Uptone Software

sjBot

sjBot is a python IRC bot made by Sjc1000 (Steven J. Core). This document instructs you how to make a plugin or command for sjBot. Have fun ;)

Where do I begin?

Be aware, sjBot is python3. I have not tested python2 code as a plugin but common sense suggests it will not work well. Now that we have that out of the way, we can begin.

There are 2 types of extensions you can make for sjBot. The first being a command, the second being a plugin. These sound similar but sjBot handles them very differently.

Adding a command:

Commands are run by using ` ( or whatever bot command you have ) `command param1 param2 param2 etc.

Create the file:

The first thing you need to do is create a file in the /commands folder. This is where sjBot loads all the commands from. You NEED to put the .py extension at the end of the file otherwise sjBot will not see it as a command. Naming the file is not all that important but it will be one of the triggers for the command, you can add more later.

Add the meta data:

sjBot's command files require a meta_data dictionary in them. This will tell the the bot what kind of info it has.

The keys for the meta_data dictionary include:

- help: The help for this command. This should be a list, the first item being what it is, the second how to use it.
- aliases: The aliases for the command. This should be a list containing all the aliases for this command. **This list should also include the name of the file, without the .py extension.**
- owner: This is a boolean value, 0 is the command can be used by anyone, 1 if the command is owner specific. **Owners are specified in the owner list in the main bot file.**

Define the execute function:

You will need to define a function called 'execute' in this file. SjBot runs this function when the command is ran.

```
def execute(parent, commands, irc, user, host, channel, params):
```

This is an example of the line needed. I will explain the params.

- Parent – This is the sjBot object itself, you can access everything from this.
- Commands – This is the command dictionary, containing all commands. You can find meta_data or use .execute() on commands from here. commands['command_name'].execute() for example.
- Irc – The irc object, This is not the same as the socket object. It is the bot module. (Found in the bot.py file)
- user – The user who called the command.
- Host – The host of the user who called the command.
- Channel – The channel it was called in.
- Params – The params for the command. This is a list. Example: `command param1 param2 param3

Returning a value from this will make sjBot say it to channel. Easy huh? :) You can specify a list to make sjBot say more than 1 line. Returning 0 will make sjBot say nothing.

Alright cool, but I want to make a plugin!

I thought you would never ask! Plugins are very powerful. They are setup to receive the same information sjBot's internal system does. This doesn't allow you to change any of the internal systems, you will need to change the main file for this.

First, I will need to explain how most of the internal system of sjBot works and how IRC works.

IRC sends data with different message types. Things such as PRIVMSG (for a channel message), JOIN (for a join), etc etc. The list is pretty big so I wont list them all here. SjBot handles these messages and calls a function like onPRIVMSG() or onJOIN(). The params passed to these functions change depending on the message type. For PRIVMSG its

```
def onPRIVMSG(self, uhost, channel, *message):
```

Now, in IRC data it would look like

:Sjc1000!~Sjc1000@somehosthere PRIVMSG #Sjc_Bot :This is the

message

sjBot splits the data by spaces then uses them as params, now you can see why its different for every command. Since a join is

def onJOIN(self, uhost, channel)

:Sjc1000!~Sjc1000@somehosthere JOIN #Sjc_Bot

You will notice that JOIN does not appear in the params. **This is currently not the case for the plugins. You will have to add a param just for this. In a plugin JOIN would be def onJOIN(this, uhost, join, channel).**

I use this in the place of self since the plugins are not in a class but still get passed the self data.

Now we know all of this, you can create a .py file in the /plugins folder, you do not have to name it anything specific.

Put in a def onTYPE() function and it will be called when sjBot runs into one. You can have more than 1 onTYPE() function in the file.

Warnings:

BE AWARE. PLUGINS DON'T RETURN DATA TO SJBOT! To send data to IRC you will need to use this.irc.send()

This tutorial is only valid for v8. I try to keep the plugin and command systems the same through the recent versions but its slowly getting dated. I will have to change it soon.