GDPyS

Feb 27, 2021

Contents

CHAPTER 1

Documentation Contents

1.1 Introduction

GDPyS is a Python based Geometry Dash server, it has many unique features including plugins and custom magic and awarded sections. What makes GDPyS better than other alternatives is because of the speed. GDPyS runs much faster than other alternatives and offers many configuration options like Cheatless AntiCheat and more.

1.2 Plugins

GDPyS plugins are extensions to the main server and they are kinda like if discord.py cogs and spigot plugins had a baby. This guide will show you how to make and use plugins correctly.

First thing we want to create a python file in the plugins directory, lets call our plugin anticheat.

Lets put some starter code in there:

```
import gdpys # importing the main module

class AntiCheat(gdpys.Plugin): # create our plugin
    def __init__(self): # init
        print("Plugin started") # if your plugin needs to initalize, you may do that␣
↪here. (make sure to call super()__init__() last)
        super().__init__()

    async def loop(self): # create a loop that will run every second
        pass # in our case, in this loop we can try to see if someone is cheating.

def setup():
    return AntiCheat # tell the plugin manager what class to find
```

This is not even the slightest bit of what plugins are capable of.

1.2.1 Metadata and dependencies

Next, we can set the metadata of our plugin, and even add dependencies (the metadata of a plugin is almost required but it might still work without it):

```
import gdpys # importing the main module

class AntiCheat(gdpys.Plugin): # create our plugin
    def __init__(self): # init
        self.set_metadata(name="anticheat", author="spook", description="anticheat␣
↪for GDPyS", version="1.0.0", dependencies=[])
        print("Plugin started") # if your plugin needs to initalize, you may do that␣
↪here. (make sure to call super()__init__() last)
        super().__init__()

    async def loop(self): # create a loop that will run every second
        pass # in our case, in this loop we can try to see if someone is cheating.

def setup():
    return AntiCheat # tell the plugin manager what class to find
```

1.2.2 Config

If we want our plugin to be configurable we can use the built in config method:

```
import gdpys # importing the main module

class AntiCheat(gdpys.Plugin): # create our plugin
    def __init__(self): # init
        # set the metadata of the plugin so other plugins and the server admin can␣
↪see it.
        self.set_metadata(name="anticheat", author="spook", description="anticheat␣
↪for GDPyS", version="1.0.0", dependencies=[])
        self.create_config({
            "ban_people": True, # set the default values
            "other_config_option": "1"
        })
        print("Plugin started") # if your plugin needs to initalize, you may do that␣
↪here. (make sure to call super()__init__() last)
        super().__init__()

    async def loop(self): # create a loop that will run every second
        if self.config["ban_people"]: # in our case, in this loop we can try to see␣
↪if someone is cheating.
            pass # now we can ban people

def setup():
    return AntiCheat # tell the plugin manager what class to find
```

1.3 Client

The client class is the main class for building plugins that interact with GDPyS.

We can use the client class by importing gdpys.client to get a working client object:

```
import gdpys

client = gdpys.client # getting the client class

class Example(gdpys.Plugin):
    def __init__(self):
        self.set_metadata(name="example", author="spook", description="example plugin
↪", version="1.0.0", dependencies=[])
        super().__init__()

    async def loop(self):
        user_id = 42069
        comment = "This is an account comment"
        await client.post_account_comment(user_id, comment)
        self.stop()

def setup():
    return Example
```

1.3.1 Client

class gdpys.client.**Client**

await account_id_to_user_id(*accountid*) → int
Convert a user id to an account id

await ban_user(*userid: int*) → None
Ban a user

command(*name: str = None*, *permission: constants.Permissions = None*)
Decorator to create commands

create_command(*name: str*, *coro: asyncio.coroutines.coroutine*, *permission: constants.Permissions*, *type='command'*)
Create a command

await create_user_object(*account_id: int*) → objects.accounts.Account
Create a users object

await get_daily_level() → objects.levels.DailyLevel
Get the current daily level

await get_level(*id: int*) → objects.levels.Level
Get a level object

await get_user_object(*account_id: int*) → objects.accounts.Account
Get a users object

await get_user_rank(*id: int*) → int
Get the rank of a user

await get_weekly_level() → objects.levels.DailyLevel
Get the current weekly level

await like_level(*id: int*)
Bump a level's likes by one

on_comment(*name: str = None*, *permission: constants.Permissions = None*)
Decorator to create on_comment commands

await post_account_comment (*id: int*, *comment: str*) → bool
Post an account comment to a user's account

await rate_level (*rating: objects.levels.Rating*)
Rates a level given a Rating object

await send_message (*subject: str*, *body: str*, *fromuser: int*, *touser: int*) → None
Send a message to a user

await star_to_difficulty (*stars: int*) → int
Convert star rating to a difficulty

await upload_level (*level: objects.levels.Level*)
Uploads a level from a level object

await username_to_id (*username: str*) → int
Convert a username to a user id

1.4 Commands

Commands are built into the Client class of GDPyS plugins and can be used like this:

```
import gdpys

permissions = gdpys.permissions # getting permissions object
client = gdpys.client # getting the client class

class CommandExample(gdpys.Plugin):
    def __init__(self):
        self.set_metadata(name="commandexample", author="spook", description="command␣
↪example plugin", version="1.0.0", dependencies=[])
        super().__init__()

    @client.command(name="rate", permission=Permissions.MOD_RATE)
    async def rate(self): # arguments are not yet implemented
        pass # do stuff on [prefix]rate

def setup():
    return CommandExample
```

CHAPTER 2

Other

- genindex
- search

Index