API Documentation

April 22, 2005

Contents

Contents **1**

1 Package fraendz **2**
1.1 Modules 2

2 Module fraendz.admin **3**
2.1 Functions 3

3 Module fraendz.chatserver **6**
3.1 Functions 6
3.2 Variables 6
3.3 Class ChatServer 7
3.3.1 Methods 7
3.3.2 Class Variables 7

4 Module fraendz.config **8**
4.1 Functions 8
4.2 Variables 8

5 Module fraendz.security **10**
5.1 Functions 10
5.2 Variables 12

6 Module fraendz.template **13**
6.1 Functions 13
6.2 Variables 14

7 Module fraendz.userio **16**
7.1 Functions 16
7.2 Variables 19

1 Package fraendz

Modules associated with the fraendz-system

1.1 Modules

- **admin**: admin.py - part of fraendz...
  (Section 2, p. 3)
- **chatserver**: chatserver.py – chat-server for the fraendz-system...
  (Section 3, p. 6)
- **config**: config.py - part of fraendz...
  (Section 4, p. 8)
- **security**: security.py - part of fraendz
  * includes security-related stuff...
  (Section 5, p. 10)
- **template**: template.py - component of fraendz...
  (Section 6, p. 13)
- **userio**: userio.py - part of fraendz
  (Section 7, p. 16)

2 Module fraendz.admin

```
admin.py - part of fraendz

 * includes function needed by the admin user
```

2.1 Functions

addAdminableUser(*auser*)

* add a user to the ADMIN_FILE

addDefaultAdminScope(*scope*, *user*, *code*, *pw*)

```
* appends admin-specific scope variables to
  a given dictionary
```

changeAdminPassword(*newpw*)

* 0 for success * 1 for failure

checkAdminPassword(*enc_pw*)

* 0 if password is wrong * 1 if password is correct

checkAdminRssID(*id*)

returns username if id is valid * performed checks:

- user registered?
- user adminable?
- admin-password correct?

else 0 is returned

delAdminableUser(*duser*)

* remove user's entry in the ADMIN_FILE

getAdminableUsers()

* returns list of all adminable users (defined in ADMIN_FILE)

getAllScriptNames()

```
* returns a list of names of all scripts in the WEB_CGI_ROOT and
  the WEB_CGI_ROOT+admin directories
* scriptname is returned as complete path
```

getDefinedVariables(*location*)

```
* returns a list of all found variables in a location
        location:
         fraendz.module[:function]
* return:
        [(name, description, value), ...]
```

getLastLogs(*n*)

return a list of the last n logs -> returns: [(datestring, event), ...]

getPossibleParameters(*script*)

```
* takes script as the full path to a script and tries to extract
  the possible parameters from the header
* the typical header of a script looks like:
   name_of_script.py -- usage
   * description

   PARAMTERS:
        p1 - something
        p2 - something etc
```

standardAdminSecurityChecks(*form*)

```
* called at the beginning of every front-end script to perform the standard security
  checks, with only the CGI-form as parameter
* the script extracts parameters and performs necessary checks
   checks performed:
        - are all necessary keys present?
        - is the user registered?
        - does the user-provided stamp coincide with the stored stamp?
        - timeout?
        - is user adminable?
        - is the admin-password correct?
* if all checks go well, function returns the tuple:
        (scope, user, code, adminpw)
```

useradd(*user*, *pw*)

* add new user to the system
  - make entry in central user file
  - make user home-directory
  - if RSS is enabled, add user to the htpasswd file

userdel(*deluser*, *delmode*)

```
* deletes the selected user
* if mode == 1:
        - delete only registration
  elif mode ==2:
        - delete registration and complete user-directory
```

3 Module fraendz.chatserver

```
chatserver.py -- chat-server for the fraendz-system

 * once started, runs in a daemon mode until a timeout is reached

REQUESTS:
        a request consists of a keyword, followed by data or
        nothing (since the client is waiting)
        KEYWORDS:
            AUTH        - client sends authentification ## not implemented yet!!##
            POST        - message is following (first item is username)
            GET         - client is waiting for list
            QUIT        - shutdown the chat-server
            STATUS      - client is asking the status of the webserver
```

3.1 Functions

handler(*csocket*, *s*)

called as a thread for each connection

listen(*s*)

main()

3.2 Variables

Name	Description
ALLOWED_REQUESTS	**Value:** `['GET', 'POST', 'QUIT', 'STATUS']` *(type=list)*
ALLOWED_USERS	**Value:** `['thias', 'lilli', 'christoph', 'kristin', 'jemand', 'jemand2']` *(type=list)*
BUFFER	**Value:** `1024` *(type=int)*
CONFIRM	**Value:** `'confirm'` *(type=str)*
conList	**Value:** `[]` *(type=list)*
quit	**Value:** `0` *(type=int)*
REJECT	**Value:** `'reject'` *(type=str)*
u	**Value:** `[['thias', 2], ['lilli', 1], ['christoph', 1], ['kristin', 0], ['jemand', 1],...` *(type=list)*

3.3 Class ChatServer

3.3.1 Methods

__init__(*self*)

constructor
- initializes logfile
- initializes serversocket

action(*self*)

close(*self*)

destructor
- closes logfile
- closes socket

getMessages(*self*)

listen(*self*)

log(*self*, *msg*)

timeout(*self*)

update(*self*, *user*, *msg*)

3.3.2 Class Variables

Name	Description
ACCEPT_CONNECTION	**Value:** `['127.0.0.1']` *(type=list)*
buffer	**Value:** `1024` *(type=int)*
chat	**Value:** `[]` *(type=list)*
f	**Value:** `''` *(type=str)*
idle_since	**Value:** `0` *(type=int)*
LOG_DIR	**Value:** `'/Users/thias/web/internal/users/chat_log/'` *(type=str)*
logfile	**Value:** `''` *(type=str)*
MAX_RUNNING_TIME	**Value:** `86400` *(type=int)*
SERVER_ADDRESS	**Value:** `'localhost'` *(type=str)*
SERVER_PORT	**Value:** `21567` *(type=int)*
start_time	**Value:** `0` *(type=int)*
TIMEOUT	**Value:** `1800` *(type=int)*

4 Module fraendz.config

```
config.py - part of fraendz

* parses the config-file and prepares access to central
  variables/templates
```

4.1 Functions

parse_configfile(*file='../config/fraendz_config'*)

parse config file and return contents in a dictionary

4.2 Variables

Name	Description
ADMIN_FILE	**Value:** '/Users/thias/web/internal/.admin' *(type=str)*
CGI_ROOT	**Value:** '/Users/thias/web/internal/htdocs/cgi-bin/' *(type=str)*
CHAT_ACCEPT_CONNECTION	**Value:** ['127.0.0.1'] *(type=list)*
CHAT_SERVER_ADDRESS	**Value:** 'localhost' *(type=str)*
CHAT_SERVER_LOGDIR	**Value:** '/Users/thias/web/internal/users/chat_log/' *(type=str)*
CHAT_SERVER_PORT	**Value:** 21567 *(type=int)*
DEBUG	**Value:** 1 *(type=int)*
LIB_DIR	**Value:** '/Users/thias/web/internal/lib/' *(type=str)*
LOGFILE	**Value:** '/Users/thias/web/internal/logbook.dat' *(type=str)*
MAX_RSS_DEV	**Value:** 10 *(type=int)*
MAX_RSS_FORUM	**Value:** 10 *(type=int)*
MAX_RSS_LOG	**Value:** 30 *(type=int)*
MAX_RSS_MSG	**Value:** 10 *(type=int)*
MAX_RSS_NEWS	**Value:** 10 *(type=int)*
PIC_ROOT	**Value:** '/Users/thias/web/internal/pics/' *(type=str)*
RSS_ADMIN_DIR	**Value:** '/internal/admin/feed' *(type=str)*
RSS_GROUP_DIR	**Value:** '/internal/groupfeed' *(type=str)*
RSS_USER_DIR	**Value:** '/internal/userfeed' *(type=str)*
SKINS	**Value:** ['default', 'traditional', 'text'] *(type=list)*
SYSTEM_ROOT	**Value:** '/Users/thias/web/internal/' *(type=str)*
SYSTEM_USERFILE	**Value:** 'info.dat' *(type=str)*
TEMPLATE_DIR	**Value:** '/Users/thias/web/internal/templates/default' *(type=str)*
TEMPLATE_ROOT_DIR	**Value:** '/Users/thias/web/internal/templates' *(type=str)*
TIMEOUT	**Value:** 1800 *(type=int)*
USER_ADRESSFILE	**Value:** 'personal.dat' *(type=str)*
USER_DIRS	**Value:** '/Users/thias/web/internal/users/' *(type=str)*
USER_FILE	**Value:** '/Users/thias/web/internal/.users' *(type=str)*

continued on next page

Name	Description
WEB_CGI_ROOT	**Value:** `'/internal-bin/'` *(type=`str`)*
WEB_PIC_ROOT	**Value:** `'/internal/pics/'` *(type=`str`)*
WEB_ROOT	**Value:** `'/internal/'` *(type=`str`)*

5 Module fraendz.security

```
security.py - part of fraendz

 * includes security-related stuff...
```

5.1 Functions

changeUserPw(*user*, *newpw*)

```
* changes the entry in the .users file to the new, user-provided
  password
```

checkIfUserRegistered(*user*, *pw*)

```
* checks if a given user with an encrypted password
  is registered in the users-file
* returns 1 if an error occurs,
  returns 0 if everything is ok
```

checkRssID(*id*)

checks if the id provided by user is a valid RSS id and extract the user name out of it if so

checkStamp(*user*, *stamp*)

```
* checks if the stamp (probably obtained by decode()) corresponds to the
  login-stamp of the user
```

checkStampTime(*user*, *stamp*)

* checks if the timeout is reached or not

cleanUpUsers()

* delete old stamps of users who forgot to log out * this function is called each time a new user logs in

decodePw(*code*)

* decodes the cryptic code into a password/stamp pair

delStamp(*user*)

* logout -> delete the timestamp

encodePw(*user*, *pw*)

```
* takes the password and the user stamp and combines it to a cryptic code
  which is used to forward the user
```

filterHTML(*thing*)

```
* takes a list, a dictionary, or a string and returns the same
  thing without HTML-Tags (<something>)
* is used to prevent users from entering HTML-code in the entry
  fields
```

firstVisit(*user*)

* checks if user has already been registered

getAllUsers()

```
* same as getUserList(), but only a list of all usernames
  is returned
```

getOnlineUsers()

returns a list of all users who are currently online

getRestTime(*user*)

* returns the remaining time for the user-session

getStamp(*user*)

* returns stamp of user

getUserList()

```
* returns a list of all users currently registered
* entries are of the form (username, [0, 1, 2]), where
  0 stands for not registered
  1 stands for registered
  2 stands for online
```

logbook(*entry*)

* writes things in the logfile

shutdownChatServer()

guess what it's doing...

stampUser(*user*)

* makes a time stamp in a file in the user directory * an old one will be overwritten

standardSecurityChecks(*form*)

```
* standard security checks before a script reveals information
   checks performed:
        - all necessary form-parameters present?
        - user registered?
```

startupChatServer(*user*)

check if the chat-server is running, if not start it

substituteHTML(*thing*)

```
* takes a list, a dictionary, or a string and returns the same
  thing with HTML-Tags exchanged to &gt; and &lt;
```

touch(*file*, *mode*=511)

* creates a file and sets the permissions to mode

userAdminable(*user*)

```
* checks the .admin file, if the user has the priviledge to become
  the admin
```

userExists(*user*)

* returns 1 if the user exists in .users-file

userOnline(*user*)

* returns 1 if user is online (stamp.log exists)

5.2 Variables

Name	Description
ALPHABET	**Value:** `'ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrst-uvwxyz0123456789'` *(type=`str`)*
DELIMITER	**Value:** `'$'` *(type=`str`)*
RANDOM_LETTERS	**Value:** `5` *(type=`int`)*

6 Module fraendz.template

```
template.py - component of fraendz

 * contains template definitions and insertion
   functions
```

6.1 Functions

defineDefaultScope(*scope*={}, *code*='', *user*='')

```
* returns a dictionary containing default-Values for scope variables,
  which cannot be defined as global variable in template.py (due to
  variable nature)
* function takes a dictionary (by default empty) and appends the
  values
```

formatTime(*stamp*)

```
* takes a time stamp as provided by time.time() and returns a nicely
  formatted time
* current format: 'dd-mm-yy hh:mm'
```

fortune()

getPager(*scope*, *items*, *item_per_page*, *current_page*, *template*)

```
* returns the HTML-code for a pager
  takes:
        - scope         - the default scope dictionary
        - items         - the complete item set from which displayed items will be selected
                items should be a list
        - item_per_page - how many items per page?
        - template      - use which template?
        - extra         - an additional argument
* returns:
        (selected_items, pagerstring)
```

getTemplate(*template*, *s*={'LINK_FRAENDZ': 'http://www.vug.de/f', 'getOnlineUsers':...)

```
* new function to provide the template-system
* same method as in earlier version, but more
  sophisticated
* e.g. recursive template includings possible
- see documentation for details
```

getUserImageLink(*user*, *code*, *whom*)

* returns a link to the user's personal picture

printHTMLPart(*which*='upper',
s={'LINK_FRAENDZ': 'http://www.vug.de/f', 'getOnlineUsers':...)

```
prints either the upper or the lower part of the
standard HTML-Header
* the additional dictionary-argument is a bit tricky,
  it is used to make VARIABLES defined in the main program
  available in this module (in the eval-statements)...
  see py4cs page 404
-- with version 0.4 a new template system is to be used
       obsoleted here!
```

6.2 Variables

Name	Description
CURRENT_DATE	**Value:** 'Fri Apr 22 01:13:39 2005' *(type=str)*
FAVICON	**Value:** '/internal/fr1.ico' *(type=str)*
FORTUNE_FILE	**Value:** '/Users/thias/web/internal/lib/fortune.dat' *(type=str)*
INTERNAL_MSG	**Value:** '\nLiebe(r) %s, \n\ndu bist jetzt also im Freundessystem und hast sicher scho... *(type=str)*
LINK_ADMINLOGIN_PLAIN	**Value:** '/internal-bin/adminlogin.py' *(type=str)*
LINK_CHAT_PLAIN	**Value:** '/internal-bin/htmlchat.py' *(type=str)*
LINK_FORUM_PLAIN	**Value:** '/internal-bin/forum.py' *(type=str)*
LINK_FRAENDZ	**Value:** 'http://www.vug.de/f' *(type=str)*
LINK_HOME	**Value:** 'http://www.psych.uni-goettingen.de/home/ihrke/-download/en/inf/fraendz.html' *(type=str)*
LINK_HOMEPAGE_PLAIN	**Value:** '/internal-bin/homepage.py' *(type=str)*
LINK_HTMLCHAT	**Value:** '/internal-bin/htmlchat.py' *(type=str)*
LINK_INTERNAL	**Value:** '/internal/' *(type=str)*
LINK_LOGIN	**Value:** '/internal/login.html' *(type=str)*
LINK_MESSAGE_PLAIN	**Value:** '/internal-bin/message.py' *(type=str)*
LINK_NEWS_PLAIN	**Value:** '/internal-bin/news.py' *(type=str)*
LINK_PORTAL_PLAIN	**Value:** '/internal-bin/portal.py' *(type=str)*
LINK_REGISTER_PLAIN	**Value:** '/internal-bin/register.py' *(type=str)*
LINK_ROOT_EXTERNAL	**Value:** 'http://www.stud.uni-goettingen.de' *(type=str)*
PIC_ADMIN	**Value:** '/internal/pics/big_brother.jpg' *(type=str)*
PIC_ANTENNE	**Value:** '/internal/pics/antenne.gif' *(type=str)*
PIC_BOOK	**Value:** '/internal/pics/book_klein.png' *(type=str)*
PIC_CHAT	**Value:** '/internal/pics/chat.gif' *(type=str)*
PIC_CHAT_MANAGEMENT	**Value:** '/internal/pics/chat_management.gif' *(type=str)*
PIC_CORNER_LOW	**Value:** '/internal/pics/corner_down.jpg' *(type=str)*
PIC_CORNER_UP	**Value:** '/internal/pics/corner_up.jpg' *(type=str)*
PIC_DISCUSSION	**Value:** '/internal/pics/discussion.jfif' *(type=str)*
PIC_DOWNARROW	**Value:** '/internal/pics/down.gif' *(type=str)*
PIC_FORUM_MANAGEMENT	**Value:** '/internal/pics/discussion_management.jpg' *(type=str)*
PIC_FRIENDZ_LOGO_BIG	**Value:** '/internal/pics/fr7.png' *(type=str)*

continued on next page

Name	Description
PIC_FRIENDZ_LOGO_EXTRA_BIG	**Value:** '/internal/pics/fr6.png' *(type=str)*
PIC_HOUSE	**Value:** '/internal/pics/house.gif' *(type=str)*
PIC_LEFTARROW	**Value:** '/internal/pics/left.gif' *(type=str)*
PIC_LOGFILE	**Value:** '/internal/pics/log.png' *(type=str)*
PIC_MESSAGEWRITE	**Value:** '/internal/pics/schreib.gif' *(type=str)*
PIC_NEWS	**Value:** '/internal/pics/news.jfif' *(type=str)*
PIC_NEWS_MANAGEMENT	**Value:** '/internal/pics/news_management.jpg' *(type=str)*
PIC_PARAMETER	**Value:** '/internal/pics/param.png' *(type=str)*
PIC_PORTAL	**Value:** '/internal/pics/tuer.gif' *(type=str)*
PIC_RSS_FEED	**Value:** '/internal/pics/rss_feed.gif' *(type=str)*
PIC_RSS_FEED_EXTERNAL	**Value:** 'http://www.stud.uni-goettingen.de/internal/pics/rss_feed.gif' *(type=str)*
PIC_RUN_SCRIPTS	**Value:** '/internal/pics/run_scripts.jpg' *(type=str)*
PIC_SKIN	**Value:** '/internal/pics/lipstick.gif' *(type=str)*
PIC_SOURCE	**Value:** '/internal/pics/quelle.gif' *(type=str)*
PIC_TABLE_BG	**Value:** '/internal/pics/ohinten.gif' *(type=str)*
PIC_TABLE_ENTRY_BG	**Value:** '/internal/pics/2ohinten.gif' *(type=str)*
PIC_TABLEBG_O	**Value:** '/internal/pics/x1h.gif' *(type=str)*
PIC_TABLEBG_OL	**Value:** '/internal/pics/x1hol.gif' *(type=str)*
PIC_TABLEBG_OR	**Value:** '/internal/pics/x1hor.gif' *(type=str)*
PIC_TABLEBG_UL	**Value:** '/internal/pics/x1hul.gif' *(type=str)*
PIC_TABLEBG_UR	**Value:** '/internal/pics/x1hur.gif' *(type=str)*
PIC_UPARROW	**Value:** '/internal/pics/up.gif' *(type=str)*
PIC_USER_MANAGEMENT	**Value:** '/internal/pics/user_management.gif' *(type=str)*
PIC_XS	**Value:** '/internal/pics/xs.gif' *(type=str)*
PIC_XVUG	**Value:** '/internal/pics/xvug.gif' *(type=str)*
STYLESHEET	**Value:** '/internal/t.css' *(type=str)*
STYLESHEET2	**Value:** '/internal/tx.css' *(type=str)*
SYSTEM_TITLE	**Value:** 'FrAendZ' *(type=str)*
TEMPLATE_ERROR	**Value:** 'Error encountered during processing of a template!' *(type=str)*

7 Module fraendz.userio

userio.py - part of fraendz

* stores given data in appropriate config/user files

7.1 Functions

addNewEntry(*topic*, *author*, *entry*)

* adds a new entry to the 'topic'

addToLatestDiscussionEntries(*entry*)

* adds an entry in latest.dat under users/forum -> takes: entry=(author, date, topic, text) -> return: 0 success / 1 failure

appendNews(*user*, *text*)

* appends the news 'text' to the newsfile

archivateMessage(*user*, *msg*)

* moves msg from MSG_DIR to OLD_MSG_DIR

createNewTopic(*user*, *topic*, *description*)

```
* creates a new topic for the forum-discussion
* creates a file in FORUM_DIR, reflecting the time,
  containing title and author in the first two lines
```

getAllDiscussionTopics()

```
* returns a nested list with all discussion topics:
  [ ['date', 'topic', 'author', 'description'], ... ]
```

getAllMessages(*user*, *folder*='messages')

```
* returns a list of tuples of the form
        [(id, author, sent-time, message-text), ...]
  of all messages of user
```

getAllNews()

```
* gets all news from the news file and returns a dictionary of the form:
        {'date':['username', 'newstext']}
```

getCurrentChatContent(*user*)

getDiscussionEntries(*topic*)

```
* returns a list of all entries for the topic on hand, of the form
        [ ['date', 'author', 'entrytext'], ...]
* the list is sorted by date (latest addition last)
```

getLastMessages(*user*, *n*, *folder*='messages')

* return the id's of the last n messages

getLastNews(*num*)

```
* returns the last 'num' news from the NEWS_FILE in the same form
  as getAllNews() (see above)
```

getLatestDiscussionEntries()

* returns the entries in the latest.dat file under users/forum * this file contains the MAX_RSS_FORUM last entries made to the forum -> return: [(author, date, topic, text), ...]

getMessage(*user*, *msg*, *folder*='messages')

* returns tuple (author, sent-time, message-text) for msg

getMessageList(*user*, *folder*='messages')

* returns list of all files in users message-dir

getNumEntries(*topic*)

* returns the number of entries in 'topic'

getRealUsername(*user*)

* returns the complete Name for user if applicable

getSkin(*user*)

* return the entry in skin.dat in the user's directory

getTopicInfo(*topic*)

```
* takes the stripped filename to return information on the
  corresponding topic
```

getUserAdress(*user*)

* returns a dictionary of the information found in USER_ADRESSFILE

getUserHomepage(*user*)

* returns the content of the users homepage.dat file

getUserInfo(*user*)

```
* returns a dictionary of the information found in the SYSTEM_USERFILE
       {'FirstName':name, 'LastName':lastname, 'EMailAdress':email}
```

markAllMessagesSeen(*user*)

* changes the seen-attribute of all messages in msg-file to 1

markMessageSeen(*user*, *msg*)

* changes the seen-attribute in msg-file to 1

postChatEntry(*user*, *msg*)

sawMessage(*user*, *msg*)

```
* returns 1 if seen-attribute in msg-file in user's message dir is 1,
  0 if 0
```

sendMessage(*user*, *to_user*, *text*)

* writes the message supplied by user in the messages-folder of to_user * saves a copy of the message in user's messages/sent-folder

setSkin(*user*, *skin*)

```
* creates skin.dat in user directory containing the name
  of the skin
```

storeNewHomepage(*user*, *content*)

* writes 'content' into the users homepage.dat file

storeUserImage(*user*, *img*)

* img is a file-handle given by cgi-script * stores this image as personal_picture.extension in the user directory

- if the pic is < 100k
- if the pic is .png, .jpg or .gif

unseenMessages(*user*)

* returns 1 if there are any unseen messages for user, else 0

userHasPersonalPicture(*user*)

```
* determines if the user has a personal picture, and
  returns the name if found
```

writeSystemUserFile(*user*, *data*)

* writes data to the system file (SYSTEM_USERFILE, s.a.)

writeUserAdressFile(*user*, *data*)

* writes data to the users adress-file (USER_ADRESSFILE, s.a.)

7.2 Variables

Name	Description
ADRESS	**Value:** `('Phone', 'StreetName', 'HouseNumber', 'ZipCode-', 'City', 'Country', 'Homepag...` *(type=`tuple`)*
FORUM_DIR	**Value:** `'/Users/thias/web/internal/users/forum'` *(type=`str`)*
HOMEPAGE_FILE	**Value:** `'homepage.dat'` *(type=`str`)*
INFO	**Value:** `('FirstName', 'LastName', 'EMailAdress')` *(type=`tuple`)*
LATEST_DISCUSSION_FILE	**Value:** `'/Users/thias/web/internal/users/forum/latest.dat'` *(type=`str`)*
MSG_DIR	**Value:** `'messages'` *(type=`str`)*
NEWS_DIR	**Value:** `'/Users/thias/web/internal/users/news'` *(type=`str`)*
NEWS_FILE	**Value:** `'/Users/thias/web/internal/users/news/news.dat'` *(type=`str`)*
OLD_MSG_DIR	**Value:** `'messages/old'` *(type=`str`)*
SENT_MSG_DIR	**Value:** `'messages/sent'` *(type=`str`)*