Djabot – Python Jabber Bot

+ *Echo* & *HTTP-like JSON API* Bots

+ *djabotd* – Django daemon command

a presentation by Marek Kuziel <marek@tracklr.com>

Marek Kuziel <marek@tracklr.com>

#my #experience #in #hashtags #... #opensource #php #python #java #javascript

#html #xhtml #docbook #xslt #css #linux #nginx #postgresql #mysql #memcache

#apache #subversion #mercurial #phing #pear #deployment #scaling #automation

#tomcat #javaws #arduino #8051 #Z80 #assembly #GWT #drupal #django #wsgi

#jabber #xmpp #thegimp #openid #oauth #ecommerce #zencart #titanium #mobile

#tracklr #tracklr2mindmap #djauthy #djabot

Jabber/XMPP

Good security – TLS support

Decentralized – You can run your own Jabber/XMPP server

Open – Standards, community, code

What Is a Jabber Bot?

An unmanned Jabber/XMPP client

Runs as a program on a computer (ie. as a Linux daemon)

Processes incoming messages from other clients

Introducing Djabot

Python Jabber client – TLS support, extensible (bots, auth)

Example bots: *echo*, *HTTP-like JSON API*

djabotd – Django command that implements Djabot and allows you to run bots for your Django based app as a daemon

Djabot – More Information

Requires: PyXMPP, M2Crypto, libxml2-python, dnspython

Allows you to provide your own auth callback function to protect your bots

Allows you to write as many bots as you like and configure your instance to use them

Djabot – Example Code

```
#/usr/bin/env python

from pyxmpp.all import JID
from djabot.djabot import Client

if __name__ == '__main__':
    def auth_check(requestSenderJID):
        return True          # reply to any sender / do not check sender's JID at all
    BOTS = ['djabot.bots.echo']
    c=Client(JID('USERNAME'), 'PASSWORD', 'tls_noverify', BOTS, auth_check)
    c.connect()
    c.loop(1)
```

Djabot – Example Echo Bot Code

```
from djabot.bots.base import Djabot as Base

class Djabot(Base):

    NAME = 'djabot-echo-bot'

    def command_echo(self, inputs):
        return inputs['body']

    def status(self, inputs):
        return 'echoed back: echo %s' % inputs['body']
```

Djabot commands – *inputs*

djabot processes received Jabber msg & passes the following dictionary as *inputs* parameter to any command that you define:

```
inputs = {
            'subject': subject,    # subject of received msg
            'body': body,          # body of received msg
            'body_json': None, # JSON object (if possible to load body as JSON)
            'jid': sender,         # JID of sender
            }
```

Jabber/XMPP HTTP-like JSON API

What? request/response JSON API over Jabber/XMPP that mimics HTTP (status code; PUT|POST|GET|DELETE verbs)

Why? I prefer “chatting” with my app instead of RESTful HTTP based approach

Seriously. Why? Faster & easier implementation of APIs

Core properties of the API protocol

HTTP status *code* and *message* property in response

transaction property to uniquely identify sent/received messages

api property for version namespace of given *action* property in request

“GTalk – djabot” example chat about **GET**

me: {"action": "GET", "api": "1.0", "data": {"id": "123"}, "object": "OBJECT", "transaction": "1234567890"}

djabot: {'message': u'GET OBJECT ID 123 OK', 'code': '200', 'data': '{"id": "123", "example_data": "EXAMPLE"}', 'transaction': u'1234567890'}

“GTalk – djabot” example chat about **PUT**

me: {"action": "PUT", "api": "1.0", "data": {"id": 123}, "object": "OBJECT", "transaction": "1234567890"}

djabot: {'message': u'PUT OBJECT ID 123 OK', 'code': '200', 'transaction': u'1234567890'}

“GTalk – djabot” example chat about **POST**

me: {"action": "POST", "api": "1.0", "data": {"id": 123}, "object": "OBJECT", "transaction": "1234567890"}

djabot: {'message': u'POST OBJECT ID 123 OK', 'code': '200', 'transaction': u'1234567890'}

“GTalk – djabot” example chat about **DELETE**

me: {"action": "DELETE", "api": "1.0", "data": {"id": 123}, "object": "OBJECT", "transaction": "1234567890"}

djabot: {'message': u'DELETE OBJECT ID 123 OK', 'code': '200', 'transaction': u'1234567890'}

Demo HTTP-like JSON API Bot

djabot@tracklr.com

Try it :-)

Djabot – Source Code

Released 2010/07/17 under BSD License

http://bitbucket.org/vshivak/djabot/

Feedback or contributions:
Marek Kuziel <marek@tracklr.com>

Thank You!

Marek Kuziel, Christchurch, New Zealand, July 2010

This presentation can be found at **http://tracklr.com/tm/presentations/tracklr-djabot.pdf**