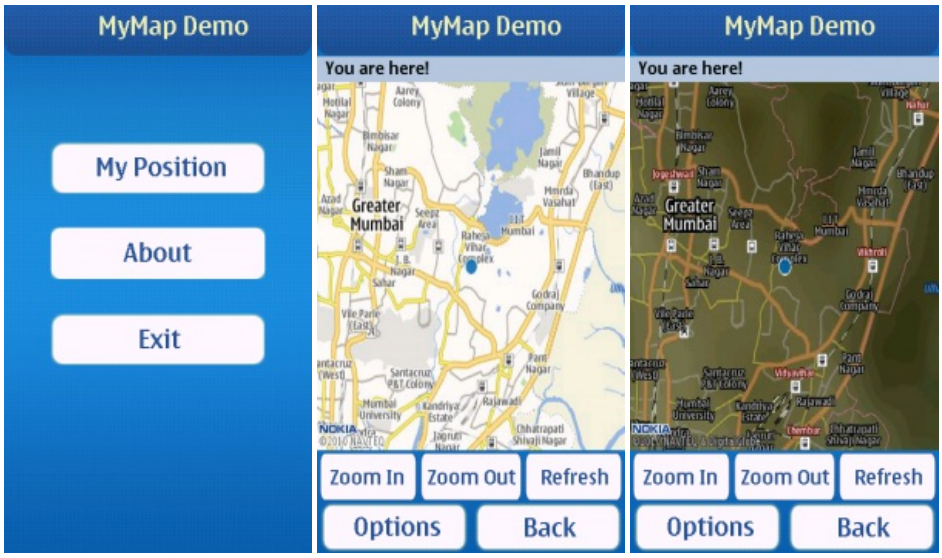
MyMapDemo - Using the RESTful Map API with QtQuick

This article illustrates how to use the **RESTful Map API** in a QtQuick application using a demo project **MyMapDemo**.



Introduction

Nokia Location Platform offers the RESTful Map API - a RESTful service that gives pre-rendered map images for the requested location. It also offers sophisticated APIs like Maps API for Qt (for instance, QGraphicsGeoMap part of the Qt Mobility APIs 1.2) - however the RESTful Map API is very handy to be used in applications which do not have constant or have limited data (GPRS/3G) connectivity. This article explains how to use the "RESTful Map API" in a QtQuick application - we will use a Demo application (MyMapDemo)to illustrate this.

MyMapDemo shows the current location of the user on a Map. Interactivity is added by features such as Zoom in, Zoom out and Refresh Map. By default, the app shows a normal map view; but the user can choose to change the map mode to any of the following modes,

1. Normal map view, day light mode
2. Hybrid map view, day light mode
3. Satellite map view, day light mode
4. Terrain map view, day light mode

The app also contains a simple caching logic for Map images, so that the same map image is not fetched again, this saves data and loading time.

MyMapDemo uses positioning information from the integrated GPS receiver, and uses SimpleViewManager for QML view switching.

RESTful Map API

The simplicity of the API lies in the fact that it is an HTTP request to the server with desired parameters.

Each RESTful Map API request is of the format:

```
http://m.nok.it/{resource}?app_id=XXX&token=YYY&meter=value&...
```

Nokia provides several services options within the Maps API offering. The service is free to use, but you must obtain and use authentication and authorization credentials to use the services. Please read the Terms and Conditions and check the Pricing Plans page to decide which business model best fits your needs. Authentication requires unique Maps API credentials, namely an AppId and a token. You can get these credentials free for free following the instructions here

In some cases, the platform may offer access even without providing credentials, but this is not guaranteed. In this article, and in MyMapDemo we are not including the credentials so that it can be compiled and run right away, serving it's tutorial purpose; however you may need to include your **app_id** and **token** .

(C) Copyright Nokia 2014. All rights reserved.

Basic UI

The app contains 2 important screen views - Home screen and MyMap screen. It also contains 2 dialogs - Options dialog and About dialog. The basic structure and view switching is shown below. SimpleViewManager.qml handles the view switching from the home screen and back buttons.

```
Image  {
    id: myApp
    width: 360
    height: 640
    source: "images/background.png"

    Image
    {
        id: titleBar
        source: "images/title.png"
    }

    // Switching between screens
    SimpleViewManager
    {
        id: myViewManager
        width: parent.width
        height: parent.height - titleBar.height
        anchors.top: titleBar.bottom

        // Home Screen
        HomeScreen
        {
            id: homeScreenView
        }

        // MyMap Screen
        MyMap
        {
            id: myMapView
        }
    }

    // Options dialog
    Options
    {
        id: optionsDialog
        anchors.top: parent.top
    }

    // About dialog
    About
    {
        id: aboutDialog
        anchors.top: parent.top
    }
}
```

The UI code for each screen can be explored in the attached source code.

(C) Copyright Nokia 2014. All rights reserved.

GPS Requestor

The GPSRequestor class is responsible for obtaining the positioning info - latitude and longitude. Also, whenever the position is changed `updateLatLong` signal is released.

```
GPSRequestor::GPSRequestor()
{
    startGPS();
}

void GPSRequestor::startGPS()
{
    // Obtain the location data source if it is not obtained already
    if (!locationDataSource)
    {
        locationDataSource =
                QGeoPositionInfoSource::createDefaultSource(this);
        // Whenever the location data source signals that the current
        // position is updated, the positionUpdated function is called
        connect(locationDataSource, SIGNAL(positionUpdated(QGeoPositionInfo)),
                this, SLOT(positionUpdated(QGeoPositionInfo)));

        // Start listening for position updates
        locationDataSource->startUpdates();
    }
}

void GPSRequestor::positionUpdated(QGeoPositionInfo geoPositionInfo)
{
    if (geoPositionInfo.isValid())
    {
        // We've got the position. No need to continue the listening.
        locationDataSource->stopUpdates();

        // Save the position information into a member variable
        myPositionInfo = geoPositionInfo;

        // Get the current location as latitude and longitude
        QGeoCoordinate geoCoordinate = geoPositionInfo.coordinate();
        qreal latitude = geoCoordinate.latitude();
        qreal longitude = geoCoordinate.longitude();
        updateLatLong(QString("%1").arg(latitude),QString("%1").arg(longitude));
    }
}
```

Map Requestor

Basically, the MyMapRequestor class does the HTTP GET request for obtaining the map from the RESTful Map API, using the GPS positioning info available from GPSRequestor.

The URL for HTTP GET request is created as below,

```
QString url = "http://m.nok.it/?c="+iLat+","+iLng+",&h="+ MAP_HEIGHT + ",&w="+ MAP_WIDTH
+ "&z="+ QString::number(MAP_ZOOM) + "&nord";
```


(C) Copyright Nokia 2014. All rights reserved.

Besides height and width parameters, the url contains latitute, longitude, zoom level and type of the map requested. The default zoom level (MAP_ZOOM) is defined to be 14 . More info about the type of the map here.

When response is received in the form of a map image, it is passed to the QML UI and is displayed in the MyMap View.

MyMap View

Besides displaying the map obtained from the MyMapRequestor class, the Map Screen View contains some buttons like,

- Zoom in
- Zoom out
- Refresh
- Options

The MyMapRequestor class is exposed to QML so that functions for the above actions can be accessed from QML.

```
// Exposing MyMapRequestor class to QML
ctxt = viewer.rootContext();
ctxt->setContextProperty("MyMapRequestor", this);
```

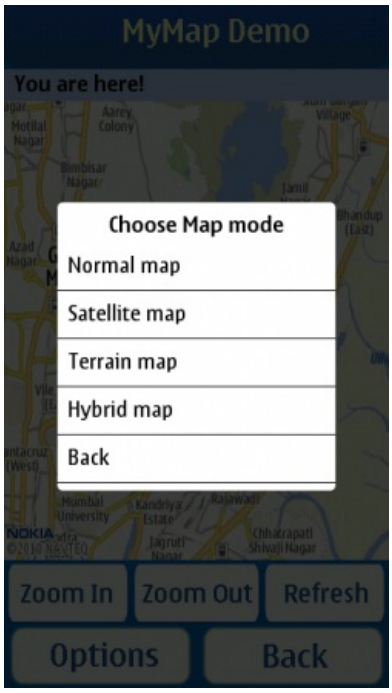
When the Zoom in or Zoom out button is clicked, the value of MAP_ZOOM is increased or decreased and a new HTTP GET request is executed to obtain the new map image. Again, as the response is received, the map on the MyMap view is updated.

```
    // Zoom in button
    ImageButton{
        id: zoomInButton
        normalImage: "images/zoomin.png"
        pressedImage: normalImage
        width: parent.width/3
        height: 60
        anchors.top:mapImage.bottom
        anchors.left: parent.left

        MouseArea{
            anchors.fill: parent
            // Accessing the zoomIn() from MyMapRequestor.cpp
            onClicked:MyMapRequestor.zoomIn();
        }
    }

    // Zoom out button
    ImageButton{
        id: zoomOutButton
        normalImage: "images/zoomout.png"
        pressedImage: normalImage
        width: parent.width/3
        height: 60
        anchors.top:  mapImage.bottom
        anchors.left: zoomInButton.right

        MouseArea{
            anchors.fill: parent
            // Accessing the zoomOut() from MyMapRequestor.cpp
            onClicked:MyMapRequestor.zoomOut();
        }
    }
```


(C) Copyright Nokia 2014. All rights reserved.

Refresh is suggested to be used when the location of the user has changed and the user wants to see the updated map. Clicking on refresh would use the positioning info from the GPSRequestor and update the map by doing a HTTP GET request.

```
    // Refresh button
    ImageButton{
        id: refreshButton
        normalImage: "images/refresh.png"
        pressedImage: normalImage
        width: parent.width/3
        height: 60
        anchors.top:  mapImage.bottom
        anchors.left: zoomOutButton.right

        MouseArea{
            anchors.fill: parent
           // Accessing the getMap from MyMapRequestor.cpp
            onClicked:MyMapRequestor.getMap(0);
        }
    }
```



When the Options button is clicked, an options dialog is displayed. The user can change the type of the map being displayed. The map type is denoted by t in the url; below are the different values of t and the type of map they represent.

Map type (t)	value of t
Normal map view, day light mode	t=0
Hybrid map view, day light mode	t=1
Satellite map view, day light mode	t=2
Terrain map view, day light mode	t=3

When the map type is changed, the map is updated by making a new HTTP GET request with corresponding value of t.

```
MouseArea{
    anchors.fill: parent
    onClicked: {

        if (index!=4){
```


(C) Copyright Nokia 2014. All rights reserved.

```
            // Requesting map with selected map type
            MyMapRequestor.getMap(index);
        }
        optionsDialog.opacity = 0
    }
}
```

Caching Logic

It made sense to have some sort of caching logic in the app, so that when Zoom in and Zoom out is used - unnnecessary connections/HTTP requests are not made to the RESTful Map API. The logic behind this is to cache last couple of images and always check the cache before making a new request. Since in this demo app, a map image (map snap) can be identified uniquely from parameters like latitude, longitude, zoom level and type of map - the following class and functions are used for caching.

```
/* MapSnaps class for caching */
class MapSnaps{
public:
    MapSnaps(QImage a_map, QString a_lat, QString a_lng,
             int a_type, int a_zoomLevel)
    {
        map = a_map;
        lat= a_lat;
        lng = a_lng;
        type = a_type ;
        zoomLevel = a_zoomLevel;
    }

    QImage map;
    QString lat, lng;
    int type, zoomLevel;
};

// Adding map to Cache
void MyMapRequestor::addToCacheMap(QImage image)
{
    mapCache.append(new MapSnaps(image, iLat, iLng,iType, MAP_ZOOM ));
    if (mapCache.length() >5)
    {
        mapCache.takeFirst();
    }
}

// Checking if the requested map alreading exits in Cache
bool MyMapRequestor::checkCache()
{
    for (int i = 0; i < mapCache.length(); i++) {

        if (mapCache.at(i)->lat == iLat &&
                mapCache.at(i)->lng == iLng &&
                mapCache.at(i)->type == iType &&
                mapCache.at(i)->zoomLevel == MAP_ZOOM)
        {
            mapCache.at(i)->map.save(MAP_PATH);
            MapResponseReceived();
```


(C) Copyright Nokia 2014. All rights reserved.

```
            // return true if Map image is available from cache
             return true;
        }
    }
    // return false if Map image is not available from cache
    return false;
}
```

Video demo

Below is a video of MyMapDemo in action. The media player is loading...

Source code

- The complete source for MyMapDemo is available here.

(C) Copyright Nokia 2014. All rights reserved.