Qt Location API

The Qt Mobility (version 1.1.0 and higher) Qt Location API provides a map widget, routing, geocoding and reverse geocoding functionality. In addition to this Qt Mobility provides positioning and landmark functionality.

Installation process

1. Install the latest Qt SDK (version 1.1 and higher)
2. When installing the SDK you have to accept the T&C for the Location APIs
3. Start developing software
4. Register your application before deploying it to the store or launching your service.

For more information on the business options please read our FAQs.

Maps, routing, geocoding and reverse geocoding

The Maps and Navigation API is based on plugins.

Since most providers of mapping, geocoding and routing information offer no guarantees that their data is interoperable with the data provided by other services, the plugins are used to group the functionality per service provider.

The plugins are accessed via `QGeoServiceProvider`, and a Nokia based plugin is part of Qt Mobility. See the section The Nokia plugin for more details.

```
QGeoMappingManager *mappingManager = 0;
QGeoRoutingManager *routingManager = 0;
QGeoSearchManager *searchManager = 0;

QGeoServiceProvider serviceProvider("plugin name");

if (serviceProvider.error() == QGeoServiceProvider::NoError) {
    mappingManager = serviceProvider.mappingManager();
    routingManager = serviceProvider.routingManager();
    searchManager = serviceProvider.searchManager();
}
```

Common classes

`QGeoBoundingArea`	Defines a geographic area
`QGeoBoundingBox`	Defines a rectangular geographic area
`QGeoBoundingCircle`	Defines a circular geographic area
`QGeoServiceProvider`	Aggregates access to services which provide geographical information

Mapping

The `QGraphicsGeoMap` class is the main class used for displaying and interacting with maps. It is designed for use within the Graphics View Framework, and is a subclass of `QGraphicsWidget`.

The `QGeoMappingManager` provides most of the functionality required by `QGraphicsGeoMap`. The details of `QGeoMappingManager` are mostly only important to plugin implementers, as regular users should not need to make use of `QGeoMappingManager` outside of the `QGraphicsGeoMap` constructor:

```
QGraphicsGeoMap *map = new QGraphicsGeoMap(mappingManager);
```

`QGeoMapOverlay`	Used to draw overlays on the map
`QGeoMappingManager`	Support for displaying and interacting with maps
`QGraphicsGeoMap`	Used to display a map and manager the interactions between the user and the map

Map objects

`QGeoMapObject` and its subclasses provide the ability to add graphics to the map specified in terms of coordinates and distances. `QGeoMapObject` instances can also be grouped into hierarchies in order to simplify the process of creating compound objects and managing groups of objects.

`QGeoMapCircleObject`	Used to draw overlays on the map
`QGeoMapGroupObject`	Support for displaying and interacting with maps
`QGeoMapObject`	QGeoMapObject used to draw a pixmap on a map
`QGeoMapPixmapObject`	Used to display a map and manager the interactions between the user and the map
`QGeoMapPolygonObject`	QGeoMapObject used to draw a polygon on a map
`QGeoMapPolylineObject`	QGeoMapObject used to draw a segmented line on a map
`QGeoMapRectangleObject`	QGeoMapObject used to draw a rectangular region on a map
`QGeoMapRouteObject`	QGeoMapObject used to draw a route on a map

Routing

`QGeoRoutingManager` handles requests for routing information.

The requests are created as QGeoRouteRequest instances, which are passed to QGeoRoutingManager::calculateRoute(). The returned QGeoRouteReply instance will contain the result of the request when it is completed.

The QGeoRoute class describes the resulting route. Each route is broken up into a number of QGeoRouteSegment instances, with the division usually occurring at either user specified waypoints or at changes in the mode of transport, like when changing from a train to a bus.

Each QGeoRouteSegment has a QGeoNavigationInstruction instance which describes the instructions that would be issued to a user attempting to follow a route. These instructions a location, which is typically somewhere near the end of the associated QGeoRouteSegment, and instruction text describing how the next QGeoRouteSegment should be reached.

QGeoManeuver	Represents the information relevant to the point at which two QGeoRouteSegments meet
QGeoRoute	Represents a route between two points
QGeoRouteReply	Manages an operation started by an instance of QGeoRoutingManager
QGeoRouteRequest	Represents the parameters and restrictions which define a request for routing information
QGeoRouteSegment	Represents a segment of a route
QGeoRoutingManager	Support for geographic routing operations

Geocoding and searching for places

QGeoSearchManager handles geocoding, reverse geocoding and free-text search for places.

The free-text search will attempt to geocode text that looks like an address while simultaneously searching any landmark databases that the service pro. It is even possibly to add additional QLandmarkManager instances to the soures of data, so that users can search online databases alongside their personal offline landmarks store.

QGeoSearchManager	Support for searching operations related to geographic information
QGeoSearchReply	Manages an operation started by an instance of QGeoSearchManager

The Nokia plugin

Qt Mobility ships with a Maps and Navigation API plugin which accesses the relevant Ovi services provided Nokia. The use of these services is governed by the terms and conditions available in the file plugins/geoservices/nokia/OVI_SERVICES_TERMS_AND_CONDITIONS.txt.

The Ovi services plugin can be loaded by using the plugin key "nokia".

Note that accepting the terms and conditions only applies those terms and conditions to the use of the Ovi Maps Services plugin and does not limit the use of the other maps and navigation API plugins that may be included with the Qt Mobility package.

Implementing plugins

A plugin implementer needs to subclass QGeoServiceProviderFactory and as many of the ManagerEngine classes as they want to provide implementations for.

Subclassing QGeoServiceProviderFactory will only involve exposing a name and a version by overriding QGeoServiceProviderFactory::providerName() and QGeoServiceProviderFactory::providerVersion(), and overriding QGeoServiceProviderFactory::createSearchManagerEngine(), QGeoServiceProviderFactory::createMappingManagerEngine() and QGeoServiceProviderFactory::createRoutingManagerEngine() as appropriate.

QGeoMapData	Are used as a bridge between QGraphicsGeoMap and QGeoMappingManager
QGeoMapObjectInfo	The base class used to define the parts of QGeoMapObject and its subclasses that are specific to a particular QGeoMapData subclass
QGeoMappingManagerEngine	Interface and convenience methods to implementors of QGeoServiceProvider plugins who want to provides support for displaying and interacting with maps
QGeoRoutingManagerEngine	Interface and convenience methods to implementers of QGeoServiceProvider plugins who want to provide access to geographic routing information
QGeoSearchManager	Interface and convenience methods to implementers of QGeoServiceProvider plugins who want to provide support for searching operations related to geographic data
QGeoServiceProviderFactory	Factory class used as the plugin interface for services related to geographical information

Tile-based map convenience classes

Most of the current tile based mapping APIs are very similar, and so we provide a number of classes intended to make writing tile based mapping plugins much simpler.

If the Mercator projection and the most common tile addressing scheme is used this will mainly involve subclassing QGeoTiledMappingManagerEngine and providing an implementation of QGeoTiledMappingManagerEngine::getTileImage().

QGeoTiledMapData	Subclass of QGeoMapData provided to make working with tile based mapping services more convenient
QGeoTiledMapReply	Manages a tile fetch operation started by an instance of QGeoTiledManagerEngine
QGeoTiledMapRequest	Represents a request for a map tile from a tile-based mapping service
QGeoTiledMappingManagerEngine	Provided to make writing Qt Maps and Navigation API plugins for tiled based mapping services easier

QML support

For details on the QML support provided for the Location API see the documentation for the Location QML Plugin.

Note: At the time of the Qt Mobility 1.1.0 release the QML support for the Maps and Navigation API is incomplete and likely to be refined and improved in the next patch release. Also note that on Symbian (as at Qt Mobility 1.1.0), the QML for the Landmarks API may not behave as expected due to a bug described in QTMOBILITY-611. There may be problems in updating a LandmarkAbstractModel element.