# 使用Cpp17标准

## 一简单使用，cpp调用qml

### 1.需要在pro文件在添加配置：CONFIG += c++17

### 2.在qml文件在实例化一个Label对象，设置它的文本，并且给它添加一个函数利用返回它的文本内容

|  |
| --- |
| //main.qml  import QtQuick 2.15  import QtQuick.Window 2.15  import QtQuick.Controls 2.15  Window {  width: 640  height: 480  visible: true  title: *qsTr*("Qml+ Cpp")  //用cpp17标准需要修改pro文件  Label{  objectName: "labelForCpp"  font.pointSize: 18  text:"qml with cpp"  function *getText*(){  return *text*  }  }  } |

### 3.在cpp端获取qml Label的内容

|  |  |
| --- | --- |
| //main.cpp  #include <QGuiApplication>  #include <QQmlApplicationEngine>  #include<QMetaObject>  #include<QDebug>  int **main**(int **argc**, char \***argv**[])  {  #if QT\_VERSION < QT\_VERSION\_CHECK(6, 0, 0)  QCoreApplication::setAttribute(Qt::AA\_EnableHighDpiScaling);  #endif  QGuiApplication **app**(*argc*, *argv*);  QQmlApplicationEngine **engine**;  const QUrl **url**(QStringLiteral("qrc:/main.qml"));  QObject::connect(&engine, &QQmlApplicationEngine::objectCreated,  &app, [url](QObject \***obj**, const QUrl &**objUrl**) {  if (!obj && url == objUrl)  QCoreApplication::exit(-1);  }, Qt::QueuedConnection);  engine.load(url);  //在这里调用qml的getText函数  auto **root** = engine.rootObjects();  auto **labelQml** = root.first()->findChild<QObject\*>("labelForCpp");  // QString text = labelQml->getText();  QVariant **ret**;  QMetaObject::invokeMethod(*labelQml*,"getText",Q\_RETURN\_ARG(QVariant,*ret*));  qDebug()<<ret;//可以调用toString()方法来变为Qstring  return app.exec();  } |  |

## 二、qml中的方法需要传入参数

### 1.qml文件

|  |
| --- |
| //main.qml  import QtQuick 2.15  import QtQuick.Window 2.15  import QtQuick.Controls 2.15  Window {  width: 640  height: 480  visible: true  title: *qsTr*("Qml+ Cpp")  //用cpp17标准需要修改pro文件  Label{  objectName: "labelForCpp"  font.pointSize: 18  text:"qml with cpp"  function *getText*(info){  return *text* + " " +*info*  }  }  } |

### 2.Cpp文件中需要使用Q\_ARG()宏

|  |  |
| --- | --- |
| #include <QGuiApplication>  #include <QQmlApplicationEngine>  #include<QMetaObject>  #include<QDebug>  int **main**(int **argc**, char \***argv**[])  {  #if QT\_VERSION < QT\_VERSION\_CHECK(6, 0, 0)  QCoreApplication::setAttribute(Qt::AA\_EnableHighDpiScaling);  #endif  QGuiApplication **app**(*argc*, *argv*);  QQmlApplicationEngine **engine**;  const QUrl **url**(QStringLiteral("qrc:/main.qml"));  QObject::connect(&engine, &QQmlApplicationEngine::objectCreated,  &app, [url](QObject \***obj**, const QUrl &**objUrl**) {  if (!obj && url == objUrl)  QCoreApplication::exit(-1);  }, Qt::QueuedConnection);  engine.load(url);  //在这里调用qml的getText函数  auto **root** = engine.rootObjects();  auto **labelQml** = root.first()->findChild<QObject\*>("labelForCpp");  // QString text = labelQml->getText();  QVariant **ret**;  QMetaObject::invokeMethod(*labelQml*,"getText",Q\_RETURN\_ARG(QVariant,*ret*),Q\_ARG(QVariant,"kenny"));  qDebug()<<ret.toString();  return app.exec();  } |  |

## 三cpp创建自定义ListView的模型

|  |  |
| --- | --- |
| //customlistmodel.h  #ifndef \_CUSTOM\_LIST\_MODEL\_H  #define \_CUSTOM\_LIST\_MODEL\_H  #include <QtCore>  #include<QList>  #include<QtQuick>  class MyListItem:public QObject{  Q\_OBJECT  Q\_PROPERTY(QString name READ getName WRITE setName NOTIFY nameChanged)  Q\_PROPERTY(qint32 value READ getValue WRITE setValue NOTIFY valueChanged)  public:  using QObject::QObject;  const QString &getName() const;  void setName(const QString &newName);  qint32 getValue() const;  void setValue(qint32 newValue);  signals:  void nameChanged();  void valueChanged();  private:  QString name;  qint32 value;  };  class CustomListModel: public QAbstractListModel{  Q\_OBJECT  enum Role{  Name = Qt::DisplayRole + 1,  Value  };  Q\_PROPERTY(QQmlListProperty<MyListItem> datas READ datas())  public:  using QAbstractListModel:: QAbstractListModel;  Q\_INVOKABLE void append(QJsonObject item);  int rowCount(const QModelIndex &parent = QModelIndex()) const override;  QHash<int, QByteArray> roleNames() const override;  QVariant data(const QModelIndex &index, int role = Qt::DisplayRole) const override;  QQmlListProperty<MyListItem> datas();  static void AppendFunction(QQmlListProperty<MyListItem> \*property, MyListItem \*value);  static MyListItem\* AtFunction(QQmlListProperty<MyListItem> \*property, int index);  static void ClearFunction(QQmlListProperty<MyListItem> \*property);  static int CountFunction(QQmlListProperty<MyListItem> \*property);  static void ReplaceFunction(QQmlListProperty<MyListItem> \*property,int index, MyListItem \*value);  static void RemoveLastFunction(QQmlListProperty<MyListItem> \*property);  private:  mutable QHash<int, QByteArray> roles;  QList<MyListItem\*> items;  };  #endif | //customlistmodel.cpp  #include "customlistmodel.h"  void CustomListModel::**append**(QJsonObject **item**)  {  auto **tmp** = new MyListItem(this);  if(!tmp) return;  //写法一  tmp->setProperty("name",item["name"].toString());//注意这个方法使用的地方是有讲究的，放错地方程序崩溃  tmp->setProperty("value",item["value"].toInt()); //这个函数其实就是调用下面的setXXX方法  //写法二  // tmp->setName(item["name"].toString());  // tmp->setValue(item["value"].toInt());  items.append(*tmp*);  endResetModel();//添加数据后必须调用这个函数  }  int CustomListModel::***rowCount***(const QModelIndex &**parent**) const  {  return items.count();  }  QHash<int, QByteArray> CustomListModel::***roleNames***() const  {  if(roles.isEmpty()){  roles.insert(Role::Name,"name");  roles.insert(Role::Value,"value");  }  return roles;  }  QVariant CustomListModel::***data***(const QModelIndex &**index**, int **role**) const  {  if(!index.isValid())  return QVariant();  if(role == Role::Name)  return QVariant(items[index.row()]->getName());  else if(role == Role::Value){  return QVariant(items[index.row()]->getValue());  }  return QVariant();  }  QQmlListProperty<MyListItem> CustomListModel::**datas**()  {  return QQmlListProperty<MyListItem>(this,&items,&AppendFunction,&CountFunction,&AtFunction,&ClearFunction  ,&ReplaceFunction,&RemoveLastFunction);  }  void CustomListModel::**AppendFunction**(QQmlListProperty<MyListItem> \***property**, MyListItem \***value**)  {  auto **items** = static\_cast<QList<MyListItem\*> \*>(property->data);  items->append(*value*);  }  MyListItem \*CustomListModel::**AtFunction**(QQmlListProperty<MyListItem> \***property**, int **index**)  {  auto **items** = static\_cast<QList<MyListItem\*> \*>(property->data);  return items->at(index);  }  void CustomListModel::**ClearFunction**(QQmlListProperty<MyListItem> \***property**)  {  auto **items** = static\_cast<QList<MyListItem\*> \*>(property->data);  items->clear();  }  int CustomListModel::**CountFunction**(QQmlListProperty<MyListItem> \***property**)  {  auto **items** = static\_cast<QList<MyListItem\*> \*>(property->data);  return items->count();  }  void CustomListModel::**ReplaceFunction**(QQmlListProperty<MyListItem> \***property**, int **index**, MyListItem \***value**)  {  auto **items** = static\_cast<QList<MyListItem\*> \*>(property->data);  items->replace(index,*value*);  }  void CustomListModel::**RemoveLastFunction**(QQmlListProperty<MyListItem> \***property**)  {  auto **items** = static\_cast<QList<MyListItem\*> \*>(property->data);  items->pop\_back();  }  const QString &MyListItem::**getName**() const  {  return name;  }  void MyListItem::**setName**(const QString &**newName**)  {  //最快不能使用setProperty方法，会崩溃  name = newName;  emit nameChanged();  }  qint32 MyListItem::**getValue**() const  {  return value;  }  void MyListItem::**setValue**(qint32 **newValue**)  {  value = newValue;  emit valueChanged();  } |
| //main.cpp  #include <QGuiApplication>  #include <QQmlApplicationEngine>  #include "customlistmodel.h"  int main(int argc, char \*argv[])  {  #if QT\_VERSION < QT\_VERSION\_CHECK(6, 0, 0)  QCoreApplication::setAttribute(Qt::AA\_EnableHighDpiScaling);  #endif  QGuiApplication app(argc, argv);  //类型注入  qmlRegisterType<CustomListModel>("MyModule",1,0,"MyListModel");  qmlRegisterType<MyListItem>("MyModule",1,0,"MyListItem");  QQmlApplicationEngine engine;  const QUrl url(QStringLiteral("qrc:/main.qml"));  QObject::connect(&engine, &QQmlApplicationEngine::objectCreated,  &app, [url](QObject \*obj, const QUrl &objUrl) {  if (!obj && url == objUrl)  QCoreApplication::exit(-1);  }, Qt::QueuedConnection);  engine.load(url);  return app.exec();  } | //main.qml  import QtQuick 2.15  import QtQuick.Window 2.15  import QtQuick.Controls 2.15  import MyModule 1.0  Window {  width: 640  height: 480  visible: true  title: *qsTr*("CustomListModel")  MyListModel{  id:*mModel*  datas: [  MyListItem{  id:*jack*  name:"Jack"  value:100  },  MyListItem{  id:*m*  name:"Mary"  value:1200  }  ]  }  Button{  id:*btn*  text: "count"  onClicked: {  *console*.log(*mModel*.datas.length)  }  }  ListView{  anchors.top: *btn*.bottom  anchors.bottom: *parent*.bottom  id:*lstv*  model:*mModel*  delegate: Text{  text: name +":"+value  MouseArea{  anchors.fill: *parent*  onClicked: {  *mModel*.append({name:"Susi",value:300})  }  }  }  }  } |