Author: dailey.dai@openthinks.com

D-Bus Sample

Environment: Ubuntu 16.04.10

D-Bus Server

create Qt project from template: Qt Console Application

add d-bus module in project config file

```
QT -= gui
QT += dbus

CONFIG += c++11 console
CONFIG -= app_bundle
```

define a sample class as service object

```
//person.h
#ifndef PERSON_H
#define PERSON_H
#include <QObject>
class Person : public QObject{
    Q_OBJECT
    Q_CLASSINFO("D-Bus Interface","com.openthinks.dbus.interface")
public:
    explicit Person();
    ~Person();
signals:
    void nameChanged(QString);
    void ageChanged(int);
public slots:
    QString name() const;
    void setName(QString name);
    int age() const;
    void setAge(int age);
private:
    QString m_name;
    int m_age;
};
#endif // PERSON_H

//person.cpp
#include "person.h"
#include <QObject>
Person::Person(){
}
```

```
Person::~Person(){
}
QString Person::name() const{
    return m_name;
}
void Person::setName(QString name){
    this->m_name=name;
    emit nameChanged(name);
}
int Person::age() const{
    return m_age;
}
void Person::setAge(int age){
    this->m_age=age;
}
```

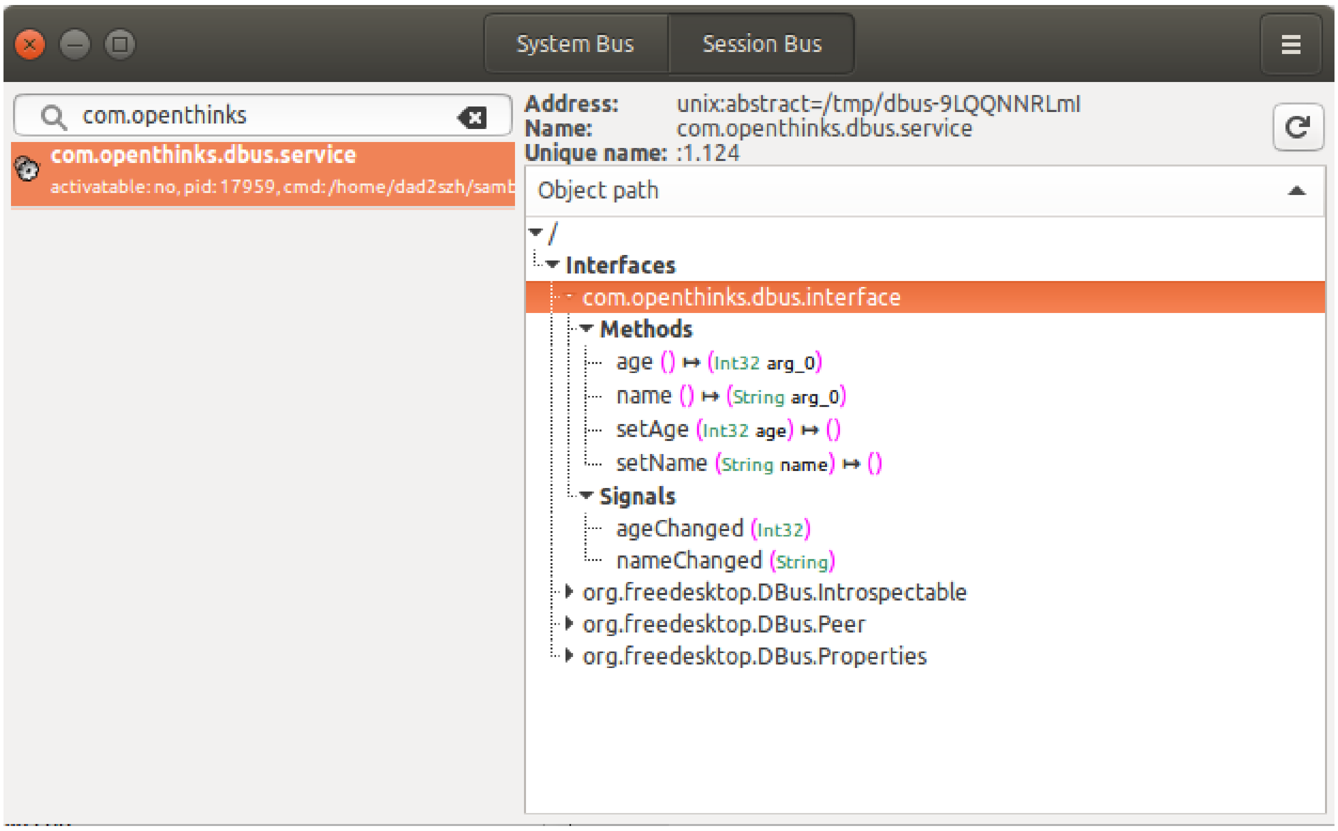
register session service in main.cpp

```
#include <QCoreApplication>
#include <QtDBus/QDBusConnection>
#include "person.h"
int main(int argc, char *argv[])
{
    QCoreApplication a(argc, argv);
    QDBusConnection sessionBus = QDBusConnection::sessionBus();
    if(sessionBus.registerService("com.openthinks.dbus.service")){
        sessionBus.registerObject("/",new Person,QDBusConnection::ExportAllContents);
    }
    return a.exec();
}
```

run the main.cpp in Qt project.

a console will be popup.

open D-Feet to check D-Bus services (could be found in Ubuntu Software)


System Bus
Session Bus

com.openthinks
com.openthinks.dbus.service
activatable: no, pid: 17959, cmd: /home/dad2szh/samb

Address: unix:abstract=/tmp/dbus-9LQQNNRLmI
Name: com.openthinks.dbus.service
Unique name: :1.124

```
Object path
/
  Interfaces
    com.openthinks.dbus.interface
      Methods
        age () ↦ (Int32 arg_0)
        name () ↦ (String arg_0)
        setAge (Int32 age) ↦ ()
        setName (String name) ↦ ()
      Signals
        ageChanged (Int32)
        nameChanged (String)
    org.freedesktop.DBus.Introspectable
    org.freedesktop.DBus.Peer
    org.freedesktop.DBus.Properties
```

D-Bus Client

create Qt project from template: Qt Console Application

add d-bus module in project config file same as above project

define a class to place communication logic

```
//caller.h
#ifndef CALLER_H
#define CALLER_H
#include <QDBusConnection>
#include <QDBusReply>
#include <QDebug>
#include <QObject>
class Caller : public QObject
{
    Q_OBJECT
public:
    explicit Caller(QObject *parent = nullptr);
    void callByMsg();//call method by dbus message
    void callByInterface();//call method by dbus interface
    void callAsync();//call method as async
signals:
public slots:
    void callFinishedSlot(QDBusPendingCallWatcher *call);//async mecthod callback
    void onNameChanged(QString name);//signal callback
};
```

```
#endif // CALLER_H

//caller.cpp
#include "caller.h"
#include <QDBusInterface>
Caller::Caller(QObject *parent) : QObject(parent)
{
    //receive signal method 1
//    QDBusConnection::sessionBus().connect("com.openthinks.dbus.service",
"/","com.openthinks.dbus.interface","nameChanged", this,SLOT(onNameChanged(QString)));
    //receive signal method 2
    QDBusInterface *interface=new
QDBusInterface("com.openthinks.dbus.service","/","com.openthinks.dbus.interface",QDBusConne
ction::sessionBus());
    QObject::connect(interface, SIGNAL(nameChanged(QString)),this,
SLOT(onNameChanged(QString)));
}
void Caller::callByMsg(){
    QDBusMessage msg =
QDBusMessage::createMethodCall("com.openthinks.dbus.service","/","com.openthinks.dbus.inter
face","setName");
    msg<<QString("Alex");
    QDBusMessage response = QDBusConnection::sessionBus().call(msg);
    msg =
QDBusMessage::createMethodCall("com.openthinks.dbus.service","/","com.openthinks.dbus.inter
face","name");
    response = QDBusConnection::sessionBus().call(msg);
    if(response.type()== QDBusMessage::ReplyMessage){
        QString name = response.arguments().takeFirst().toString();
        qDebug()<<"name = "<< name;
    }
}
void Caller::callByInterface(){
    QDBusInterface
interface("com.openthinks.dbus.service","/","com.openthinks.dbus.interface");
    interface.call("setName","Jack");
    QDBusReply<QString> reply = interface.call("name");
    if(reply.isValid()){
        qDebug()<<"name = "<< reply.value();
    }else{
        qDebug()<<"reply not valid.";
    }
    interface.call("setName","Bluce");
    reply = interface.call("name");
    if(reply.isValid()){
        qDebug()<<"name = "<< reply.value();
    }else{
        qDebug()<<"reply not valid.";
    }
}
void Caller::callAsync(){
    QDBusInterface
interface("com.openthinks.dbus.service","/","com.openthinks.dbus.interface");
```

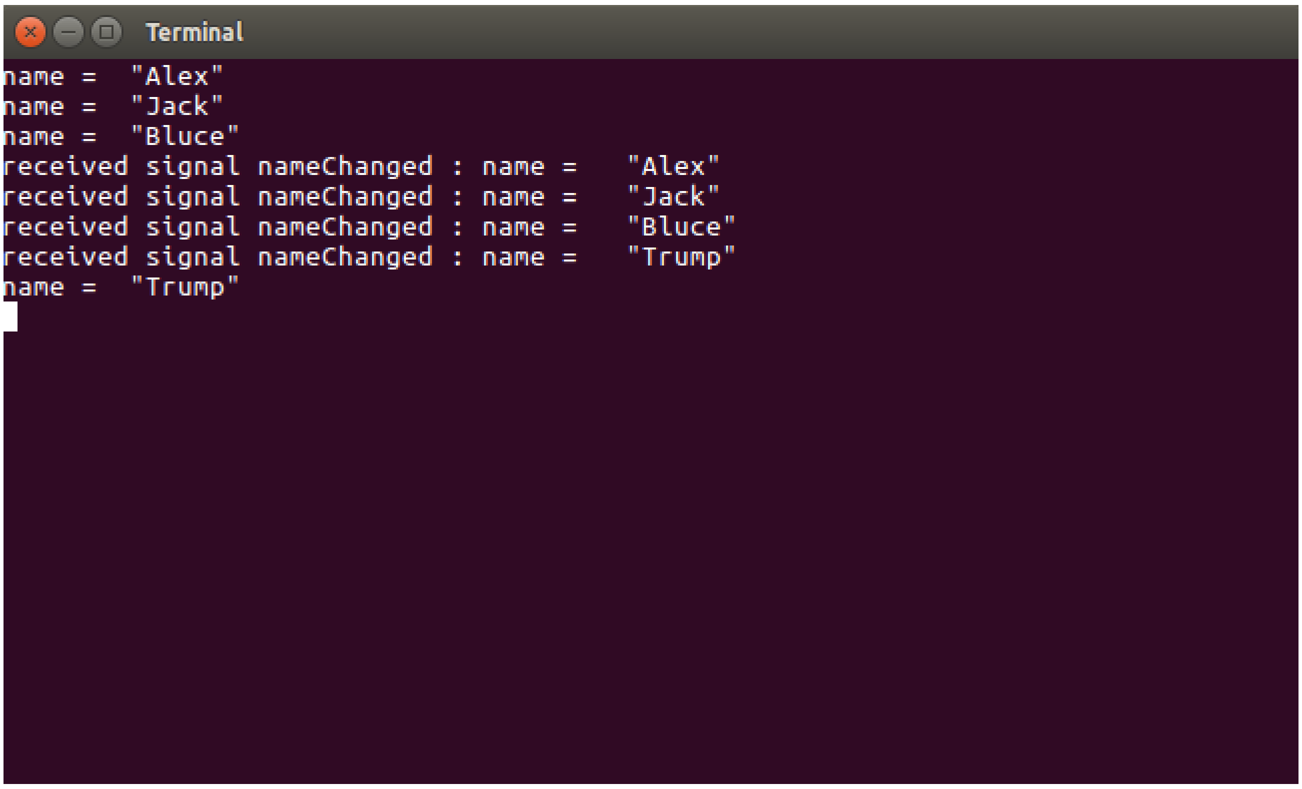
```
    QDBusPendingCall async = interface.asyncCall("setName","Trump");
    async=interface.asyncCall("name");
    QDBusPendingCallWatcher *watcher = new QDBusPendingCallWatcher(async,this);

 QObject::connect(watcher,SIGNAL(finished(QDBusPendingCallWatcher*)),this,SLOT(callFinished
Slot(QDBusPendingCallWatcher*)));
}
void Caller::callFinishedSlot(QDBusPendingCallWatcher *call){
    QDBusPendingReply<QString> reply = *call;
    if(!reply.isError()){
        QString name = reply.argumentAt<0>();
        qDebug()<<"name = "<<name;
    }
    call->deleteLater();
}
void Caller::onNameChanged(QString name){
     qDebug()<<"received signal nameChanged : name =  "<<name;

}
```

```
//main.cpp
#include <QCoreApplication>
#include "caller.h"
int main(int argc, char *argv[])
{
    QCoreApplication a(argc, argv);
    Caller caller;
    caller.callByMsg();
    caller.callByInterface();
    caller.callAsync();
    return a.exec();
}
```

run the main.cpp in Qt project



```
name =  "Alex"
name =  "Jack"
name =  "Bluce"
received signal nameChanged : name =   "Alex"
received signal nameChanged : name =   "Jack"
received signal nameChanged : name =   "Bluce"
received signal nameChanged : name =   "Trump"
name =  "Trump"
```