- ☰ Menu
  - Consulting
  - Development
  - Training
  - Support
  - Projects
  - Blog
  - About Us

- Services ▾
  - Consulting
  - Development
  - Training
  - Support
- Projects
- Blog
- About Us

Jul 06 2014 pyqgis qgis

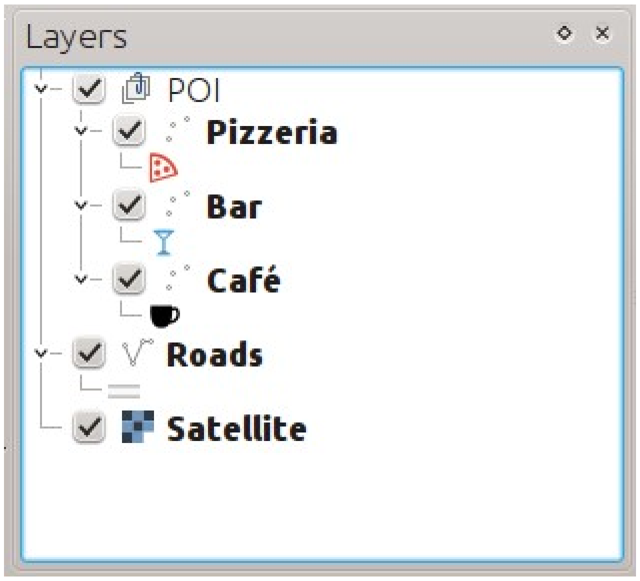
Recent blog posts

QGIS Layer Tree API (Part 1)

This blog post will be about the QGIS component responsible for showing the list of layers. In the QGIS project we typically call this component the “legend widget”. People used to other GIS software may also use other names such as “table of contents (ToC)”.

Layers in the legend widget can be organised into groups. This grouping allows easier manipulation of layers. For example it is possible to toggle the visibility of all layers at once. In addition to layers, groups can also contain other groups, effectively creating a hierarchy of groups and layers. From now on, we will refer to this hierarchy as the *layer tree*.

The legend widget might look like this:

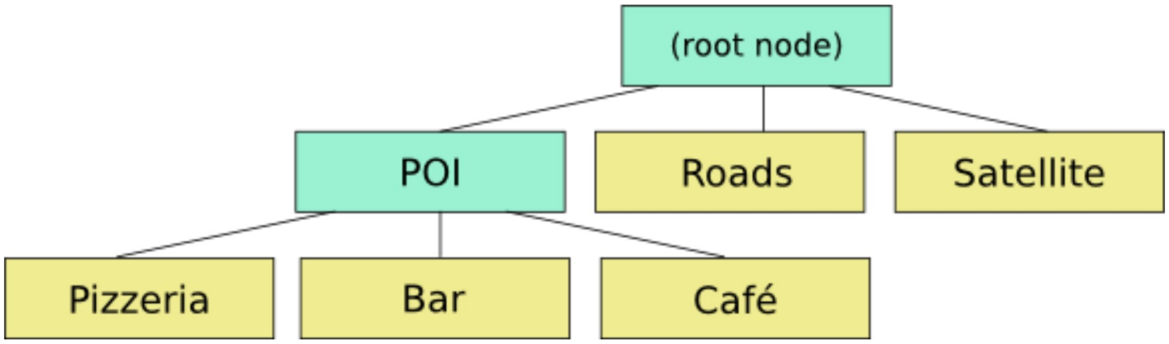


Until QGIS 2.4, there has been only limited support for interacting with the legend widget using the QGIS API. There is a `QgsLegendInterface` class (which can be obtained with `iface.legendInterface()`) available for plugins. The legend interface has emerged in an ad-hoc way, leading to various issues when used in plugins. It is also worth noting that third-party applications based on QGIS have no access to the legend interface.

Layer Tree API

The layer tree API has been introduced in QGIS 2.4 to overcome these existing problems and add even more flexibility to the way the layer tree can be queried or modified.

The layer tree is a classical tree structure built of nodes. There are currently two types of nodes: group nodes and layer nodes. Group nodes can contain other (child) nodes, while layer nodes are 'leaves' of the tree, without any child nodes. The layer tree for the legend widget shown in the picture above looks like this:



The green nodes are group nodes (`QgsLayerTreeGroup` class) and the yellow nodes are layer nodes (`QgsLayerTreeLayer` class).

The legend widget also displays items using symbols, making it look like a real legend. The symbology is not part of the layer tree and will be discussed in an upcoming post.

To start working with the layer tree, we first need a reference to its root node. The project's layer tree can be accessed easily:

```
root = QgsProject.instance().layerTreeRoot()
```

The root node is a group node – its children are shown as top-level items in the legend widget.

```
print root
print root.children()
```

This returns a list of the children of a node. The list includes only direct children – children of sub-groups need to be queried directly from those sub-groups.

Now let's try to access the first child node in the tree and do a little bit of introspection:

```
child0 = root.children()[0]
print child0
print type(child0)
print isinstance(child0, QgsLayerTreeLayer)
print child0.parent()
```

With the `children()` and `parent()` methods it is possible to traverse the layer tree. A node is the root node of a tree if it has no parent:

```
print root.parent()
```

The following example shows how to list top-level items of the layer tree. For group nodes it will print the group name, for layer nodes it will print the layer name and ID.

```
for child in root.children():
  if isinstance(child, QgsLayerTreeGroup):
    print "- group: " + child.name()
  elif isinstance(child, QgsLayerTreeLayer):
    print "- layer: " child.layerName() + "  ID: " + child.layerId()
```

In order to traverse the full layer tree, it would be necessary to recursively call the same code for sub-groups.

There are some helper routines for common tasks like finding nodes representing layers in the tree. These take into account all descendants, not just top-level nodes.

```
ids = root.findLayerIds()
print ids
print root.findLayers()
print root.findLayer(ids[0])
```

It is assumed that a single layer is represented in a layer tree only once. There may however be temporary situations when a layer is represented by more than one node, for example when moving nodes (a new node is created before the old one is removed shortly after).

Similarly it is possible to search for group nodes by name:

```
print root.findGroup("POI")
```

Group names are not necessarily unique – if there are multiple groups with the same name, the first encountered during tree traversal will be returned.

Summary

In this blog post we have shown how to query the project’s layer tree. Upcoming blog entries will focus on modifying the layer tree and interacting with other parts of QGIS.

Posted by Martin Dobias

Tweet G+1 16 Like Share Be the first of your friends to like this.