# Qt 之界面数据存储与获取

# 简述

在GUI开发中，往往需要在界面中存储一些有用的数据，这些数据可以来自配置文件、注册表、数据库、或者是Server。

无论来自哪里，这些数据对于用户来说都是至关重要的，它们在交互过程中大部分都会被用到，例如：单击一个用户头像，显示该用户的详细信息（等级、昵称、姓名、个人说明）。

# 常见接口

Qt中，可以通过绝大部分已有的接口来存储数据、获取数据。例如：

* 存储数据
  + setData()
  + setItemData()
  + setUserData()
* 获取数据：
  + data()
  + itemData()
  + userData()

常用的基本就这些，当然，还有其他的一些接口。。。

# 数据源

为了便于演示，我们定义两个数据源：结构体User、枚举LANGUAGE。

*// 用户信息*

struct User : QObjectUserData {

int nID; *// ID*

QString strName; *// 用户名*

};

*// 语言*

typedef enum{

UI\_ZH, *// 中文*

UI\_EN *// 英文*

} LANGUAGE;

Q\_DECLARE\_METATYPE(User)

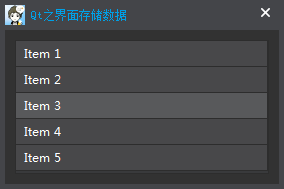
Q\_DECLARE\_METATYPE(LANGUAGE)

其中，User定义为QObjectUserData类型，为了后面setUserData()和userData()使用。如果不使用这两个接口，则不需要定义为QObjectUserData。

绝大部分存储用户数据的接口都使用的是QVariant，也就是我们常说的“万能变量”。对于自定义数据类型，如果要使用QVariant，就必须使用Q\_DECLARE\_METATYPE注册。

# setData()和data()

创建一个QListWidget列表，添加5个Item项，然后给每一个都存储属于自己的数据。



## 单独存储

通过setData()，我们可以单独存储用户数据，使用Qt::UserRole、Qt::UserRole + 1……

QListWidget \*pListWidget = new QListWidget(this);

int i = 0;

do {

++i;

QListWidgetItem \*pItem = new QListWidgetItem(pListWidget);

pItem->setData(Qt::UserRole, i); *// 用户数据*

pItem->setData(Qt::UserRole + 1, QString("Qter %1").arg(i)); *// 用户数据*

pItem->setText(QString("Item %1").arg(i)); *// 文本*

pListWidget->addItem(pItem);

} while (i < 5);

*// 连接信号槽*

connect(pListWidget, &QListWidget::itemClicked, this, &MainWindow::onItemClicked);

槽函数，获取用户数据，执行相应操作。

void onItemClicked(QListWidgetItem \*item) {

int nID = item->data(Qt::UserRole).toInt(); // 获取用户数据

QString strName = item->data(Qt::UserRole + 1).toString(); // 获取用户数据

qDebug() << "ID : " << nID;

qDebug() << "Name : " << strName;

}

## 整体存储

也可以通过setData()进行整体存储，这时候只需要使用Qt::UserRole即可，把用户数据当成一个结构体来存储。

QListWidget \*pListWidget = new QListWidget(this);

int i = 0;

do {

++i;

QListWidgetItem \*pItem = new QListWidgetItem(pListWidget);

User user;

user.nID = i;

user.strName = QString("Qter %1").arg(i);

pItem->setData(Qt::UserRole, QVariant::fromValue(user)); *// 设置用户数据*

pItem->setText(QString("Item %1").arg(i));

pListWidget->addItem(pItem);

} while (i < 5);

槽函数，获取用户数据，执行相应操作。

void onItemClicked(QListWidgetItem \*item) {

QVariant variant = item->data(Qt::UserRole); // 获取用户数据

User user = variant.value<User>();

qDebug() << "ID : " << user.nID;

qDebug() << "Name : " << user.strName;

}

无论那种方式都可以，如果在信号和槽的传递过程中，参数过多，建议使用“整体存储”方式。

# setItemData()和itemData()

在[Qt之国际化](http://blog.csdn.net/liang19890820/article/details/50276673)一节中，分享了多语言的切换，下面就以此为例：

QComboBox \*pComboBox = new QComboBox(this);

pComboBox->addItem("Chinese");

pComboBox->addItem("English");

pComboBox->setItemData(0, QVariant::fromValue(UI\_ZH)); *// 设置用户数据*

pComboBox->setItemData(1, QVariant::fromValue(UI\_EN)); *// 设置用户数据*

*// 连接信号槽*

connect(pComboBox, static\_cast<void (QComboBox::\*)(int)>(&QComboBox::currentIndexChanged), this, &MainWindow::onCurrentIndexChanged);

槽函数，获取用户数据，执行相应操作。

void onCurrentIndexChanged(int index) {

*// 获取发送者*

QObject \*pObject = this->sender();

QComboBox \*pComboBox = qobject\_cast<QComboBox \*>(pObject);

*// 获取用户数据*

QVariant variant = pComboBox->itemData(index);

LANGUAGE language = variant.value<LANGUAGE>();

qDebug() << "Language : " << language;

}

# setUserData()和userData()

通过setUserData()，我们可以存储用户数据。

**注意：**用户数据需要被定义为QObjectUserData类型。

QPushButton \*pButton = new QPushButton(this);

pButton->setText("Qter");

*// 用户数据*

User \*pUser = new User();

pUser->nID = 1;

pUser->strName = "Qter";

pButton->setUserData(Qt::UserRole, pUser); *// 设置用户数据*

*// 连接信号槽*

connect(pButton, &QPushButton::clicked, this, &MainWindow::onClicked);

槽函数，获取用户数据，执行相应操作。

void onClicked() {

*// 获取发送者*

QObject \*pObject = this->sender();

QPushButton \*pButton = qobject\_cast<QPushButton \*>(pObject);

*// 获取用户数据*

User \*pUser = (User \*)(pButton->userData(Qt::UserRole));

qDebug() << "ID : " << pUser->nID;

qDebug() << "Name : " << pUser->strName;

}

通过userData()可以获取QObjectUserData数据，然后转换成我们需要的类型User。

# 自定义数据

用现有的接口固然方便，有时对于复杂的功能，我们也不得不自定义。

举一个简单的例子：

#include <QPushButton>

class PushButton : public QPushButton

{

Q\_OBJECT

public:

explicit PushButton(QWidget \*parent = 0)

: QPushButton (parent),

m\_nID(-1),

m\_strName("")

{

}

void setID(int id) {

m\_nID = id;

}

void setName(QString name) {

m\_strName = name;

}

int id() const {

return m\_nID;

}

QString name() const {

return m\_strName;

}

private:

int m\_nID;

QString m\_strName;

};

我们定义了一个按钮QPushButton，可以通过setID()、setName()来设置ID和名称信息，如果要获取信息，则可以调用id()、name()函数。

使用很简单：

PushButton \*pButton = new PushButton(this);

pButton->setText("Qter");

*// 设置自定义数据*

pButton->setID(1);

pButton->setName("Qter");

*// 连接信号槽*

connect(pButton, &QPushButton::clicked, this, &MainWindow::onClicked);

槽函数，获取用户数据，执行相应操作。

void onClicked() {

*// 获取发送者*

QObject \*pObject = this->sender();

PushButton \*pButton = qobject\_cast<PushButton \*>(pObject);

*// 获取用户数据*

qDebug() << "ID : " << pButton->id();

qDebug() << "Name : " << pButton->name();

}

基本的数据存储与获取方式就这些，以后会经常用到，Good luck。