在[用QtContact 实现MeeGo通讯录（一）](http://software.intel.com/zh-cn/blogs/2011/08/23/qtcontact-meego-0)中我们了解了QtContact基本数据类型和API，以及数据存储和调用的方法，在本篇中，将给出一个详细的例子，来实现上篇中的方法。  
本例子代码来源于Meego-app-contacts 项目，详细请浏览该项目：  
http://meego.gitorious.org/meego-ux/meego-app-contacts  
  
**1、指定数据库**  
QContactManager class 提供了一个接口，让应用程序与特定的backend进行交互。QContactManager默认指定的backend为EDS，建议使用MeeGo官方的release，已经预装了EDS即相关依赖环境。只需要下面的一行代码就可以创建一个指定backend的接口。  
QContactManager \*m\_manager;  
  
**2、填充数据**  
The QContactDetail class represents a single, complete detail about a contact. 基本数据都可以通过QContactDetail的相关继承函数来填充，如：  
QContactAddress adr = curr.detail(QContactAddress::DefinitionName);  
adr.setStreet(m\_addrEdit->text());  
curr.saveDetail(&adr);  
  
**3、查找数据**  
The QContactFilter class is used to select contacts made available through a QContactManager. QContactFilter支持以下类型的查找：  
QContactFilter::InvalidFilter  
QContactFilter::ContactDetailFilter  
QContactFilter::ContactDetailRangeFilter  
QContactFilter::ChangeLogFilter  
QContactFilter::ActionFilter  
QContactFilter::RelationshipFilter  
QContactFilter::IntersectionFilter  
QContactFilter::UnionFilter  
QContactFilter::LocalIdFilter  
QContactFilter::DefaultFilter  
  
**4.显示数据**  
  
**4.1.建立Model**  
  
用QAbstractListModel建立数据模型  
//people.h  
class PeopleModel: public QAbstractListModel  
{  
Q\_OBJECT  
Q\_ENUMS(PeopleRoles)  
Q\_ENUMS(FilterRoles)  
public:  
PeopleModel(QObject \*parent = 0);  
virtual ~PeopleModel();  
enum FilterRoles{  
AllFilter = 0,  
FavoritesFilter,  
OnlineFilter,  
ContactFilter  
};  
enum PeopleRoles{  
ContactRole = Qt::UserRole + 500,  
FirstNameRole, //533  
FirstNameProRole,  
LastNameRole,  
LastNameProRole,  
CompanyNameRole,  
FavoriteRole,  
UuidRole,  
PresenceRole,  
AvatarRole,  
ThumbnailRole,  
IsSelfRole,  
BirthdayRole,  
OnlineAccountUriRole,  
OnlineServiceProviderRole,  
EmailAddressRole,  
EmailContextRole,  
PhoneNumberRole,  
PhoneContextRole,  
AddressRole,  
AddressStreetRole,  
AddressLocaleRole,  
AddressRegionRole,  
AddressCountryRole,  
AddressPostcodeRole,  
AddressContextRole,  
WebUrlRole,  
WebContextRole,  
NotesRole,  
FirstCharacterRole  
};  
//From QAbstractListModel  
Q\_INVOKABLE virtual int rowCount(const QModelIndex& parent= QModelIndex()) const;  
virtual int columnCount(const QModelIndex& parent) const;  
virtual QVariant data(const QModelIndex&, int) const;  
void queueContactSave(QContact contact);  
void removeContact(QContactLocalId contactId);  
//QML API  
Q\_INVOKABLE QVariant data(const int row, int role) const;  
Q\_INVOKABLE bool createPersonModel(QString avatarUrl, QString thumbUrl, QString firstName, QString firstPro,  
QString lastName, QString lastPro,  
QString companyname, QStringList phonenumbers, QStringList phonecontexts,  
bool favorite, QStringList accounturis, QStringList serviceproviders,  
QStringList emailaddys, QStringList emailcontexts, QStringList street,  
QStringList city, QStringList state, QStringList zip, QStringList country,  
QStringList addresscontexts, QStringList urllinks, QStringList urlcontexts,  
QDate birthday, QString notetext);  
Q\_INVOKABLE void deletePerson(const QString& uuid);  
Q\_INVOKABLE void editPersonModel(QString contactId, QString avatarUrl, QString firstName, QString firstPro,  
QString lastName, QString lastPro, QString companyname,  
QStringList phonenumbers, QStringList phonecontexts, bool favorite,  
QStringList accounturis, QStringList serviceproviders, QStringList emailaddys,  
QStringList emailcontexts, QStringList street, QStringList city, QStringList state,  
QStringList zip, QStringList country, QStringList addresscontexts,  
QStringList urllinks, QStringList urlcontexts, QDate birthday, QString notetext);  
Q\_INVOKABLE QMap availableAccounts() const;  
Q\_INVOKABLE QStringList availableContacts(QString accountId) const;  
Q\_INVOKABLE void launch (QString cmd) {  
QProcess::startDetached (cmd);  
}  
Q\_INVOKABLE void exportContact(QString uuid, QString filename);  
Q\_INVOKABLE void sort(int flags);  
Q\_INVOKABLE void setCurrentUuid(const QString& uuid);  
QString currentUuid();  
Q\_INVOKABLE void toggleFavorite(const QString& uuid);  
bool isSelfContact(const QContactLocalId id) const;  
bool isSelfContact(const QUuid id) const;  
Q\_INVOKABLE void setSorting(int role);  
Q\_INVOKABLE void setFilter(int role, bool dataResetNeeded = true);  
Q\_INVOKABLE int getSortingRole();  
Q\_INVOKABLE void searchContacts(const QString text);  
Q\_INVOKABLE void clearSearch();  
Q\_INVOKABLE void fetchOnlineOnly(const QVariantList &stringids);  
protected:  
void fixIndexMap();  
void addContacts(const QList contactsList, int size);  
private slots:  
void onSaveStateChanged(QContactAbstractRequest::State requestState);  
void onRemoveStateChanged(QContactAbstractRequest::State requestState);  
void onDataResetFetchChanged(QContactAbstractRequest::State requestState);  
void onAddedFetchChanged(QContactAbstractRequest::State requestState);  
void onChangedFetchChanged(QContactAbstractRequest::State requestState);  
void onMeFetchRequestStateChanged(QContactAbstractRequest::State requestState);  
void contactsAdded(const QList& contactIds);  
void contactsChanged(const QList& contactIds);  
void contactsRemoved(const QList& contactIds);  
void dataReset();  
void savePendingContacts();  
void createMeCard();  
void vCardFinished(QVersitWriter::State state);  
private:  
PeopleModelPriv \*priv;  
Q\_DISABLE\_COPY(PeopleModel);  
};  
  
People.cpp  
实现people.h中相关函数。  
  
**4.2.实现View**  
可以使用Contacts QML Plugin中的ListView 来显示。  
property PeopleModel dataModel: contactModel  
ListView {  
id: cardListView  
anchors.top: groupedViewPortrait.top  
anchors.right: groupedViewPortrait.right  
anchors.left: groupedViewPortrait.left  
height: groupedViewPortrait.height  
width: groupedViewPortrait.width  
snapMode: ListView.SnapToItem  
highlightFollowsCurrentItem: false  
focus: true  
keyNavigationWraps: false  
clip: true  
model: sortModel  
opacity: 0  
  
delegate: ContactCardPortrait  
{  
id: card  
dataPeople: dataModel  
sortPeople: proxyModel  
onClicked:  
{  
cardListView.currentIndex = index;  
window.currentContactIndex = index;  
  
//When querying the DataModel, use the index of the contact in the  
//not the index of the contact in the ProxyModel  
var srcIndex = sortModel.getSourceRow(index);  
window.currentContactId = dataPeople.data(srcIndex, PeopleModel.UuidRole);  
window.addPage(myAppDetails);  
}  
onPressAndHold: {  
cardListView.currentIndex = index;  
window.currentContactIndex = index;  
window.currentContactId = uuid;  
window.currentContactName = name;  
groupedViewPortrait.pressAndHold(mouseX, mouseY);  
}  
Binding{target: cardListView; property: "height"; value: ((cardListView.count > 1) ? groupedViewPortrait.height : cardListView.childrenRect.height)}  
Binding{target: cardListView; property: "interactive"; value: ((cardListView.count > 1) ? true : false)}  
}  
  
section.property: "firstcharacter"  
section.criteria: ViewSection.FirstCharacter  
section.delegate: HeaderPortrait{width: cardListView.width;}  
}   
  
基本数据类型就介绍到此，下一篇我们将深入探讨如何保存社交网络和及时通讯相关的数据。