# Documentation of C#-webserver

This documentation contains a small explanation of all functions and classes in the c# webserver. Further information will be found as comments in the specified files.

## MessageHandler.cs

## Class MessageObject

Used as structure for all messages sent to the client. Contains MessageType and MessageContent

## Class MessageHandler

### MessageHandler(Server)

Assigning server variable

### void decryptMessage(MessageAsBytes Byte[])

decrypting the incoming message

after decrypting calling handleDecryptedMessage

### void handleDecryptedMessage(Message (string))

handles the decrypted messages and executes appropriate functions depending on messagetype

### void constructResponseMessage(messagetype (int),optional bool succeeded, optional string datamessage)

Constructs the final response message based on messagetype and two optional parameters

### void sendResponseMessage(message(string))

Calls the servers writeToSocket function

### int getMessageType(Message (string))

returns the type of the specified message

### int getMessageContentAsInt(Message (string)

returns the message content of the specified message if it actually is an int

## EYE.cs

Handles all communication with the eye-tracker

## Class data

Small structure class.

Contains TESTID, USER and x and y coordinates.

Used as structure when saving eyetracking data to json file

## Class EYE

EYE(displayHeight,displayWidth(int,int))

Initializes variables and starts the EyeXHost

### bool isRecording()

returns the status of the m\_isRecording variable.

### string getDataString()

returns the string of data collected from the eyetracker

### setScrollPosition(InArg newposition (int))

updates variable m\_scrollPosition to a new value

### saveTemporaryCoordinates(InArg x,y (ints))

called by the eye tracker event everytime a new gaze point is found. Saving the x and y coordinate of the eye to a temporary list structure.

Also offsetting the eye y-value with the value of the scrollPosition

Saving data only when the recording isn’t paused

Ignores out of bounds coordinates

### bool startRecording(string testuser)

creates a new data stream from the EyeXHost and tells it to send all gaze point events to the saveTemporaryCoordinates function.

Updates the recording status

Only starts recording if there is no active recording.

### bool pauseRecording()

pausing the recording if there is an active recording

### bool resumeRecording()

resume a paused recording if there is an existing recording and it is paused

### bool stopRecording()

stopping an active recording if there is any and then calls the prepareSaveCoordinatesToFile function

### void prepareSaveCoordinatesToFile()

Prepares everything that needs to be done before saving the data to a file.

Checks if the file and the folder for data exists and when they do the saveDataToFile function is called

### void createFolder(folderpath (string))

Creating folder at the specified path

### void createJSONFile(filepath (string))

Creating json file at the specified path

### void saveDataToFile(filepath (string))

inserts the new data into the json file at the specified path

## Server.cs

## Class Server

### public Server(IPADDRESS(string),Portnumber(int),Form1 formreference)

Initializes variables, start a new TcpListener using the specified address and port

### public bool requestStartRecording()

Requests the EYE recorder class to start recording. Returns true if the start succeeded and false if it failed.

### public bool requestPauseRecording()

Requests the EYE recorder class to pause recording. Returns true if the start succeeded and false if it failed.

### public bool requestResumeRecording()

Requests the EYE recorder class to resume recording. Returns true if the start succeeded and false if it failed.

### public bool requestStopRecording()

Requests the EYE recorder class to stop recording. Returns true if the start succeeded and false if it failed.

### public string requestDataString()

Requesting the private data string with information from the last test from the EYE recorder class

### public void requestScrollUpdate(newScrollHeight(int))

Requesting the EYE recorder class to update its scrollheight variable with a new value

### public void listenForConnection()

A function used by a thread. Waits for incoming connection attempt.

If there was a successful connection this function will perform an handshake with the connected client. If the handshake was successful a new thread is started with the clientListen function.

When the new thread is successfully created, this thread terminates.

### void clientListen()

Listening for incoming packages from the client stream. Collects the message from the client in bytes and forwards the message to the MessageHandler.

Since this listening thread contains while loops there is also some variables which controls these loops added. So if the thread is supposed to terminate the loop will stop and the thread will terminate safely when all work is done.

### public void updateOutputLog(text(string))

Notifies the form reference to update the output log

### public void updateDisplayHeightWidthToForm(displayheight,displaywidth(int,int))

Notifies the form reference to update the display height and width boxes

### public void updateClientStatus(type (int))

Notifies the form reference to update the client status label with a new value depending on incoming int

0 = Connected 1 = Not connected

### public void updateEyeTrackingStatus(type(int))

Notifies the form reference to update the visual text of the eye recorder status depending on incoming int.

0 = Online, 1 = Offline and 2 = Paused

### public bool handleClientDisconnectRequest()

Handles a client disconnect request. Returns true if the disconnect succeeded and false if not.

Stops recording if it is active.

Updates a terminate variable to true so that the client receive thread will exit

Finally restarting the listening thread again and waiting for a new connection

### public void writeToSocket(message(string))

Writes a response message to the client stream.

Masking the header of the message according to the size of the message.

Sends message over one frame.

## Form1.cs

## Class Form1

### Form1()

Initializes form components

Starting a new server instance

### void Form1\_Closing(FormClosingEvent)

Closing the form when it is safe to close

### void updateEyeTrackStatus(status(string))

Updating the form eye tracker status box with the current status of the Eye tracker

### void updateClientStatusLabel(status(string))

Updating the form client current status box with the chosen status

### void updateOutputLogBox(logMessage(string))

prints a requested logmessage to the form listbox

### void updateDisplayHeight(newDisplayHeight (int))

updating the display height scroll box with the new display height.

### void updateDisplayWidth(newDisplayWidth (int))

Updating the display width scroll box with new display width