如何通过C++实现动态加载qml文件，扩展的话可以实现一个简单的插件系统.涉及类QQuickView,QQuickWindow以及QQmlEngine。QQuickView 提供了一个窗体用于显示UIQQmlEngine 提供QML运行的环境QQuickWindow 显示窗体，以及对item对象的管理及用户交互

1. 新建一个QQuick2工程，添加一个QMLDynLoader类，实现动态加载qml文件

class QMLDynLoader : public QObject{

Q\_OBJECT

public:

QMLDynLoader();

void setEngine( QQmlApplicationEngine\* engine );

void loadQml(const QUrl& qmlFile );

private:

QQuickView\* mView\_; //显示的view

QQuickWindow\* mParentWindow\_; //框架的父窗体

QQmlApplicationEngine\* mEngine\_;

signals:

void sError(const QString errorMsg);

};

//======================cpp=====================

QMLDynLoader::QMLDynLoader()

{

this->mEngine\_ = NULL;

this->mView\_ = NULL;

this->mParentWindow\_ = NULL;

}

void QMLDynLoader::setEngine(QQmlApplicationEngine \*engine)

{

this->mEngine\_ = engine ;

if( this->mEngine\_ == NULL )

{

emit this->sError("初始化qml引擎失败");

return ;

}

this->mEngine\_->clearComponentCache();

if( this->mView\_ ) return ; //获取主窗体

if( this->mEngine\_->rootObjects().count( ) >= 0 )

{

this->mParentWindow\_ =\

qobject\_cast( this->mEngine\_->rootObjects().value(0));

if(this->mParentWindow\_ == NULL )

{

emit this->sError("无法独立加载QML，必须先加载主窗体.");

return ;

}

}

this->mView\_ = new QQuickView(\

this->mEngine\_ , \

this->mParentWindow\_ );

}

void QMLDynLoader::loadQml(const QUrl& qmlFile)

{

if(this->mView\_)

{

this->mView\_->setSource(qmlFile);

this->mView\_->show();

}

}

//=========================== 调用==============================

QMLDynLoader\* loader = new QMLDynLoader();

loader.setEngine(xxx);

loader.loadQml(xxx)

2.创建一个插件文件

text.qml

 import QtQuick 2.0

Rectangle{

id : root

width: 200 ;

height:200

color:"green"

property alias lbtext : lb.text

Rectangle {

width: 100

height: 100

border.color: "#99ccff"

border.width: 4

radius: 10

Text{

id:lb

text:"andy"

anchors.centerIn: parent

color:"red"

}

MouseArea{

anchors.fill: parent

drag.target: parent

drag.axis: drag.XAndYAxis

drag.maximumX: root.width - parent.width

drag.maximumY: root.height - parent.height

}

}

}

3.main.cpp 运行，

调用

QApplication app(argc, argv);

QQmlApplicationEngine engine; engine.load(QUrl(QStringLiteral("qrc:///main.qml")));

QMLDynLoader\* loader = new QMLDynLoader();

loader->setEngine( &engine );

loader->loadQml(QUrl::fromLocalFile("c:\\text.qml"));

return app.exec();

 4.其他文件=>

main.qml

Window {

Rectangle{

anchors.fill: parent color:"#404040"

Text{

id:lb

text:"andy"

anchors.centerIn: parent

}

}

}

View Code