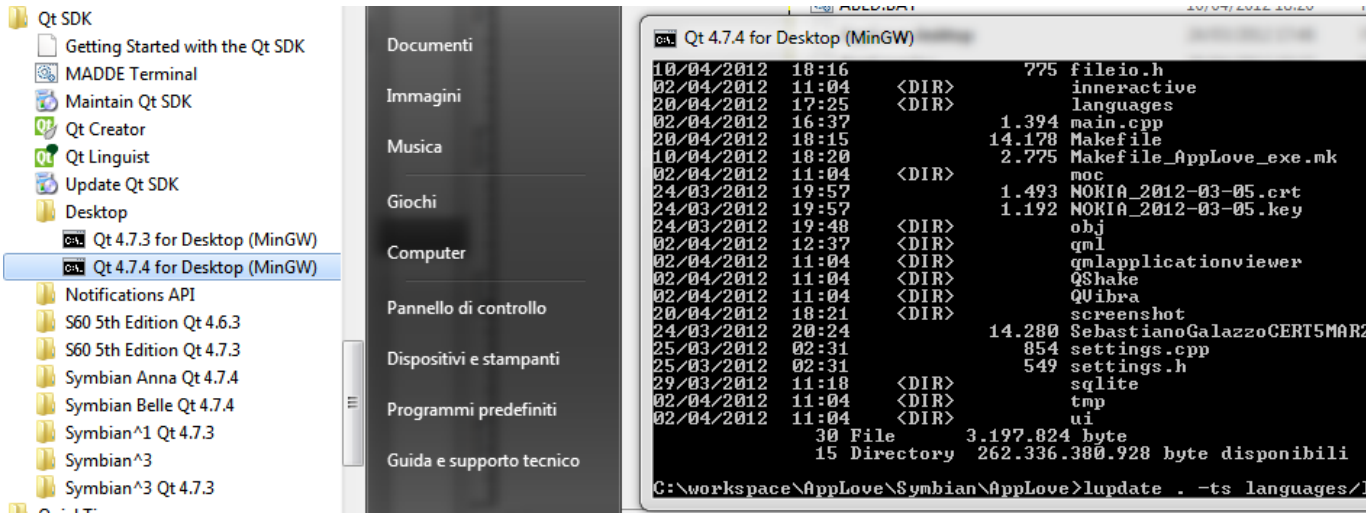
content	28/03/2012 17:21	Cartella di file	
inneractive	02/04/2012 11:04	Cartella di file	
languages	20/04/2012 17:25	Cartella di file	
moc	02/04/2012 11:04	Cartella di file	
obj	24/03/2012 18:48	Cartella di file	
qml	02/04/2012 12:37	Cartella di file	
qmlapplicationviewer	02/04/2012 11:04	Cartella di file	
QShake	02/04/2012 11:04	Cartella di file	
QVibra	02/04/2012 11:04	Cartella di file	
screenshot	25/03/2012 22:30	Cartella di file	
sqlite	29/03/2012 11:18	Cartella di file	
tmp	02/04/2012 11:04	Cartella di file	
ui	02/04/2012 11:04	Cartella di file	
.make.cache	20/04/2012 18:15	File CACHE	1 KB
ABLD.BAT	10/04/2012 18:20	File batch Windows	1 KB
AppLove.desktop	24/03/2012 17:46	File DESKTOP	1 KB
AppLove.loc	20/04/2012 18:15	File LOC	1 KB
applove.png	25/03/2012 10:33	Immagine PNG	10 KB
AppLove.pro	20/04/2012 18:14	Qt Project file	3 KB
AppLove.pro.user	20/04/2012 18:16	Visual Studio Proj...	53 KB
AppLove.rss	20/04/2012 18:15	File RSS	1 KB
AppLove.sis	20/04/2012 18:15	File SIS	1.445 KB
AppLove.svg	25/03/2012 10:36	File SVG	14 KB
AppLove_exe.mmp	20/04/2012 18:15	File MMP	4 KB
AppLove_harmattan.desktop	24/03/2012 17:46	File DESKTOP	1 KB
AppLove_installer.pkg	20/04/2012 18:15	File PKG	1 KB
AppLove_installer.sis	02/04/2012 11:54	File SIS	1.473 KB
AppLove_reg.rss	20/04/2012 18:15	File RSS	1 KB

lupdate

lupdate is a tool that create the .ts file, an XML file that map all the prhases enclosed by **tr** and **qsTr** function. Open the Qt shell, move to your directory project and run the following command for each language you are interested to:

```
lupdate . -ts languages/lang_it_IT.ts
lupdate . -ts languages/lang_es_ES.ts
lupdate . -ts languages/lang_pt_PT.ts
.....
```

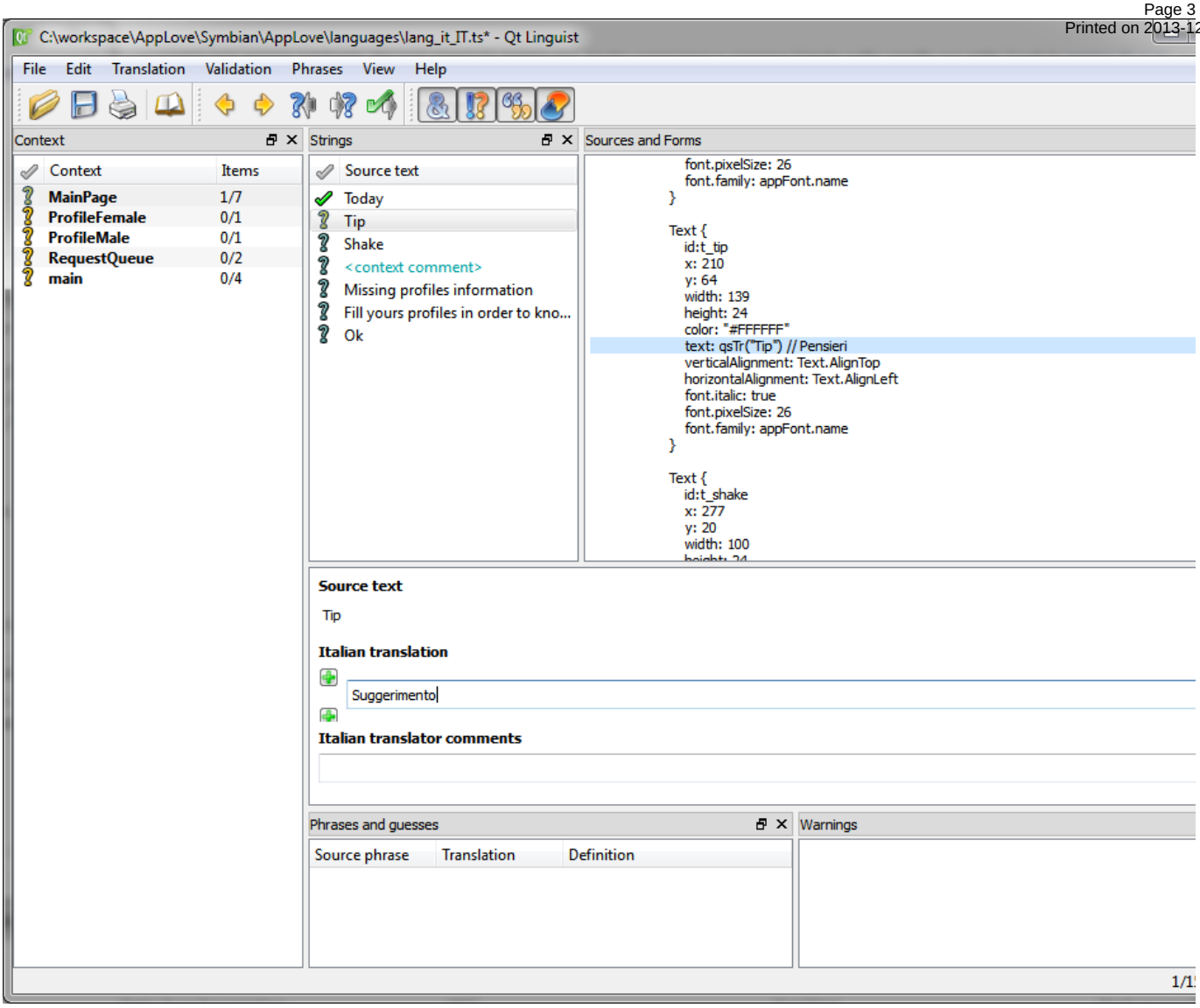
Although you can use any name, for ease of use and order, my suggestion is to keep the syntax described.



Qt Linguist

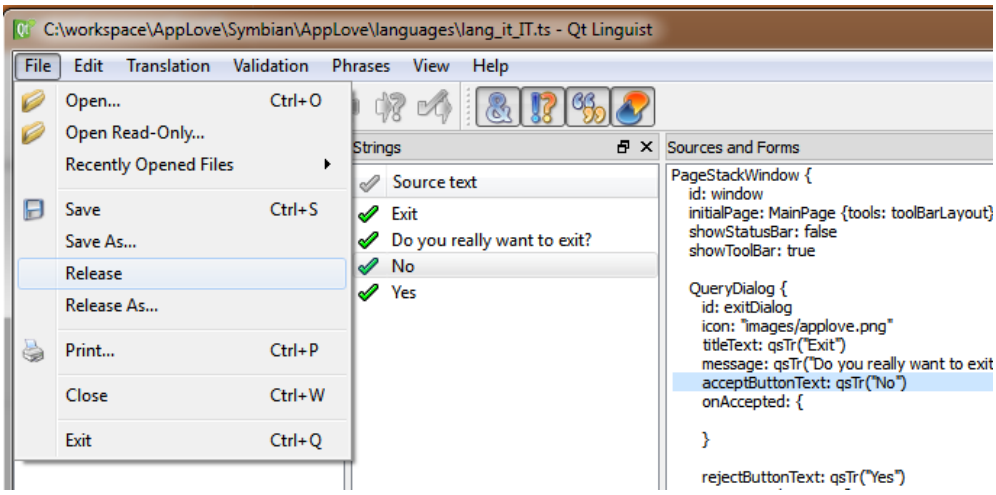
Move to the **languages** directory and open the .ts file you are interested to with **Qt Linguist**, for example **lang_it_IT.ts** and for each sentence add the translation.

(C) Copyright Nokia 2013. All rights reserved.



lrelease

Although the common way is to use the command line **lrelease** the fastest way is to use **Qt Linguist** clicking on : File -> Release

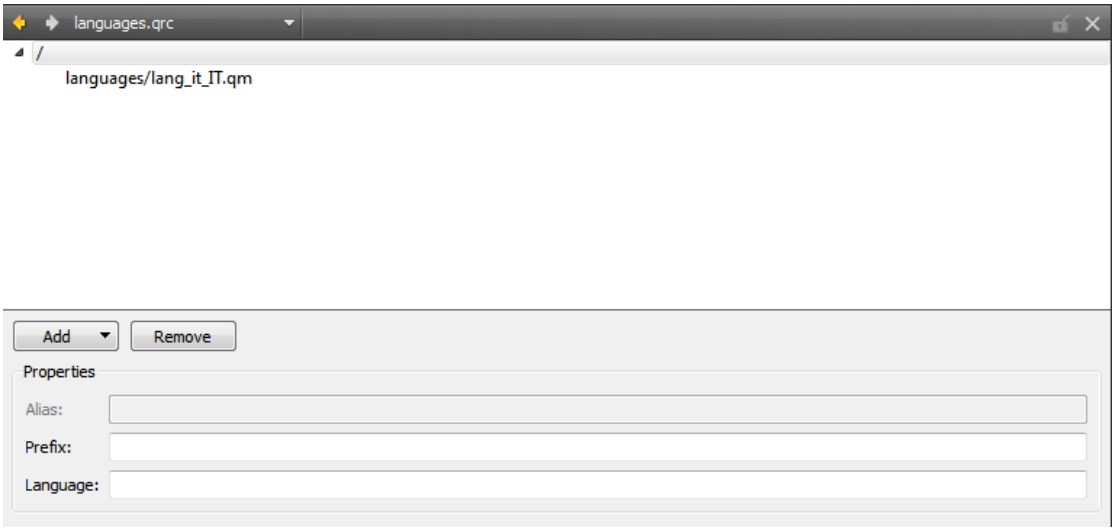


the .qm file

Now if all steps went well you should find the **lang_it_IT.qm** into the languages directory. This is the binary file you have to import into your application.

Add a new resource file to your project called **languages** and add all the **.qm** fille without using **prefix**

(C) Copyright Nokia 2013. All rights reserved.



main.cpp

Adding the following code lines your app will be able to load the correct language file based on the phone settings on witch your app will be installed.
You can test by changing language settings on your phone.
If a language file is not available for the current phone settings the original text will be used.

```
#include <QtGui/QApplication>
#include "qmlapplicationviewer.h"

#include <QTranslator>
#include <QTextCodec>
#include <QLocale>

Q_DECL_EXPORT int main(int argc, char *argv[])
{
    QScopedPointer<QApplication> app(createApplication(argc, argv));

    qDebug() << "local:" << QLocale::system().name();
    QString locale = QLocale::system().name();
    QString filename = QString("languages/lang_") + locale;

    static QTranslator translator;
    if( translator.load(filename, ":/") ){
        app->installTranslator(&translator);
        QTextCodec::setCodecForTr(QTextCodec::codecForName("utf8"));
        qDebug() << "Translation file loaded" << filename;
    } else
        qDebug() << "Translation file not loaded:" << filename;
 .....
```

Now you can compile, run your app and enjoy with the magic :-)

(C) Copyright Nokia 2013. All rights reserved.

 (C) Copyright Nokia 2013. All rights reserved.