个人资料 [管理]

一去、二三里

Qing 微博

博客等级：18
博客积分：78新
博客访问：407,918
关注人气：378
获赠金笔：0支
赠出金笔：0支
荣誉徽章：

相关博文

- Qt之界面出现、消失动画效果 一去、二三里
- Qt之QTreeView（一） 一去、二三里
- Qt之QTreeView（三） 一去、二三里
- CSS显示图片指定范围 古儿
- QTableWidget详解（样式、右键菜 一去、二三里
- Qt之打包发布（NSIS详解） 一去、二三里
- Qt之Windows开发移植 一去、二三里
- Qt之QTreeView（二） 一去、二三里
- QwtPlot之绘制统计图（三） 一去、二三里
- Qt之密码框不可全选、复制、粘贴 一去、二三里

更多>>

正文 字体大小：大 中 小

Qt之多语化

(2013-08-22 17:13:46) [编辑][删除] 转 载

标签： qt 多语化 qt多语化 qt动态切换语言 qtlinguist 分类： Qt

一个好的软件，只允许用户使用自己的本地语言是不够的，最好能让整个界面允许翻译才行。所以，多语化的需求必不可少，不止是为了当前使用，包括以后扩展都很方便。。。

多语化属于Qt高级中的一部分，很多人都在用，亦或者不会用，更甚者在滥用！今天在此一谈，说说那些我所了解的多语化。。。

对于绝大多数的应用程序，在main()中检测首选语言并进行加载是非常必要的！但在一些情况下，用户也需要动态的切换语言的功能，如果能够做到不重启应用程序而实现语言的动态切换，那么软件至少在切换语言方面的易用性可以说是接近完美的。

总结一下：

（1）动态切换语言，即：当需要改变语言的时候，不需要重启软件就可以实现不同语言之间的切换。

（2）在应用程序首次启动的时候，加载用户最后一次选择的语言。

要进行多语化的切换，比加载一个单一的翻译文件要稍微复杂一些，但也并不困难，需要做以下几件事：

（1）对用户可见的文本信息全部使用tr()进行封装

（2）提供用户可以用来切换语言的一种方法

（3）对于每一个窗口部件或者对话框，把它所有可翻译的字符串放在一个单独的函数中，当语言发生改变的时候调用此函数即可。

如果细心的朋友可以发现，我博客分享的代码中通常都会有一个公用的方法translateLanguage()，此函数就是用来切换语言的。

现在开始手把手教你动态切换！

（1）创建主界面、设置界面（切换语言一般是属于设置部分）

（2）每一个窗口部件或者对话框，把它所有可翻译的字符串放在一个单独的函数中，此处使用translateLanguage（）。

实现方式：根据选择不同下拉语言选项实现语言的动态切换！

完整源代码下载地址：http://download.csdn.net/detail/u011012932/5995043

主界面：

```
#include "main_widget.h"
#include "util.h"
MainWidget::MainWidget(QWidget *parent)
: QWidget(parent)
{
this->setMinimumSize(400, 300);

welcome_label = new QLabel();
setting_button = new QPushButton();
ok_button = new QPushButton();
cancel_button = new QPushButton();
setting_dialog = new SettingDialog();

QHBoxLayout *botton_layout = new QHBoxLayout();
botton_layout->addStretch();
botton_layout->addWidget(setting_button);
```

```
botton_layout->addWidget(ok_button);
botton_layout->addWidget(cancel_button);
botton_layout->setSpacing(10);
botton_layout->setContentsMargins(0, 0, 0, 0);

QVBoxLayout *main_layout = new QVBoxLayout();
main_layout->addWidget(welcome_label, 0, Qt::AlignCenter);
main_layout->addLayout(botton_layout);
main_layout->setSpacing(10);
main_layout->setContentsMargins(10, 10, 10, 10);

this->setLayout(main_layout);
this->translateLanguage();

connect(setting_button, &QPushButton::clicked, this, &MainWidget::showSetting);
connect(setting_dialog, &SettingDialog::switchLanguage, this, &MainWidget::switchLanguage);
}

MainWidget::~MainWidget()
{

}

void MainWidget::translateLanguage()
{
this->setWindowTitle(tr("main widget"));
welcome_label->setText(tr("welcome to Qt") + QString("26197884"));
setting_button->setText(tr("setting"));
ok_button->setText(tr("ok"));
cancel_button->setText(tr("cancel"));
}

void MainWidget::setLanguage(LANGUAGE current_language)
{
this->current_language = current_language;
}

void MainWidget::setTranslator(QTranslator* translator)
{
this->translator = translator;
}

void MainWidget::switchLanguage(int index)
{
LANGUAGE language;
QString language_qm;
switch(index)
{
case UI_ZH:
language = UI_ZH;
language_qm = QString(":/qm/main_widget_zh");
break;

case UI_EN:
language = UI_EN;
language_qm = QString(":/qm/main_widget_en");
break;

default:
language = UI_ZH;
language_qm = QString(":/qm/main_widget_zh");
}

if(current_language != language)
{
current_language = language;
```

```
translator->load(language_qm);
setting_dialog->translateLanguage();
this->translateLanguage();

Util::writeInit(QString("./user.ini"), QString("language"), QString::number(language, 10));
}
}

void MainWidget::showSetting()
{
setting_dialog->loadConfig();
setting_dialog->exec();
}
```

设置界面：

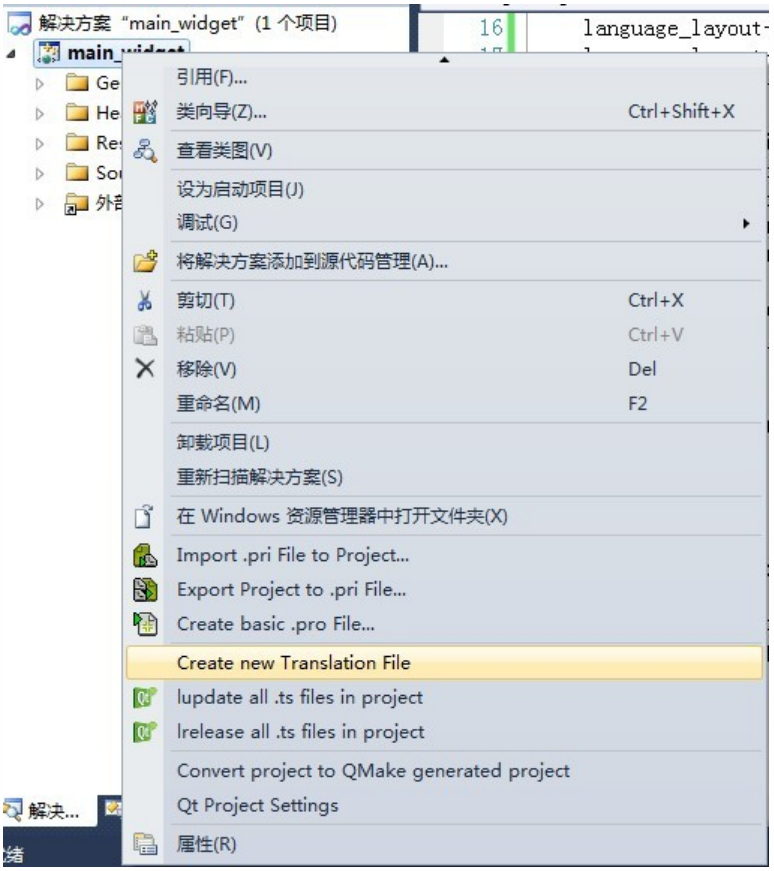
```
#include "setting_dialog.h"
#include "util.h"
SettingDialog::SettingDialog(QWidget *parent)
: QDialog(parent)
{
this->setMinimumSize(300, 200);
this->setWindowFlags(windowFlags()&~Qt::WindowContextHelpButtonHint);

language_label = new QLabel();
language_combo_box = new QComboBox();
info_label = new QLabel();

language_combo_box->addItem("chinese", UI_ZH);
language_combo_box->addItem("english", UI_EN);

QHBoxLayout *language_layout = new QHBoxLayout();
language_layout->addStretch();
language_layout->addWidget(language_label);
language_layout->addWidget(language_combo_box);
language_layout->addStretch();
language_layout->setSpacing(5);
language_layout->setContentsMargins(0, 0, 0, 0);

QVBoxLayout *main_layout = new QVBoxLayout();
main_layout->addWidget(info_label, 0, Qt::AlignCenter);
main_layout->addLayout(language_layout);
main_layout->setSpacing(10);
main_layout->setContentsMargins(10, 10, 10, 10);

this->setLayout(main_layout);
this->translateLanguage();

connect(language_combo_box, static_cast(&QComboBox::currentIndexChanged), this,
&SettingDialog::switchLanguage);
}

SettingDialog::~SettingDialog()
{

}

void SettingDialog::translateLanguage()
{
this->setWindowTitle(tr("setting dialog"));
info_label->setText(tr("no brothers no programming"));
language_label->setText(tr("language"));
language_combo_box->setItemText(UI_ZH, tr("chinese"));
language_combo_box->setItemText(UI_EN, tr("english"));
}
```

```
void SettingDialog::loadConfig()
{
QString language_value;
QString language_suffix = QString("zh");
LANGUAGE language = UI_ZH;
bool is_read = Util::readInit(QString("./user.ini"), QString("language"), language_value);
if(is_read)
{
language = (LANGUAGE)language_value.toInt();
if(language == UI_EN)
{
language_suffix = QString("en");
}
}

int count = language_combo_box->count();
    for(int i=0; i
{
        if(language == i)
{
            language_combo_box->setCurrentIndex(language);
            break;
        }
    }
}
```

主方法：

```
#include "main_widget.h"
#include
#include
#include "util.h"

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

QString language_value;
QString language_suffix = QString("zh");
LANGUAGE language = UI_ZH;
bool is_read = Util::readInit(QString("./user.ini"), QString("language"), language_value);
if(is_read)
{
language = (LANGUAGE)language_value.toInt();
if(language == UI_EN)
{
language_suffix = QString("en");
}
}

QTranslator translator;
translator.load(QString(":/qm/main_widget_") + language_suffix);
app.installTranslator(&translator);

MainWidget main_widget;
main_widget.setTranslator(&translator);
main_widget.setLanguage(language);
main_widget.show();
return app.exec();
}
```

完成代码编写之后，创建多语化文件：



然后进行翻译：

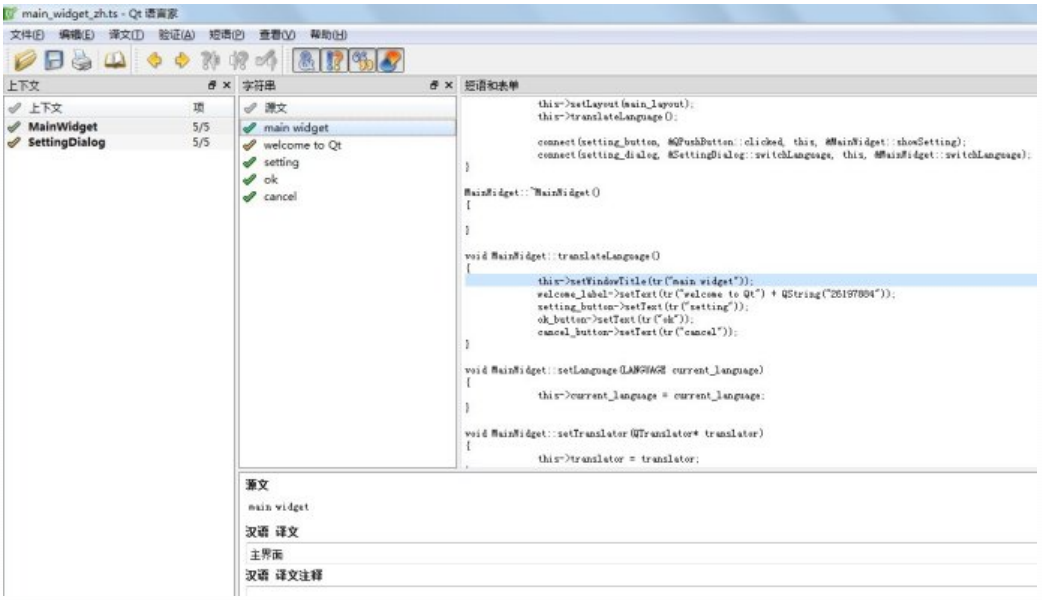
翻译一个含有tr()调用的Qt应用程序需要以下三步：

（1）运行ludate，从应用程序的源代码中提取所有用户可见的字符串。

（2）使用Qt Linguist翻译该应用程序。

（3）运行lrelease，生成二进制的.qm文件，应用程序可以使用QTranslator加载这个文件。

如果翻译有改动，可以多次执行此过程！

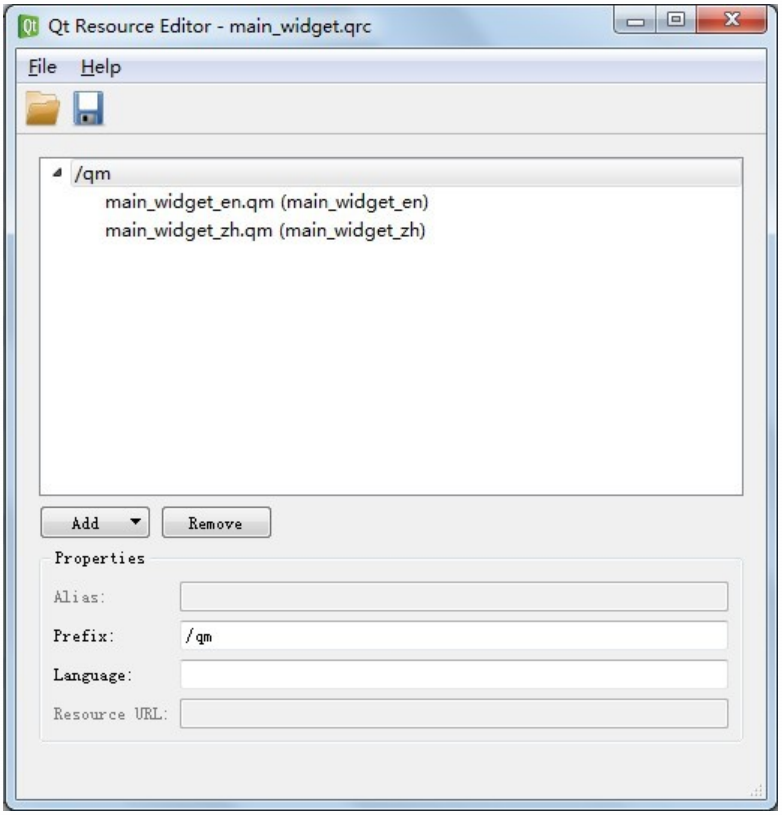


然后进行发布，重新编译后会出现相应的.qm文件，将其添加至资源文件当中（也可以使用绝对路径，但不建议）。通过Qt的资源系统将翻译文件嵌入至可执行文件来改变应用程序，可以提供很多方便！

（1）减少了作为产品发布的一部分文件数量

（2）避免了翻译文件丢失或者被意外删除的风险（若翻译文件不存在了，那么翻译工作定然完成不了）

注意：使用资源文件时，通常指明为":/qm"作为翻译文件的路径，冒号说明：指向资源文件的路径与文件系统中指向文件的路径是不同的。

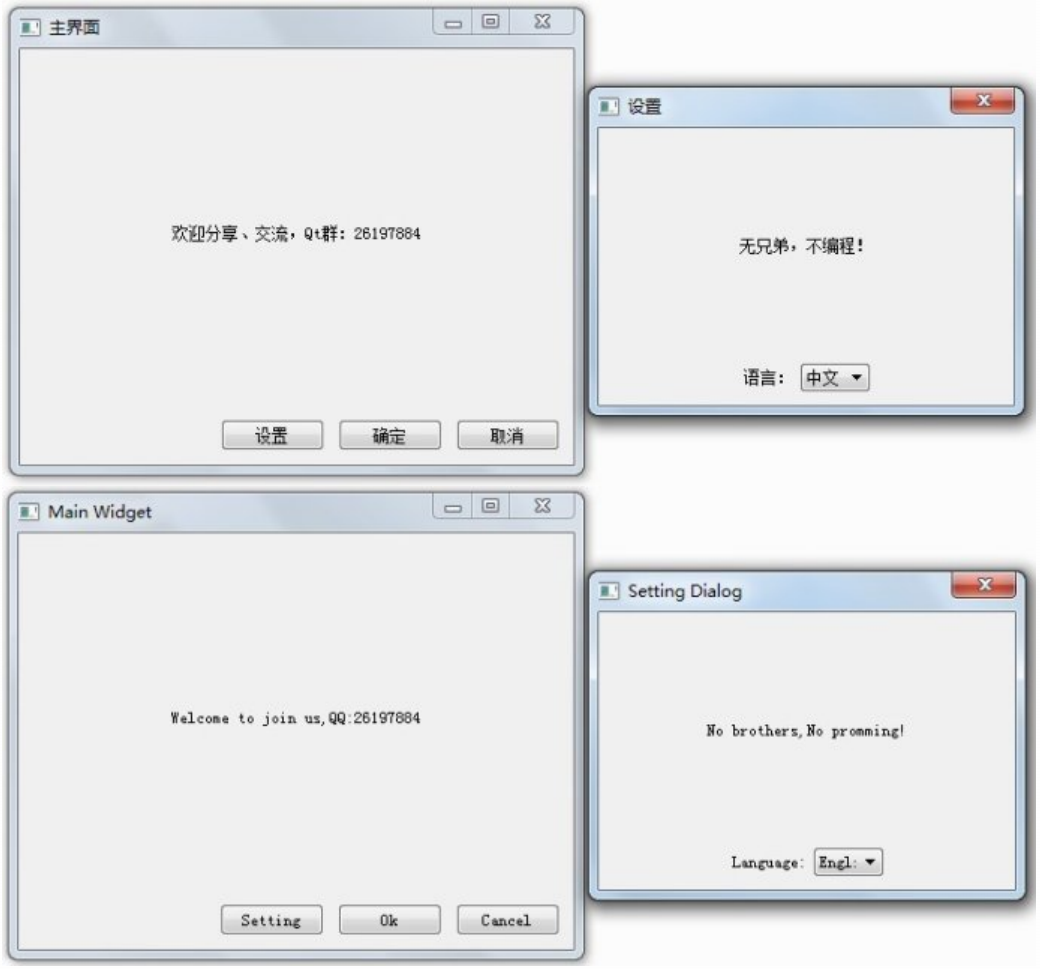


大功告成，可以随意切换语言！（这里举例中英文切换、如果有需求可以自己扩展）。

说明：这些翻译文件都是以.ts为扩展名，使用的格式都是简单的XML，并且没有像QTranslator理解的二进制.qm文件那么紧凑。lupdate的工作是重新提取tr()包含的内容，更新到.ts文件当中。lrelease的则是把可读的.ts文件转换成高效率的机器可理解的.qm文件。.ts的名字代表“翻译源”（translation source），而.qm代表“Qt消息”(Qt message)。

每一次增加tr()或者修改tr()中内容之后，需要重写执行lupdate提取新的tr()，提取之后再进行编辑，完成之后还要再执行lrelease来更新.qm文件，结束之后重新编译即可！

效果如下：



注意：强调一下，请不要滥用tr()，不要包含中文在其中！如果你是在想在字符串里使用中文，请使用QString()，建议QStringLiteral()。

注：

技术在于交流、沟通，转载请注明出处并保持作品的完整性。

作者：`\☆奋斗ing❤孩子` 原文：http://blog.sina.com.cn/s/blog_a6fb6cc90101f89v.html。

喜欢

分享：

前一篇：如何将网页转化为PDF
后一篇：Qt之设置窗口背景

评论 重要提示：警惕虚假中奖信息 [发评论]

zddblog

学习了~

2013-8-22 20:45 回复(1)

luckstar

写的真好，以后会借鉴的，呵呵。

2013-8-23 09:58 回复(1)

str999_cn

好文章！调理清楚，代码清晰！

2013-8-23 15:50 回复(0)

str999_cn

条理清楚。。。。不是调理。。。哈哈

2013-8-23 15:51 回复(1)

用户2116122001

相当的赞啊，顶起来！

2013-8-27 12:36 回复(1)

阿灿

前段时间也做了国际化，但是一直没实现动态切换语言，看了楼主的文章，顿时喜大奔普

2013-10-23 17:28 回复(1)

迷路人1986

我想问的是你的所有text都是在translateLanguage这个里面定义，要是我的QLabel的内容会动态变化，你这个又如何翻译呢。

2014-4-13 10:07 回复(1)

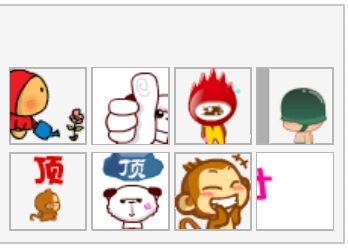
jacky_ije

主工程的依赖工程如何进行多语言化？

2014-4-21 09:46 回复(1)

发评论

一去、二三里：



分享到微博 新

匿名评论

验证码： 请点击后输入验证码 收听验证码

发评论

以上网友发言只代表其个人观点，不代表新浪网的观点或立场。

〈 前一篇
如何将网页转化为PDF

后一篇 〉
Qt之设置窗口背景