QGIS Application - Feature request #2392
Plugin layer registry

2010-01-25 07:27 AM - Mathias Walker

Status:	Closed		
Priority:	Low		
Assignee:	Martin Dobias		
Category:	Python plugins		
Pull Request or Patch supplied:		**Resolution:**	fixed
Easy fix?:	No	**Copied to github as #:**	12452

Description

This patch adds a plugin layer registry to support plugin specific maplayers.

Plugins can register a function that creates the specific subclass of QgsMapLayer. This allows to instantiate the corresponding plugin layer when reading the project file.

Workflow

see sample plugin for details

- create subclass of QgsMapLayer
  - implement isEditable(), draw() and writeXml()
- create subclass of QgsPluginLayerCreator
  - implement createLayer() e.g. as callback to a plugin function
- init plugin
  - register PluginLayerCreator using QgsPluginLayerRegistry.instance().addCreator()
- add layer
  - create PluginMapLayer instance of layer type QgsMapLayer.PluginLayer
  - add layer using QgsMapLayerRegistry.instance().addMapLayer()
- save project
  - calls PluginMapLayer.writeXml()
  - save plugin id string as attribute "type"
- load project
  - QgsPluginLayerRegistry calls createLayer() for all registered QgsPluginLayerCreators
  - plugin creates and adds a new PluginMapLayer instance if the attribute "type" matches their id string
- unload plugin
  - unregister PluginLayerCreator using QgsPluginLayerRegistry.instance().removeCreator()

History

#1 - 2010-01-25 07:31 AM - luca76 -

+1

very interesting patch!

#2 - 2010-01-25 08:31 AM - Marco Hugentobler

Martin, do you have time to review this patch?

#3 - 2010-01-26 07:50 AM - Giovanni Manghi

- Resolution set to fixed

- Status changed from Open to Closed

applied in commit:8f32f25a (SVN r12835)

#4 - 2010-01-26 08:03 AM - Martin Dobias

Applied in commit:8f32f25a (SVN r12835) with several modifications:

- plugin layers are subclassed from [[QgsPluginLayer]] and not directly from [[QgsMapLayer]] for convenience
- each plugin layer type has its unique name used in [[QgsPluginLayerRegistry]] - instead of dynamically assigned IDs
- layer creator has been enhanced to return the unique layer type name and to be able to open layer's properties dialog (and renamed to [[QgsPluginLayerType]] as it's not just a creator)
- simplified creation of plugin layers: for the unique type name the registry returns instance of the layer
- when the plugin layer type is removed from registry, layer of that type are removed automatically

I've modified also the sample plugin to reflect my changes.

Sample plugin from the example:

```
LAYER_TYPE = "sample"

class [[SamplePluginLayer]](QgsPluginLayer):
  def +init+(self, width=None):
    [[QgsPluginLayer]].+init+(self, LAYER_TYPE, "Sample plugin layer")

    self.width = width if width is not None else 256
    self.setValid(True)

  def draw(self, rendererContext):
    painter = rendererContext.painter()
    painter.setPen(Qt.red)
    painter.drawRect(32, 32, self.width, 128)
    return True

  def readXml(self, node):
    self.width = node.toElement().attribute("width", "256").toInt()r0
    return True

  def writeXml(self, node, doc):
    element = node.toElement()
    # write plugin layer type to project (essential to be read from project)
    element.setAttribute("type", "plugin")
    element.setAttribute("name", LAYER_TYPE)
    # custom properties
    element.setAttribute("width", str(self.width))
    return True
```

Definition of the layer's type with auxiliary methods:

```
class [[SamplePluginLayerType]](QgsPluginLayerType):
 def +init+(self):
  [[QgsPluginLayerType]].+init+(self, LAYER_TYPE)

 def createLayer(self):
  return [[SamplePluginLayer]]()

 def showLayerProperties(self, layer):
  res = QInputDialog.getInt(None, "Sample plugin", "Set width of the rectangle", layer.width, 1, 1000)
  if resr1: # dialog was not cancelled
   layer.width = resr0
   # trigger repaint
   layer.setCacheImage(None)
   layer.emit(SIGNAL("repaintRequested()"))

  # indicate that we have shown the properties dialog
  return True
```

Registration:

```
[[QgsPluginLayerRegistry]].instance().addPluginLayerType( [[SamplePluginLayerType]]() )
```

Creation of the layer:

```
layer = [[SamplePluginLayer]]()
# -or-
[[QgsPluginLayerRegistry]].instance().pluginLayerType(LAYER_TYPE).createLayer()
```

Finalization:

```
[[QgsPluginLayerRegistry]].instance().removePluginLayerType(LAYER_TYPE)
```

Files

enh2392_plugin_layer_registry.diff	10.6 KB	2010-01-25	Mathias Walker
enh2392_sample_layer_plugin.tar.gz	5.26 KB	2010-01-25	Mathias Walker
sample_layer_plugin_updated.tar.gz	7.15 KB	2010-01-26	Martin Dobias