Using the RESTful Map API in QML

The **RESTful Map API** is a REST API that provides easy and fast access to pre-rendered map data for all regions of the world for displaying position data. This article shows how the RESTful Map API can be used with plain QML.

19 Jun 2011

The API

The Map API allows access to Nokia Maps data via standard JPEG or PNG images. For this reason, a map can be loaded and handled in QML just as any other (remote) image.

Note: The RESTful Map API is useful when a static map image needs to be loaded and embedded in a mobile application. For more complex scenarios, the **Location QML Plugin from Qt Mobility** is available: QML Location Plugin.

The Image element

The QML Image element supports the loading of remote images, just setting the URI of the image as value of its **source property**.

When loading a remote image, it is highly recommendable to **give the user a visual feedback of the ongoing loading operation**. A QML application knows about the current state of the loading operation thanks to the **status property** of the Image element itself.

The base structure

A basic component should be composed of the following elements:

- a root Item, that contains the component itself
- an Image element, that displays the map image
- another Item, that displays a visual indicator while the loading operation is ongoing

So, taken an empty OviMapsTile.qml file, the base structure can be written as follows:

```
Item {
    id: mapTile

    Image {
 id: mapImage
 anchors.fill: parent
    }
    Column
    {
 id: loadingView
 anchors.centerIn: parent

 AnimatedImage {
     source: "pics/ajax-loader.gif"
     anchors.horizontalCenter: parent.horizontalCenter
 }

 Text {
     text: "Loading map..."
```

```
 }
    }
}
```

Adding map properties

A complete list of all the applicable map properties is available on the RESTful Map API reference page. This article considers the very basic properties needed to properly display a map:

- width
- height
- latitude
- longitude
- zoom

While width and height can be retrieved at runtime by accessing the properties of the Image element, the component defines three explicit properties to hold the values of the others.

```
Item {
    id: mapTile

    property real latitude
    property real longitude
    property int zoom

    [...]
}
```

Loading the map tile

Given the component structure defined above, it is possible to define a JavaScript function that performs the **loading of the map image**, showing the loading indicator and hiding the map tile.

```
function loadMap()
{
    mapImage.visible = false;
    loadingView.visible = true;

    mapImage.source = 'http://m.nok.it/?c=' + latitude + ',' + longitude +
 '&z=' + zoom +
 '&w=' + mapImage.width + '&h=' + mapImage.height+"&nord";
}
```

Once the map image has been loaded, the Image element needs to be displayed, and the loading indicator must be hidden. This can be done by handling the onStatusChanged signal of the Image element itself:

```
Image {
    id: mapImage

    anchors.fill: parent
    onStatusChanged: {
 loadingView.visible = (mapImage.status != Image.Ready)
 mapImage.visible = (mapImage.status == Image.Ready)
    }
}
```

To load the map tile as soon as the component is ready, it is enough to call the loadMap() method on its Component.onCompleted signal:

```
Item {
    id: mapTile

    [...]

    Component.onCompleted: {
        loadMap()
    }
}
```

Handling property changes

As it is now, the component displays a static map, with given coordinates and level of zoom. It would be nice to **add a dynamic behavior to the component** itself, allowing it to refresh the map image when those properties change (for instance, a new level of zoom is set).

Implementing this functionality is straightforward, thanks to the **QML inbuilt signals** sent when a property value changes. Modifying the onCompleted signal as follows is enough to load a new map image as soon as the value of the latitude, longitude or zoom properties changes.

```
Item {
    id: mapTile

    [...]

    Component.onCompleted: {

 mapTile.zoomChanged.connect(loadMap)
 mapTile.latitudeChanged.connect(loadMap)
 mapTile.longitudeChanged.connect(loadMap)

 loadMap()
    }
}
```

How to use it

The component defined above can be used as follows:

```
OviMapTile {
    id: map1
    x: 10
    y: 10
    width: 100
    height: 100
    latitude: 52.531
    longitude: 13.3845
    zoom: 17
}
```

A video of the component in action can be seen below.

The media player is loading...

Related content

A Qt Creator project containing the source code presented in this article is available here: File:QMLOviMaps.zip