JSONBOT Documentation

Release 0.91.0 DEVELOPMENT

Bart Thate

December 27, 2012

CONTENTS

```
23:26 <@botfather> hello jsonbot
23:26 <@jsonbot> World, greetings to all joined channels. """. World,greetings to all ppl on partylir
```

JSONBOT is a remote event-driven framework for building bots that talk JSON to each other over XMPP.

This distribution provides bots built on this framework for console, IRC, XMPP, WWW for the shell.

the jsb pakage contains the following programs:

- jsb - console version of jsb
- jsb-makecert - create keys and certifictes for web console SSL
- jsb-init - create data directory and config examples, default ~/.jsb
- jsb-irc - IRC version of jsb
- jsb-fleet - mix IRC and XMPP bots
- jsb-sed - sed a whole directory
- jsb-stop - stop a running bot
- jsb-tornado - a shell web server based on tornado
- jsb-udp - send udp packets to the bot that will relay the data
- jsb-udpdirect - a stand alone udp program, copy & edit
- jsb-xmpp - XMPP version of jsb .. now with OpenFire support !

note: JSONBOT is in BETA stage right now and still subject to change of protocols and API.

see http://jsonbot.googlecode.com.

see *INTRODUCTION* for a quick introduction on the bot.

CHAPTER

ONE

LICENSE

JSONBOT is free code (MIT) and can be cloned where needed.

CHAPTER

TWO

CONTACT THE DEVELOPER

- email: bthate@gmail.com
- jabber/xmpp: bthate@gmail.com
- IRC: botfather on #dunkbots irc.freenode.net
- twitter: http://twitter.com/jsonbot

CHAPTER

THREE

HANDBOOK

3.1 handbook

3.1.1 JSONBOT API DEFINITION

This document defines the API of JSONBOT **!!Still to be developed !!**

- /dispatch

Dispatches an command into the callback handlers.

- /event

Entrypoint for events occured on another jsonbot. event gets inserted into the callback handlers.

- /learn/<item>

Descriptions of a item that has been learned

- /karma/<item>

Karma of an item

- /quote/<nr>

Quote by quotenr

- /quote/random

Random quote

- /quote/last

Last quote

- /log/<channelname>

Return log of the current day

- /log/<channelname>/all

Return log of all the days available

- /say/<botname>/<channel>

Say something in <channel> on <botname>

3.1.2 CONFIGURATION

JSONBOT uses files under the <datadir>/config dir which consists of var = "jsonstring" lines. An example of a config file:

```
bart@monster:~/work/first$ cat ~/.jsb/config/fleet/default-ConsoleBot/config
botname = "default-ConsoleBot"

cfile = "/home/bart/.jsb/config/fleet/default-ConsoleBot/config"

createdfrom = "jsb.lib.config:78"

# directory to store bot data in.
datadir = "/home/bart/.jsb"

dir = "/home/bart/.jsb/config"

filename = "fleet/default-ConsoleBot/config"

# who to follow on the bot .. bot maintains this list.
followlist = []

isdb = false

# the name of the bot.
name = "default-ConsoleBot"

# owner of the bot.
owner = []

# the bot's type.
type = "console"

# bot generated uuid for this config file.
uuid = "94b0e351-0c2f-4e4e-8edf-2a49a790892f"
```

lefthand side is the name of the config variable, right hand side is a jsonstring that contains the value.

Config dirs

Config files are stored in the <datadir>/config directory:

the mainconfig file contains main configuration for bot, the files under the fleet directory contain configurations for the bots (jsonbot can support multiple bots running in 1 instance) and plugin configuration is stored in the plugs subdir.

Programmically

The Configuration class is defined in *jsb.lib.config*. This module contains also the mainconfiguration object which can be retrieved with the getmainconfig() function. The configation variables can be queries with the . notation, so if you do

```
cfg = getmainconfig()
print cfg.owner
```

it will print the owner as defined in the <datadir>/config/mainconfig file.

Configuration files are writable by the bot by calling the .save() function:

```
bot.cfg.server = "jsonbot.org"
bot.save()
```

as you can see the bot specific configuration is stored as bot.cfg (config taken from <datadir>/config/fleet/<botname>/config).

Plugin configuration is done with the *jsb.lib.persistconfig* class that allows for commands to configure them online.

For example the defintion of the udp plugin is done in code:

```
cfg = PersistConfig()
cfg.define('udp', 0) # set to 0 to disnable
cfg.define('udpparty', 0)
cfg.define('udpipv6', 0)
cfg.define('udpmasks', ['192.168*', ])
cfg.define('udphost', "localhost")
cfg.define('udpport', 5500)
cfg.define('udpallow', ["127.0.0.1", ])
cfg.define('udpallowednicks', ["#dunkbots", "#jsb", "dunk_"])
cfg.define('udppassword', "mekker", exposed=False)
cfg.define('udpseed', "blablablablablaz", exposed=False) # needs to be 16 chars wide
cfg.define('udpstrip', 1) # strip all chars < char(32)
cfg.define('udpsleep', 0) # sleep in sendloop .. can be used to delay pack
cfg.define('udpdblog', 0)
cfg.define('udpbots', [cfg['udpbot'] or 'default-irc', ])
```

The user can set these defined variable with the !udp-cfg commands:

```
usage:
!plug-cfg                    ->      shows list of all config
!plug-cfg key value          ->      sets value to key
!plug-cfg key                ->      shows list of key
!plug-cfg key add value      ->      adds value to list
!plug-cfg key remove value   ->      removes value from list
!plug-cfg key clear          ->      clears entire list
!plug-cfgsave                ->      force save configuration to disk
```

File creation

JSONBOT uses commandline options to generate the config files needed, but if you prefer to edit the configuration files you can run jsb-init that will create example irc and xmpp files into <datadir>/config/fleet/default-irc and default-sxmpp dirs. If you want more irc bots or xmpp bots you can copy the directories to new ones, but make sure to change the name variable of each config file (dir name needs to be the same as the name var in the config file).

For plugin configuration files, the are only created by running the !<plugin>-cfgsave command. So for the udp plugin it would be !udp-cfgsave.

3.1.3 INTRODUCTION

Getting Started

Get the latest tarball from:

```
http://code.google.com/p/jsonbot/downloads/list
```

Or install from mercurial:

```
hg clone http://jsonbot.googlecode.com/hg jsb
```

Configuration

Data and configuratiion files can be found in the datadir which defaults to ~/.jsb (or another directory if the -d option is used). You can either use commandline options to generate the configuration files or run:

```
./bin/jsb-init
```

to have them created for you. Look for the config/fleet directory for bot configuration files.

Shell console bot

In the bot dir run the ./bin/jsb command to start the console version of the bot.

```
22:44:11 dev@done:~/dev/0.7/5$ ./bin/jsb
JSONBOT 0.7 ALPHA1 CONSOLE
>
```

If you just want to execute 1 command on the bot, you can do that by giving it as an argument:

```
22:45:24 dev@done:~/dev/0.7/5$ ./bin/jsb list
available plugins: 8b, admin, alias, ask, chan, chatlog, choice,
controlchar, core, count, data, echo, fleet, foo, forward, gatekeeper,
gcalc, grep, hubbub, idle, ipcalc, irc, karma, kickban, koffie, learn, misc,
more, mpd, nickserv, not, outputcache, plug, ps, quote, relay, reload,
remind, restserver, reverse, rss, seen, shop, sort, tail, test, tinyurl, to,
todo, twitter, udp, uniq, url, urlinfo, user, userstate, watcher, weather,
welcome, wikipedia, xmpp
```

Shell IRC bot

Run the following to make the bot connect to an IRC server:

```
./bin/jsb-irc -o <userhost of owner> -s <server> -c \<channel>
```

A configuration file will be generated from the command line opions. If you already have the configuration files (generated by jsb-init) you can just use (without any options):

```
./bin/jsb-irc
```

Shell XMPP bot

Run the following to make the bot connect to an Jabber server:

```
./bin/jsb-xmpp -o <owner JID> -u <bot JID> -p <password> [-c <conference>] [-s <server>]
```

Or when configuration files already exist, just run the bot without arguments:

```
./bin/jsb-xmpp
```

Conference and server options are optional. Server is take from the -u option if not provided as a seperate option.

If you run the bot on a OpenFire server, use the –openfire option

Plugins

To see what plugins are available use the !list command. Use !help <plugin> to get help on a plugin.

```
23:03:00 dev@done:~/dev/0.7/5$ ./bin/jsb help misc
HELP ON MISC misc commands.,     !source - <i> show where to fetch the bot
source. </i> - examples: source,     !test - <i> give test response. </i> -
examples:  test
```

If a plugin is not responding try !plug-enable <plugin>:

```
23:03:05 dev@done:~/dev/0.7/5$ ./bin/jsb
JSONBOT 0.7 ALPHA1 CONSOLE
> !plug-enable misc
reloading and enabling jsb.plugs.core.misc
done - plug-enable misc
>
```

otherwise try !admin-boot:

```
23:05:46 dev@done:~/dev/0.7/5$ ./bin/jsb
JSONBOT 0.7 ALPHA1 CONSOLE
> !admin-boot
done - admin-boot
>
```

this will regenerate the callbacks and command indexes.

You can also use !apro <search> to search for commands:

```
23:01:23 dev@done:~/dev/0.7/5$ ./bin/jsb apro wiki
commands matching wiki: wikipedia
```

Commandline Help

All the programs have a –help option that shows a help page describing the command line options of the program.

3.1.4 LICENSE

3.1.5 PLUGINLIST

1. jsb.lib.periodical -=- provide a periodic structure.
2. jsb.plugs.common.8b -=- run the eight ball.
3. **jsb.plugs.common.alarm -=- the alarm plugin allows for alarms that message the user giving the** command at a certain time or number of seconds from now
4. jsb.plugs.common.ask -=- ask a user a question and relay back the response.
5. jsb.plugs.common.bugtracker -=- no docstring available
6. jsb.plugs.common.colors -=- use the morph to add color to selected words.
7. **jsb.plugs.common.controlchar -=- command to control the control (command) characters. The cc is a string** containing the allowed control characters.
8. jsb.plugs.common.feedback -=- give feedback on the bot to bthate@gmail.com. needs a xmpp server, so use jsb-xmpp or jsb-fleet.
9. jsb.plugs.common.forward -=- forward events occuring on a bot to another bot through xmpp.
10. jsb.plugs.common.gcalc -=- use google to calculate e.g. !gcalc 1 + 1
11. jsb.plugs.common.hubbub -=- the hubbub mantra is of the following: use the hb-register <feed-name> <url> command to register url and start a feed in in one pass.
12. jsb.plugs.common.idle -=- show how long someone has been idle.
13. jsb.plugs.common.imdb -=- query the imdb database.
14. jsb.plugs.common.ipcalc -=- IP subnet calculator. this module allows you to perform network calculations.
15. jsb.plugs.common.karma -=- maintain karma!
16. jsb.plugs.common.koffie -=- schenk wat koffie!
17. jsb.plugs.common.learn -=- learn information items .. facts .. factoids.
18. jsb.plugs.common.overflow -=- Grabs data for a StackOverflow user. You must enable this plugin first by running .. ;overflow-cfg enable 1
19. jsb.plugs.common.plus -=- plugin to query the Google+ API.
20. jsb.plugs.common.quote -=- manage quotes.
21. jsb.plugs.common.relay -=- relay to other users/channels.
22. jsb.plugs.common.remind -=- remind people .. say txt when somebody gets active
23. jsb.plugs.common.rss -=- the rss mantra is of the following:

(a) add a url with !rss-add <feedname> <url>

(b) use !rss-start <feed> in the channel you want the feed to appear

(c) run !rss-scan <feed> to see what tokens you can use .. add them with !rss-additem <feed> <token>

(d) change markup with !rss-addmarkup <feed> <markupitem> <value> .. see !rss-markuplist for possible markups

(e) check with !rss-feeds in a channel to see what feeds are running in a channel

(f) in case of trouble check !rss-running to see what feeds are monitored

(g) enjoy

24. jsb.plugs.common.search -=- access stats data from the spider plugin.
25. jsb.plugs.common.seen -=- nick tracking.
26. jsb.plugs.common.shop -=- maitain a shopping list (per user).
27. jsb.plugs.common.snarf -=- fetch title of url.
28. jsb.plugs.common.spider -=- Spider plugin.. Spider websites and makes an index into them.

    taken from http://code.activestate.com/recipes/576551-simple-web-crawler/

    – BHJTW 15-11-2011 Adapted for JSONBOT
29. jsb.plugs.common.tinyurl -=- tinyurl.com feeder
30. jsb.plugs.common.todo -=- manage todo lists per users .. a time/data string can be provided to set time on a todo item.
31. jsb.plugs.common.topic -=- manage topics.
32. jsb.plugs.common.tour -=- do a tour of the bot.
33. jsb.plugs.common.twitter -=- a twitter plugin for the JSONBOT, currently post only .. uses tweepy oauth.
34. jsb.plugs.common.urban -=- query urbandictionary
35. jsb.plugs.common.url -=- maintain log of urls.
36. jsb.plugs.common.urlinfo -=- Catches URLs on channel and gives information about them like title, image size, etc. Uses http://whatisthisfile.appspot.com/ via XMLRPC

    Example: 19:20 <@raspi> http://www.youtube.com/watch?v=9RZ-hYPAMFQ 19:20 <@bot> Title: "YouTube - Black Knight Holy Grail" 19:28 <@raspi> test http://www.raspi.fi foobar http://raspi.fi/wp-includes/images/rss.png 19:28 <@bot> 1. Title: "raspi.fi" Redirect: http://raspi.fi/ 2. Image: 14x14
37. jsb.plugs.common.watcher -=- watch channels. channels events can be of remote origin.
38. jsb.plugs.common.weather -=- show weather based on Google's weather API
39. jsb.plugs.common.wikipedia -=- query wikipedia .. use countrycode to select a country specific wikipedia.
40. jsb.plugs.common.yacy -=- no docstring available
41. jsb.plugs.core.admin -=- admin related commands. these commands are mainly for maintaining the bot.
42. jsb.plugs.core.alias -=- this alias plugin allows aliases for commands to be added. aliases are in the form of <alias> -> <command> .. aliases to aliases are not allowed, aliases are per channel.

43. jsb.plugs.core.all -=- output the outputcache to the user.
44. jsb.plugs.core.botevent -=- provide handling of host/tasks/botevent tasks.
45. jsb.plugs.core.cfg -=- this plugin manages various configuration settings.
46. jsb.plugs.core.chan -=- channel related commands.
47. jsb.plugs.core.choice -=- the choice command can be used with a string or in a pipeline.
48. jsb.plugs.core.core -=- core bot commands.
49. jsb.plugs.core.count -=- count number of items in result queue.
50. jsb.plugs.core.data -=- data dumper commands.
51. jsb.plugs.core.dispatch -=- this is the dispatch plugin that dispatches events to commands.
52. jsb.plugs.core.echo -=- echo typed sentences.
53. jsb.plugs.core.fleet -=- The fleet makes it possible to run multiple bots in one running instance. It is a list of bots. This plugin provides commands to manipulate this list of bots.
54. jsb.plugs.core.gatekeeper -=- gatekeeper commands.
55. jsb.plugs.core.grep -=- grep the output of bot comamnds.
56. jsb.plugs.core.ignore -=- no docstring available
57. jsb.plugs.core.irc -=- irc related commands.
58. jsb.plugs.core.misc -=- misc commands.
59. jsb.plugs.core.more -=- access the output cache.
60. jsb.plugs.core.nickcapture -=- nick recapture callback.
61. jsb.plugs.core.nickserv -=- authenticate to NickServ.
62. jsb.plugs.core.not -=- negative grep.
63. jsb.plugs.core.outputcache -=- outputcache used when reply cannot directly be delivered.
64. jsb.plugs.core.plug -=- plugin management.
65. jsb.plugs.core.rc -=- jsonbot resource files .. files with the .jsb extension which consists of commands to be executed.
66. jsb.plugs.core.remotecallbacks -=- dispatch remote events.
67. jsb.plugs.core.reverse -=- reverse pipeline or reverse <txt>.
68. jsb.plugs.core.size -=- call a size() function in every module in sys.modules
69. jsb.plugs.core.sort -=- sort bot results.
70. jsb.plugs.core.tail -=- tail bot results.
71. jsb.plugs.core.test -=- test plugin.
72. jsb.plugs.core.to -=- send output to another user .. used in a pipeline.
73. jsb.plugs.core.underauth -=- handle non-ident connection on undernet.
74. jsb.plugs.core.uniq -=- used in a pipeline .. unique elements.
75. jsb.plugs.core.user -=- users related commands.
76. jsb.plugs.core.userstate -=- userstate is stored in jsondata/state/users/<username>.
77. jsb.plugs.core.welcome -=- send welcome message.

78. jsb.plugs.core.xmpp -=- xmpp related commands.
79. jsb.plugs.db.birthday -=- manage birthdays
80. jsb.plugs.db.infoitem -=- information items .. keyword/description pairs
81. jsb.plugs.db.karma2 -=- karma plugin
82. jsb.plugs.db.lists -=- lists per user
83. jsb.plugs.db.quote2 -=- quotes plugin
84. jsb.plugs.db.todo2 -=- provide todo related commands
85. jsb.plugs.socket.autovoice -=- do voice on join
86. jsb.plugs.socket.chatlog -=- log channels to [hour:min] <nick> txt format, only logging to files is supported right now.

87) jsb.plugs.socket.confluence -=- confluence.py - jsonbot module for performing lookups on a confluence server Copyright 2011, Richard Bateman

Special thanks to Sean B. Palmer for his phenny module; many of the ideas for this were adapted from that plugin

http://inamidst.com/phenny/

88. jsb.plugs.socket.dns -=- do a fqdn loopup.
89. jsb.plugs.socket.fisheye -=- fisheye plugin.
90. jsb.plugs.socket.geo -=- This product includes GeoLite data created by MaxMind, available from http://maxmind.com/
91. jsb.plugs.socket.github -=- no docstring available

92) jsb.plugs.socket.irccat -=- irccat.py - jsonbot "irccat" Module Copyright 2011, Richard Bateman Licensed under the New BSD License.

Written to be used in the #firebreath IRC channel: http://www.firebreath.org

To test, set up the host and port, then use something like:

echo "@taxilian I am awesome" | netcat -g0 localhost 54321

echo "#channel I am awesome" | netcat -g0 localhost 54321

you can specify multiple users (with @) and channels (with #) by seperating them with commas. Not that with jabber, channels tend to be treated as users unless you set up an alias in your channel:

!irccat_add_alias #channel

93) jsb.plugs.socket.irccat2 -=- irccat.py - jsonbot "irccat" Module Copyright 2011, Richard Bateman Licensed under the New BSD License.

Written to be used in the #firebreath IRC channel: http://www.firebreath.org

To test, set up the host and port, then use something like:

echo "@taxilian I am awesome" | netcat -g0 localhost 54321

echo "#channel I am awesome" | netcat -g0 localhost 54321

you can specify multiple users (with @) and channels (with #) by seperating them with commas. Not that with jabber, channels tend to be treated as users unless you set up an alias in your channel:

!irccat_add_alias #channel

BHJTW - 28-02-2012 .. move to irccat2.py to use the normal irccat-cfg functions

94) jsb.plugs.socket.jira -=- jira.py - jsonbot module for performing lookups on a jira server Copyright 2011, Richard Bateman

Special thanks to Sean B. Palmer for his phenny module; many of the ideas for this were adapted from that plugin

http://inamidst.com/phenny/

95. jsb.plugs.socket.kickban -=- kickban functionality for IRC.
96. jsb.plugs.socket.lmgt -=- no docstring available
97. jsb.plugs.socket.markov -=- Markov Talk for Gozerbot

The Chain: (predictate) -> [list of possible words]

TODO:

- Propabilities
- Start searching for full sentence, not just the first ORDER_K words of a sentence

BHJTW:

- adapted for JSONBOT

98. jsb.plugs.socket.mpd -=- music player daemon control.
99. jsb.plugs.socket.ops -=- for op to work for a user the user must have the channel name in his/hers status .. use !user-addstatus <username> #channel normally the bot doesnt op nicks that join after a split to prevent floods, this can be disabled by using ops-cfg oponsplit 1
100. jsb.plugs.socket.restserver -=- implements a REST server, soon to be adapted for use with the jsb-tornado program.
101. jsb.plugs.socket.udp -=- the bot has the capability to listen for udp packets which it will use

     to /msg a given nick or channel.

     1. setup

        - run !udp-cfg udp 1 .. this enables the udp plugin
        - do !reload udp to enable the udp plugin
        - test with:

        ```
        echo "YOOO" | ./bin/jsb-udp -p <nick>
        ```

        - you can run !udp-cfgsave and edit ~/.jsb/config/jsb.plugs.socket.udp/config if need be.

     2. limiter

        on IRC the bot's /msg to a user/channel are limited to 1 per 3 seconds so the bot will not excessflood on the server. you can use partyudp if you need no delay between sent messages, this will use dcc chat to deliver the message. on jabber bots there is no delay

3.1.6 PROGRAMPLUGIN

```
For Annemiek, Kirsten, Danny and Doscha ;]
```

Intro

This document is here to help you program JSONBOT plugins. JSONBOT plugins are python files that contain commands and callbacks handlers that receive a bot and a event as arguments and can act upon these.

First you must know that plugins you write yourself should be placed in one of the ~/.jsb/myplugs subdirs. The bot will detect if code has changed in these directories: "socket" for plugins that run on shell bots, "common" is a remains of the old GAE code stuff.

Second thing to remember is to call the !admin-boot command if you add new commands or callbacks so the bot can recreate its command and callback indexes (used to load a plugin on demand).

Example

Very basic example of a JSONBOT command handler is:

```
from jsb.lib.commands import cmnds

def handle_demo(bot, event):
    """ do the command. """
    event.reply("This is a demo text.")

cmnds.add("demo", handle_demo, "GUEST")
```

This example implements a command !demo which just gives a reply. The jsb.plugs.core.dispatch plugin termines whether a event triggers a command or not. Basically that plugin checks for the cc (control char) in the event.txt attribute and if it does, it will check permissions and call the handle_demo function with the bot and event paramaters passed along.

To let a plugin react on events, you can add a same kind of handler to the callbacks object, and have it called when a particular event takes place on the bot. Example:

```
from jsb.lib.callbacks import callbacks

def handle_democb(bot, event):
    bot.say(event.channel, "%s joined the channel." % event.nick)

callbacks.add("JOIN", handle_democb)
```

The callback implemented here is registered on JOIN events and thus gets called on JOIN received by the bot (JOIN is a IRC message) with both bot and the JOIN event passed on as arguments. I used bot.say() here to show that the bot can be used for push operations as well (where as reply() always responds to the event in question. The handler also shows that the event contains data about the event that took place (event.nick, event.channel).

Data Flow

So plugin are about handler and handlers are about combining bot and events together. Basically the data flow is like this:

```
* bot connects to server
* bot gets data back (most of the times this is a string)
* bot creates an event from the data received
* bot calls the handler with itself and the event as arguments
* handler does something with the event and sends a response back
* bot can also push data to a channel, think RSS feeds
```

For every network that JSONBOT supports there need to be both bot and event classes that support the connecting to the network and the contruction of the events based upon the data that is received. These classed are in JSONBOT 0.7 moved into there own sub package: *jsb.drivers*, so writing new bot-event combinations becomes easier.

Using Events

So events are at the core of the JSONBOT handlers that you can write for your plugin. The basic interface of these events are defined in *jsb.lib.eventbase*, i will summ up the most important method and atributes:

methods:

```
def reply(self, txt, result=[], event=None, origin="", dot=u", ",
          nr=375, extend=0, *args, **kwargs):
    """
        use this to reply to the event with some text. the result list
        can be pipelined to other commands and uses the dot paramater to
        concat the result list into the string send back to the user. If
        you want to use the result of the plugin be sure to pass on the
        event with the event parameter as results are aggregated in the
        event itself.

        examples:

        1) event.reply("demo text")
        2) event.reply("results: ", resultlist)
        3) event.reply("results: ", resultlist, dot=" | ")
        4) event.reply("results: ", resultdict)
        5) event.reply("results: ", resultlist, event=self)

    """

def missing(self, txt):
    """
        use this if the input from the user is not correct.

        example: event.missing("<nick>")

    """

def done(self):
    """ this just prints "done - command. """

def iscmnd(self):
    """ checks whether an event is a command. returns comandstring. """

def bind(self, bot=None, user=None, chan=None):
    """
        use this to bind bot, user and channel object into the event.
        note that you dont have to provide arguments but you can if you
        have these objects already available.

    """

def ready(self, finish=True):
    """
        use this when you are finished handling the event. JSONBOT tried
        to automatically do this for you, but sometimes you need to do this
```

```
        yourself.

    """
```

attributes:

```
self.stop = False # set this if you want any processing of the
                    # events stopped

self.bonded = False # check this to see if EventBase.bind() is
                    # already called
```

after bind():

```
# the channel object containing data relataed to the channel
self.chan = ChannelBase(self.channel, bot.cfg.name)

# the user of the bot giving the command
self.user = user or bot.users.getuser(target)
```

There are more methods defined on the EventBase class but these are the most common used by plugin programmers. The bot will only execute commands when the event has bind() called on it so the user and chan attributes will be properly set.

3.1.7 UPGRADE

From 0.4 to 0.5

A number of changes went into JSONBOT.

First i needed to change the stripname() function to make used filename compatible with appengine. This means that most json data lives under a different filename. To fix this i made a !chan-upgrade command which tries to find the old filename and merge the data of that into the new file.

Second: I needed to make the bot debian compatible so we needed to be able to start the bot from a different datadir (currently the local gozerdata dir). This was a big rewrite as access to the datadir var in jsb/datadir needed to be moved from import time to runtime as it can be set in the using program now .. use the -d option to set a different datadir as opposed to the default which is ~/.jsb now

Third: forwarding was a bit too open so i added a allowwatch attribute that determines to what channel a forwarded event can be send. This way the sending side can control to whom the events may go. Use !chan-allowwatch to add remote channel to the allowlist.

to recap:

1. move your data dir into ~/.jsb or use -d gozerdata
2. run !chan-upgrade to update you channel data
3. use !chan-allowwatch on the sending side to allow channels on the receiving side if you use forwarding

From 0.5 to 0.6

The bot got refactored beyond sanity, so seperate upgrade scripts and programs are provided to upgrade to 0.6

- on shell

use the ./bin/jsb-upgrade program, this will copy your data from ~/.jsonbot to ~/.jsb

- on GAE

copy over the files from ~/jsonregs to ~/jsbregs (cp -Ra)

Best is to upload the new bot into its own version (edit ~/jsbregs/<appid>/app.yaml)

Run the !admin-upgrade command, you may need to run this multiple times until the bot says it's done upgrading, check if everything is still working and switch to the new version.

you might need to run !chan-upgrade to update your channel.

From 0.6 to 0.7

in 0.7 a number of things changed, most of them relating to how programmers use the bot code.

First: bot/event classes were move into there own new namespace namely jsb.drivers. So now you need to access for examples jsb.drivers.irc.bot etc.

Second: jsb.lib.version is now jsb.version

Third: the bot was rewritten to use the config variables directly. Instead of doing bot.name = bot.cfg.name you should now use the bot.cfg.name. This is so changes to the config can be reloaded on runtime (i'm working on the config reload code).

4th: the ~/.jsb/myplugs dir got seperate into multiplie subdirs namely: myplugs.socket (shell), myplugs.common (both), myplugs.gae (GAE). You need to move your plugins into one of those.

Thats it ! No changes needed to the filesystem so no !upgrade command to run. It might be possible that a bot's config file (see ~/.jsb/config/fleet) is not set, if so add it to the config file and restart.

CHAPTER

FOUR

URLS

- Source code: http://jsonbot.googlecode.com
- web: http://jsonbot.org:10102 (web)
- jabber: jsonbot@jsonbot.org

CHAPTER

FIVE

NOTES

5.1 notes

5.1.1 0.7 RELEASE NOTES

Hello world, greetings to all and everybody on this little planet ;] Today I am releasing version 0.7 of JSONBOT, hope you like it. I want to dedicate this release to Annemiek, Kirsten, Danny and Doscha, i would not have a live without you. **changes in this release**

- we got jsonbot.org running .. see http://jsonbot.org ;]
- convore support
- refactored core
- reloadable config files
- revamped web console
- resource files (contain commands the bot can execute)
- file change detection for myplugs plugins
- rebooting is fixed
- fixed relaying in jabber conference rooms
- added color.py plugin to color certain words
- added geo.py, googletranslate.py and imdb.py (thnx melmoth)
- chatlog plugin now uses the logging module .. log file rotates every day
- many other bugfixes

If you have programmed your own plugin see http://jsonbot.org/handbook/UPGRADE.html for upgrade notes.

Todo

1. fix runtime setting of loglevel
2. add flood control
3. docs docs docs docs docs
4. fix bugs

see http://code.google.com/p/jsonbot/issues/list

Source

- tarball - http://jsonbot.googlecode.com
- mercurial - http://jsonbot.googlecode.com/hg
- github - https://github.com/jsonbot

Demo

- webconsole - http://jsonbot.appspot.com
- xmpp - jsonbot@jsonbot.org (shell) and jsonbot@appspot.com (GAE)
- IRC - jsonbot on irc.freenode.net
- Convore - https://convore.com/convore-8/welcome-to-convore/ relaying with #convore on irc.freenode.net

Docs

- new jsonbot.org site .. http://jsonbot.org
- GAE backup docs .. http://jsonbot.appspot.com/docs

Contact

- twitter: https://twitter.com/#!/jsonbot
- facebook: http://tinyurl.com/jsonbot
- email: bthate@gmail.com
- IRC: dunker in channel #dunkbots / irc.freenode.net
- xmpp: bthate@gmail.com and bart@jsonbot.org

About

JSONBOT is a remote event-driven framework for building bots that talk JSON to each other over XMPP.

This distribution provides bots built on this framework for console, IRC, XMPP and Convore for the shell and WWW and XMPP for the Google Application engine.

JSONBOT is all of the following:

- a shell console bot
- a shell IRC bot
- a shell XMPP bot
- a shell Convore bot
- a Web bot running on Google Application Engine
- a XMPP bot running on Google Application Engine

- a Google Wave bot running op Google Application Engine
- the XMPP bots are used to communicate between bots
- plugin infrastructure to write your own functionality
- event driven framework by the use of callbacks

Install

You dont need to run the bot on GAE when you just want to use the shell bots of JSONBOT. JSONBOT can best be run from the bot dir, the bot is self contained and has all the dependancies that are needed:

- "hg clone http://jsonbot.googlecode.com/hg mybot" or download and untar the tarball.
- cd into the bot dir and run "./bin/jsb" .. if the bot is working correctly you will get the console version of JSONBOT
- same can be done for "./bin/jsb-xmpp", "./bin/jsb-convore" etc. .. check the bin dir for programs you can start
- try the –help option to a program to see what command line options are available.
- you DONT need root for this

Ofcourse you can always run "python setup.py install" or "easy_install -U jsb" when you do want to install the bot globaly. Debian packages are on their way, but might still take time as the ftpmeisters need to approve ;]

Thats it ! hope like this version of JSONBOT and dont forget to have fun with it ;]

5.1.2 RELEASE 0.80

Welcome JSONBOT 0.80 !!

Been too quiet on reporting on all that has changed in JSONBOT, so i'll revamp that here comparing with version 0.7.

about

JSONBOT is a chatbot that can take commands and react to events on the network it is connected to (IRC, XMPP, WEB mostly). Push functionality is also provided (think RSS feeds to your IRC channel or XMPP conference). It is possible to program your own plugins to create custom functionality.

source/docs

see http://jsonbot.org and http://jsonbot.googlecode.com

make backup first

I added the jsb-backup program, please run this before starting the 0.80 bot. It will make a backup of your datadir into ~/jsb-backups

changes

- GAE is no longer part of the standard distribution, as that is aimed at shell users as of 0.80 - use the mercurial repo if you want to use the GAE part of the bot
- web console is now supported on shell - use the jsb-tornado program to launch a tornado web server bot on port 10102
- jsb-xmpp now supports OpenFire - use –openfire option to enable this
- todo now uses per user databases instead of per channel - use the -c option to the todo command to show the channel todo
- learn items are not global per default - use !learn-toglobal to copy local learn data to the global learndb
- relay plugins has been rewritten to use bot.cfg.name as well - means that relays need to be created again
- jsb-udpstripped program has been added that can be used to send udp data to the bot without the need of making config files (copy and edit it)
- add fulljids = 1 to your xmpp bot config (most of the times in ~/.jsb/config/fleet/default-sxmpp/config) to enable full JID discovery in xmpp conference rooms (non anonymous)

and:

- lots of new plugins .. see !list ;]
- lots of bug fixes - thnx everybody for reporting them
- still lots of things to fix at

03:35 < jsonbot> tracker is http://code.google.com/p/jsonbot/issues/list

If you find any problems or have feature request please post that on the tracker url above.

Or try @botfather on #dunkbots on irc.freenode.net ;]

5.1.3 JSONBOT 0.84 RELEASE NOTES

please run ./bin/jsb-backup command to backup your current data directory.

1. experimental features:
   - FiSH support on IRC, thanks to Frank Spijkerman
   - sleekxmpp driver for xmpp bots, use ./bin/jsb-sleek
2. further changes
   - new boot code - bot now tries to detect changes in the code and recreates the dispatch tables when needed.
   - reconnect code should work better, disconnect should now be properly detected
   - updated tornado to version 2.2 - needed for websockets new style
   - database support for sqlite and mysql is now available, see ~/.jsb/config/mainconfig to enable it.
   - default control character is now ;
3. still todo

- there are problems with setting the control character properly
- yes .. docs still need to be written
- yes .. bugtracker needs attention

CHAPTER

SIX

PLUGINS

CHAPTER

SEVEN

RUNNING A DEVELOPMENT VERSION OF THE BOT

first checkout the main bot from the mercurial repository

```
hg clone http://jsonbot.googlecode.com/hg mybot
```

now you can run the programs in the bin directory with the `./bin/<program>` command. try ./bin/jsb for the console app

CHAPTER

EIGHT

JSB CODE

CHAPTER

NINE

JSB.PLUGS.CORE CODE

CHAPTER

TEN

JSB.PLUGS.COMMON CODE

CHAPTER

ELEVEN

JSB.PLUGS.SOCKET CODE

CHAPTER

TWELVE

JSB.PLUGS.MYPLUGS CODE