QReverseGeocode component for quick reverse geocoding in QML

Introduction

Reverse geocoding is the process of back (reverse) coding of a point location (latitude, longitude) to a readable address or place name. This permits the identification of nearby street addresses, places, and/or areal subdivisions such as neighbourhoods, county, state, or country. Combined with geocoding and routing services, reverse geocoding is a critical component of mobile location-based services.
Nokia provides one of the best location based services available and reverse geocoding is well supported in Qt/C++ and HTML5, but sometimes could be tricky to access this service from QML.
QReverseGeocode provide a quick and easy solution to manage reverse geocoding from QML.

Installation

1. Download source code from here
2. Include **QReverseGeocode** directory into your project

.pro

```
include(./qreversegeocode/reversegeocode.pri)
```

main.cpp

```
#include <QtGui/QApplication>
#include "qmlapplicationviewer.h"

#include <QtDeclarative>
#include "qreversegeocode.h"

Q_DECL_EXPORT int main(int argc, char *argv[])
{
    QScopedPointer<QApplication> app(createApplication(argc, argv));

    qmlRegisterType<QReverseGeocode>("QReverseGeocode", 1, 0, "QReverseGeocode");

    QmlApplicationViewer viewer;
    viewer.setMainQmlFile(QLatin1String("qml/QReverseGeocodeSample/main.qml"));
    viewer.showExpanded();

    return app->exec();
}
```

QML

Note: QReverseGeocode have to be used in conjunction with **PositionSource** component as shown below

```
import QtQuick 1.1
import com.nokia.symbian 1.1
import QtWebKit 1.0
import QtMobility.location 1.2
import QReverseGeocode 1.0
```


(C) Copyright Nokia 2014. All rights reserved.

```
Page {
    id: mainPage

    Text {
          id: place
          text: ""
     }

    PositionSource {
        id: positionSource
        active: true

        onPositionChanged: {
            reverseGeocoder.latitude  = positionSource.position.coordinate.latitude
            reverseGeocoder.longitude = positionSource.position.coordinate.longitude
            reverseGeocoder.process();
        }
    }

    QReverseGeocode {
        id:reverseGeocoder
        onReverseGeocodeFinished:{
            place.text=reverseGeocoder.postCode+" "+reverseGeocoder.city
        }
    }

}
```

API Reference

Properties (To use before calling `process()`)

- `latitude` - Property that hold the latitude value
- `longitude` - Property that hold the longitude value

Properties (Filled when `onReverseGeocodeFinished()` is emitted)

- `city` - Property that hold the city value
- `postCode` - Property that hold post code value
- `street` - Property that hold the street value
- `country` - Property that hold the country value

Signals (handlers)

- `onLatitudeChanged()` - Invoked when latitude property changes.
- `onLongitudeChanged()` - Invoked when the longitude changes.
- `onReverseGeocodeFinished()` - Invoked when the reverse geocode process is finished.

Slots

- `process()` - Start the reverse geocoding process. Before calling this method the `latitude` and `longitude` properties have to be filled.

onReverseGeocodeFinished

When `onReverseGeocodeFinished()` is emitted all address informations are stored into the properties described before, so you can compose your informations in the way you need. For example you could need to display just *post code* and *city*:

```
onReverseGeocodeFinished:{
            place.text=reverseGeocoder.postCode+" "+reverseGeocoder.city
```


(C) Copyright Nokia 2014. All rights reserved.

```
}
```

or just city and country:

```
onReverseGeocodeFinished:{
            place.text=reverseGeocoder.city+" "+reverseGeocoder.country
}
```

rather than all informations:

```
onReverseGeocodeFinished:{
            place.text=reverseGeocoder.street+"<br/>"+reverseGeocoder.postCode+"
"+reverseGeocoder.city+"<br/>"+reverseGeocoder.country
}
```


(C) Copyright Nokia 2014. All rights reserved.