Documentation for communication.py

Author: Lincoln Choy

This file contains all the functions and API's which help with node to node communication. Currently there is no encryption supported.

Functions:

def receive_message(data):

This function/API is called when other nodes want to send this client a message, and this function will store the message in a database and return a '0' if successful.

Inputs: data (dictionary, contains information about the sender,destination and the message)

Outputs: errorcode (string, currently this only returns '0','1',9', or '11', read the application protocol for the meanings of the error codes)

def send_message(message):

This function allows this client to send a message to another node, the destination of the message is stored in the cherrypy session object with the key 'destination'. The message is also stored in a database so that this sender may display it in the chat history.

Inputs: message (string)

Output: None

def save_message(message,sender,destination,stamp,is_file):

This helper function stores each message, whether it be sent or received, in a message table and a buffer table (in the same database file) which loads the message directly into the chat box via JavaScript

Inputs: message (string)

sender (string)

destination (string)

stamp (string)

is_file (string)

Output: None

def ping(sender):

This function/API is usually called before another client initiates communication with this node. It is useful to check the sender of the ping for rate limiting purposes.

Input: sender (string)

Output: error_code (string, '0' or '11', check application protocol for meanings of the error codes)

def send_file(file_data,mime_type):

This function is used to send a file via calling the other node's /receiveFile API, the file path of the file is stored in the database, while the file is stored somewhere else in the server directory.

Inputs: file_data (string, the file_data encoded in a base 64 string)

mime_type (string, mime_type of the file,used to guess the extension)

Output: None

def receive_file(data):

This function/API is called when another client wants to send this node a file. The file is stored somewhere on the server and the file path is stored a database. This file path is stored as a message so that the embedded viewer knows where to look when displaying this file in the chat history.

Input: data (dictionary, contains information about the sender and the file)

Output: error_code ('0' or '1')

def save_error_message(sender,destination,stamp):

This helper function saves a message in the message buffer so that JavaScript can notify the user that their message or file has not been sent successfully.

Inputs: sender (string)

destination (string)

stamp (string)

Output: None

def check_rate_limit(sender):

This function is used to check the requests of a certain user in the past minute.

Users are limited to 60 requests per minute to prevent traffic congestion of the server

Input: sender (string)

Output: error_code ('0' or '1', '0' indicates not rate limited, '1' indicates otherwise)

def get_chat_page(page,sender,destination):

This function returns a string of html code which displays the chat history on the browser

Inputs: page (string)

sender (string)

destination (string)

Output: page (string, html to be displayed in the browser, is concatenated with the input argument : 'page')

def add_embedded_viewer(file_source):

This helper function adds an embedded viewer to the html page for displaying images, videos, songs, and pdf's

Input: file_source (string, relative path of the file to be displayed)

Output: page (string, html code)

def refresh_chat():

This function is called by a JavaScript function, and is used to check for buffered messages yet to be displayed on the browser

Input: None

Output: out (json encoded dictionary, contains information about new messages received)

def notify():

This function adds notifications for new messages at the top of the page.

It does this by checking the buffered messages table.

Input: None

Output: out (json encoded dictionary, contains information about the sender)

def empty_buffer(destination):

This function empties the message buffer and is only called once JavaScript has confirmed that the messages have been displayed in the chat box

Input: destination (string)

Output: None