QuickNode Devlopment Manual

Welcome to the QuickNode Development Manual. This guide will walk you through the process of creating a plugin for QuickNode in Python.

Table of Contents

Keep This in Mind

> 1. **Have a clear workflow** in mind before you start.
> 2. Coding only in **apply function** in one class(/Node)

What is QuickNode

- QuickNode is a powerful and easy-to-use software, developed by SunDuo's team. It is designed to help users create and manage their own workflow with connected nodes.

How to create a plugin for QN in Python

Environment

- python and IDE
- QuickNode installed
- plugincore.dll
- packages (PyQt5, installed by pip)

Get to Know PluginBusiness

> For your convenience, we have provided a `PluginBusiness` class in the structure of dev project. This class is designed to help you create a plugin in most easy way. So you don't need to pay attention What's goning on about the communication between your exe file and QuickNode.

Lests go through the structure of prepared project(qn_Test):

- qn_Test
  - createprocess.py
    - **You only need to code here.**
  - main
    - The main entry of the plugin exe. **Nothing to code in this file.**
  - qn_basenode.py
    - included BaseNode class. **Nothing to code in this file.**
  - qn_BasicType.py
    - included BasicType representations and constants. **Nothing to code in this file.**
  - qn_ErroType.py
    - included ErrorType representations and constants. **Nothing to code in this file.**
  - qn_plugincore.py
    - Included Key class and PluginCore class. **Nothing to code in this file.**
  - qn_pluginbusiness.py
    - Included Key communication class and PluginBusiness class. **Nothing to code in this file.**
  - qn_errors.py
    - Included Error class. **Nothing to code in this file.**

Fast Start

In this section, we'll walk through a demo of creating an Addition Calculator Node.

Step 1: Get Prepared sorce code projects ready.

Sorce Code

With Source code, you can directly start your plugin development.

Step 2: Define a workflow for your plugin(Addition Calculator Node)

- Input: two numbers
- Action: add two numbers
- Outputs: the result of the addition
- Show the result in a message box

Step 3: Create a new class for Addition Calculator Node in createprocess.py

```
from qn_plugincore import *
from qn_basenode import *
import json

@register("QN_Add_Calculator")        # register the class to the plugin
class QNAddCalculator(BaseNode):      # inherit from BaseNode
    def __init__(self):
        super().__init__()
        self.node_name = "QN_Add_Calculator"       # set the node name, should be
same as the registered name

    def apply(self):
        rc = 0                                     # return code, 0 means
success, -1 means failure



        # get inputs from QN
        port = self._input("lhs")
        data = json.loads(port)
        value = data["value"]         # get the value from the input port


        port = self._input("rhs")
        data = json.loads(port)
        value1 = data["value"]        # get the value from the input port

        ret = value + value1



        output_ret = {}
        output_ret["name"] = "ret"
        output_ret["isSharedMemory"] = False
        output_ret["value"] = ret
        output_json = json.dumps(output_ret, indent=4)
        self._setOutput("ret", output_json)          # set the output port
```

```
        return rc
```

Step 4: Using Pyinstaller to create the plugin exe

```
pyinstaller --onefile --name qn_Test --distpath
D:\Reserch\bin\plugins\executable\qn_Test main.py

# onefile: create a single exe file
# name: the name of the exe file
# distpath: the path to save the exe file(should be the plugin folder of
QuickNode)
# main.py: the main entry of the plugin
```

Step 5: Create the conf file with same name of plugin

```
{
    {
    "version": "1.0.0",
    "brief": "qn_Test is plugin for QuickNode. It has ewo simple nodes, one is
addition calculator, another is subtract calculator.",
    "pluginpath": "plugins/executable/qn_Test",
    "nodes":
    [
        {
            "name": "QN_Add_Calculator",   # the name of the node
            "inputs":
            [
                ["int", "lhs"],              # the type and name of the input
port
                ["int", "rhs"]               # the type and name of the input
port
            ],
            "outputs":
            [
                ["int", "ret"]               # the type and name of the output
port

            ],
            "properties":
            [

            ],
            "category": "plugin_Test"      # the category of the plugin(one
plugin is for many nodes)
        }

    ]
}
}
```

Details (About structure itself, will be presented in next Week)

...

Summary

In this guide, we have walked through the process of creating a plugin for QuickNode in Python. We have also provided a demo of creating an Addition Calculator Node. We hope this guide has been helpful to you. If you have any questions, please feel free to contact us.