Python Dance Documentation

A Quick Guide to Python Dance Mastery

Francis Ilechukwu

V0.1

TABLE OF CONTENTS

INTRODUCTION

A Server is basically a program which binds to a port on a machine and listens to incomming connections on that port. Each of which are basically sockets or pipe line through which packets of data pass through and from (dance) the two parties involved (client and server).

Servers we know today are usually differentiaed by their protocols as that forms the basis for properly communcting with them. A good example is the Apache® that is know for the 'HTTP' protcol and by default bind to port 80 when running on any machine it is installed on. This server is responsible for serving browsers web pages and other processing. This server alternatively runs on port 433 as well for the 'HTTPS' protocol. another example is the SSH sever that usually runs on port 22 for remote logging in to machines. Then again SSH has it's own protocol for communication.

Any one with a decent level of the knowledge of writing code and build a custom server software no doubt. However, one needs to first design a protocol for communication. This protocol is basically a set of rules that can trigger several actions on the server. Say if a client sends 'Switch-5', the server can then actute something that has to do with the number 5 may be and so on. It's as simple as just forming rules with sets of strings right?.

Not that straight forward any way. For very complicated servers, you can end up with very long and mosnstrous streams of conditional statements that checks incoming data to perform the necessary actions.

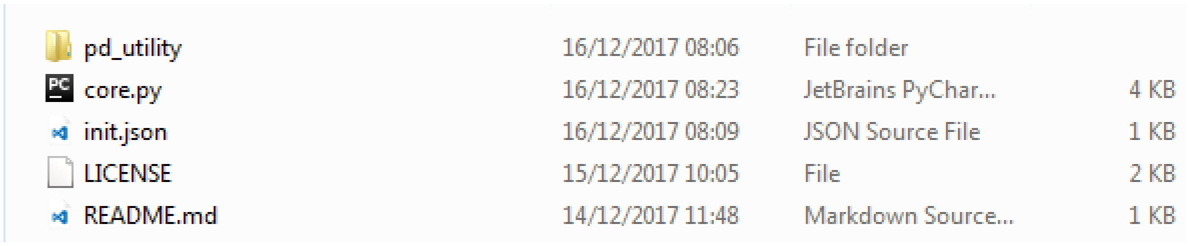
Python Dance (built with python) is an open source framework that makes the development of servers easy and more straight forward, and doing that using JSON based protocols. With this framework, all you need is to provide functions via a package and then control what happenes when certain massages come in through a json file. Pretty simple. You don't need to buffer large sizes of incoming data or worry about the other details. All you have is your json file and your python package.

Building software servers can be one of the things you will do when building or working on IoT projects. This fraework will greatly reduce the development process.

A point to note is that your protocols will be built on top of JSON and a meaningful combination of key value pairs will do the trick. In this document, you will get how to set up and build a simple server using PD (python dance).

SETUP

Setting up a PD Server is very easy and is outlined with the following steps

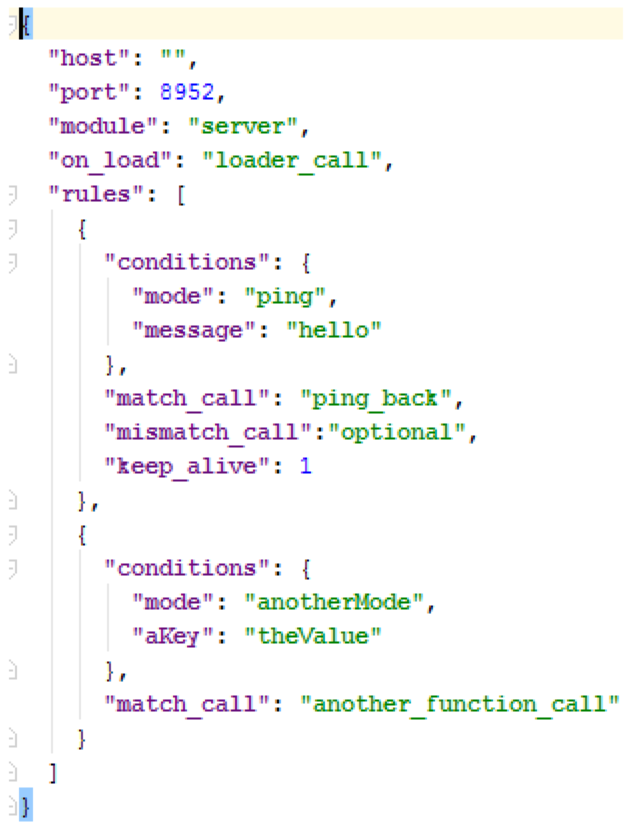
- Go to https://github.com/francis94c/python-dance to get the latest download of PD.
- Extract the zip folder.
- Copy the 'dist' folder out to any location of choice and rename the folder to any thing you want to call your project.
- The files in the folder will be look similar to the ones shown below.
- 

pd_utility	16/12/2017 08:06	File folder	
core.py	16/12/2017 08:23	JetBrains PyChar...	4 KB
init.json	16/12/2017 08:09	JSON Source File	1 KB
LICENSE	15/12/2017 10:05	File	2 KB
README.md	14/12/2017 11:48	Markdown Source...	1 KB

- Create a python module or package in this folder with any name of choice. This module will become the functionality zone of your server as you will see later. (same directory level as core.py)
- Open up the ini.json file and key in relevant information to set up your server. Basically, this file controls what functions in your creted module the server will cann when it receives certain json objects with matching rules wich you will specify in this same json file.
- Set the 'module' in the 'ini.json' file to the name of you custom python module so that your custo functions are accessible by the framework.

To understand how this frame work is used, we must first discuss the ini.json file or as we call it 'the config file'.

The configuration file (ini.json) is the file that determines the behaviour of your server. In fact, it tells the serve every thing to do ranging from which ip address to bind, the port to listen on, to what function to call if a certain pattern is receved. The configuartion file in itself contains a json object with keys whose values determine the behaviour of the server. These keys are self descriptory and in most case esy to understand. A sample ini.json file content is shown below.



```
{
  "host": "",
  "port": 8952,
  "module": "server",
  "on_load": "loader_call",
  "rules": [
    {
      "conditions": {
        "mode": "ping",
        "message": "hello"
      },
      "match_call": "ping_back",
      "mismatch_call":"optional",
      "keep_alive": 1
    },
    {
      "conditions": {
        "mode": "anotherMode",
        "aKey": "theValue"
      },
      "match_call": "another_function_call"
    }
  ]
}
```

The figure above is truly self descriptory and easy to understand as the host field is empty to indicate that it is to start on the local machine (127.0.0.1). to run on another IP address, you only need to specify it.

The most important part of this file as regards behavior is the rules key which is a JSON Array of JSON objects with each having a 'conditions' key which defines the pattern for incoming JSON data you want to capture and call the corresponding value of the 'match_call'.

For example, say a JSON string with the following key value pair came in from the TCP socket being listened to.

- screen->home
- id->ASCDRT5
- action->toggleBlinds

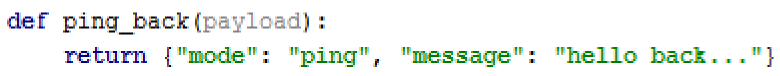
The frame work will look into the rules key in the 'init.json' file and check for conditions that match the received JSON packet and then calls the corresponding function specified by the match_call key of the matching condition. Note that the rule checking is done in the order the rule items are placed and a macth will halt the checking procedure.

CUSTOM MODULE

One necessary part of a working and deployable PD Server is your costom python module whose folder must be at the same directory level as ‘core.py’. the module should also efectivelly provide functions which can be referenced or specified from a ‘match_call’ key.

Usually, this python module should be a folder with an ‘__init__.py’ file within and any other source file you may want to include. Note also that the functions you specify with the ‘match_call’ key must be defined in the ‘__init__.py’ file as you would normally, when building a python package.

The module functions specified by ‘match_call’ keys will be called while passing a payload argument in the form of a dictionary which represents the incoming JSON packet that is currently being responded (the one that triggered the ‘match_call’ key).



```python
def ping_back(payload):
    return {"mode": "ping", "message": "hello back..."}
```

A sample custom module function with a payload argument and dictionary response.

This means that you don’t necesarrily need to define all the macthing key value pairs for your conditions key as you only need specify key ones so that further checking and processing can be done in your module function as you please.

To respond back to a client within your module function, return a dictionary. The framework will convert the dictionary to a JSON string and send back to the client.

APPENDIX

Below is a table that shows the keys in the 'init.json' file and their respective functions/Descriptions.

Keys	Data Type	Function/Description
ip	String	The IP address which you want the server to run on. Leave blank with a null string ("") to run on the local machine (127.0.0.1)
port	Int	The port number the server should bind to or listen on for incoming messages.
module	String	The name of your custom python module where your functions reside.
on_load (optional)	String	The name of the function to call when the server has finished binding to the specified port.
rules	JSON Array	A JSON array containing JSON objects with key value pairs of 'conditions' and 'match_calls' each. This array defines the behaviour of the server along with your custom module functions.
conditions	JSON Object	A JSON object that specifed the key value pair to check for a match in incomming JSON packets.
match_call	String	The name of the module function to call if its corresponding conditions key was succesfully matched.
keep_alive (optional)	Int	A control key that indicates whether the TCP socket should be closed after responding back to the client. A value of 1 keeps the connection open after a response. This key defaults to 0.

This document is still in it's development process, if you have any suggestions or corrections regarding this document or the project in general please feel free to open an issue on https://github.com/francis94c/python-dance. Pull requests are welcome as well.