

QgsVectorLayer Class Reference

- + **polymorphism (wrt arguments) (per instance)**
- + **ownership (Transfer, TransferThis)**
<http://pyqt.sourceforge.net/Docs/sip4/using.html#ownership-of-objects>
- + **class conversion ie qstringlist to list of strings**
- + **what can we automate?**
- + **can doxygen somehow incorporate corresponding sip files?**
- + **can we run the examples automatically?**
- + **python examples should follow PEP8 etc**

Summary

The QgsVectorLayer class represents a layer object from a vector data source. [...]

C++ import

```
#include <qgsvectorlayer.h>
```

Python import

```
from qgis._core import QgsVectorLayer
```

Constructing QgsVectorLayer objects:

[USE existing explanation about providers and how they work]

C++ constructor example:

```
QString uri="path/to/filename"  
+"?type=csv&xField="+lon+"&yField="+lat+"&spatialIndex=no&subsetIndex=no&watchFile=no";  
QString name="test layer";  
QString layerType = "delimitedtext";  
QgsVectorLayer layer = QgsVectorlayer(uri,name,layerType);
```

Python constructor example:

```

uri =
"{ }?type=csv&xField={ }&yField={ }&spatialIndex=no&subsetIndex=no&watch
File=no" \
.format("/path/to/csv/", "lon", "lat")
name="test_layer"
layerType="delimitedtext"
layer = QgsVectorLayer(uri, name, layerType)

```

Other ways to get instances of QgsVectorLayer:

C++:

from the registry:

```

#include "qgsmaplayer.h"
#include "qgsvectorlayer.h"
#include "qgsmaplayerregistry.h"
Q_FOREACH(QgsMapLayer* layer,
QgsMapLayerRegistry::instance()->mapLayers().values()) {
    if(layer->type() == QgsMapLayer::VectorLayer) {
        QgsVectorLayer * vlayer = dynamic_cast<QgsVectorLayer*>(layer);
        // do something with vlayer
    }
}

```

Python:

from the registry:

```

from qgis._core import QgsMapLayerRegistry , QgsMapLayer
allLayers = QgsMapLayerRegistry.instance().mapLayers()
for (id, layer) in allLayers.iteritems():
    if layer.type() == QgsMapLayer.VectorLayer:
        # do something with layer, no casting needed

```

from iface in the console:

```
# get the selected layer
layer = iface.getActiveLayer()
```

Signal Example:

This will log to the QGIS log

every time an attribute is deleted from the layer

C++:

```
#include "qgsmessagelog.h"
void logDeletion(int idx) {
    QgsMessageLog::logMessage("Attribute with index "+idx+" was deleted");
}
```

```
connect(layer, SIGNAL(attributeDeleted(int)), SLOT(logDeletion));
```

Python:

```
from qgis._core import QgsMessageLog
def logDeletion(idx):
    QgsMessageLog.logMessage("Attribute with index {} was deleted".format(idx))

layer.attributeDeleted.connect(logDeletion)
```

Pitfalls for Python API

- The following methods and members are not exported : [...]
- Explanation of what can happen in certain cases if layers are not del(eted) properly

Documentation for individual methods/signals

[use existing]