Detailed Design Document

Web Wireshark

Introduction

The Web Wireshark should be modeled as a Client Server system. The client which runs on the browser of a client, communicates with the server, which runs in a webserver. The client and the server are both encapsulated of each other. The client can be identifyed in the packages which contains „client“ and the server packages contains „server“. Classes and Interfaces which are needed from both, client and server, can be found in the packages which contains „shared“.
The Web Wireshark should represent a rudimentary version of the Wireshark. The project should be modelled with the GWT(Google Web Toolkit).

Tasks

1. **Sessions:** The demo implementation uses the app-engine for session handling. In your version you need to do this by yourself. Implement a complete session handling/management solution:
    - **Setting cookies**: when new clients arrive, using the cookies within the subsequent communication. ADDITIONAL NOTE (24.05.2012) (also within the Newsgroup): Do NOT use methods like addCookie etc. Instead, when setting/reading a cookie, set/read the appropriate headers within the HTTP response/request.
    - **Implement session timeouts**: If there is no new request from the client within 10 seconds, the session is not valid anymore. Inform the user that the session has timed-out (the short timeout makes it easier to test it).
    - **Cookie generation**: Find a good method to create random numbers that are not predictable and provide enough entropy...
    - **Security Measures**:
    - **Login/Logout Button on the client**: Use a simple login button for Webwireshark that simulates the login process. After pressing the button, the server sets the cookie and maintains the session (including timeouts etc.). The fileupload functionality and the other buttons are only active after pressing the login button. Create a logout button that logs the user out of the application (session is terminated).
    - **Research**: Do some reasearch and find guidelines for implementing session management on the server and how to use cookies in a secure way (start with https://wiki.mozilla.org/WebAppSec/Secure_Coding_Guidelines#Session_Management). Implement the measures that apply to our simple web app.
    - **BONUS**: The standard JETTY does not use HTTPS, thus stealing the cookie would be quite easy. Find a way to use HTTPS: either reconfigure Jetty or use an Apache web server that acts as proxy for Jetty.

2. **Raw HTTP Requests:** The demo implementation uses GWT RPC to communicate between client and server. In order to learn details of the HTTP protocol, you need to replace the GWT RPC communication with raw HTTP requests and Servlets (as shown in the examples).

3. **Transmitted data**: In the current implementation, GWT RPC is used. Thus, the Java objects can be directly transferred between the client and the server (and vice versa). However, since you need to replace GWT RPC with raw HTTP requests, this is not possible anymore. Use JSON to serialize the Java objects, and convert them back to Java objects on the receiving side (this was shown in the JSON example above).

4. **Websockets :** The demo implementation uses AJAX Polls for the communication between client and server. You need to replace this communication with websockets (whenever the server has a new packet it should send it to the client). In order to analyze the amount of generated traffic (see below), both implementations must be usable (e.g. two buttons that allow the user to use either websockets or polling).

5. **BONUS**: find a way to use secure WebSockets.

6. **PCAP Analysis**: In the demo application, the PCAP files are already decoded, and various protocols decoders are already setup (others, like HTTP are missing). However, the decoded packets do not contain any information yet (except for the size of the PCAP files and a simple timestamp). You need to capture the different protocol headers and **visualize all fields on the client (please read the newsgroup entry: Kraken, Wireshark and Demo implementation)**. The following protocols need to be considered:
   - ETHERNETFRAME
   - IP
   - UDP
   - TCP
   - ~~DNS~~ DHCP (This was changed to DHCP on 14.05.2012, reason there is no decoder available for DNS. You can download the DHCP decoder from http://download.krakenapps.org/org/krakenapps/kraken-dhcp-decoder/1.0.1/. It is not included in the downloaded project files. Its functionality is similar to the HTTPProcessor/Decoder that is already included in the demo project. More information about this ~~will be made~~ is available in the news group (14.05. RKN KU - Task 2 - Additional Information))
   - HTTP
     - Extract HTTP cookies!
     - Extract the request type
     - Extract the host and the URL
     - Extract the content-length
     - Show the headers and their values
   - ICMP

◦ It’s up to you how you represent this on the client: You can add additional GUI elements, or use the existing ones.

7. **Websockets vs. normal AJAX polls:** Add some functionality to your application that allows you the show the “overhead-difference” of websockets and AJAX polling (e.g. traffic counters, ratio between protocol information and real data etc.). Use this functionality to conduct an analysis on the differences and present your results in the design document (below...). Similar to the notes regarding the design-document, assume the following situation here: You - as a company - have a client (me) who wants to have an analysis regarding the amount of traffic caused by both methods (websockets, AJAX). (We ignore the fact that this amount could be reduced by long-polling which is not used in the code provided by us). You have to decide which information you need to provide in your analysis in order to enable the client to understand the differences.

8. **Design document:** Create a design document in which you describe the specifics of your implementation for each of the tasks above**.** Include that document in the root directory of your project and call it design.pdf. Regarding the level-of-detail in the document: Assume you have your own company and are doing this project for one of your clients, who wants to use the WebwireShark code. The client requires your design document in order to use the code for its own projects. Thus, the design document should include enough information in order to understand your code, your implementation and the specific details of your implementation.

General BONUS points: You can earn bonus points by providing a more-than-average user interface or by incorporating other interesting features that are not already part of the task description.

Description of the Design

The connections between the different classes can be found in the class diagram at the end of the design document.

Implementation Details:

Sessions
Cookies are generated on the server side and are set in the http responses. The client browser handles these cookies. If the client does not react after 10 seconds the session times out and the cookies must be renewed.

Websockets
Websockets are implemented on Client side in the package client.websocket. When the Pcap analysis should be started with Websockets at first the browser is tested if it supports Websockets.
When an Pcap analysis should be startet the client sends the session Id to the server. Then the server starts the Pcap analysis and returns packets to the client who displays the received pakets. If the final packet is reached the server closes the Websocket Connection. The connection can also be closed when the client hits the Stop Analysis button.

Client

The client is contains several packages which are:

- at.iaik.websharkgwt.client
- at.iaik.websharkgwt.client.gui
- at.iaik.websharkgwt.client.gson
- at.iaik.websharkgwt.client.service
- at.iaik.websharkgwt.client.websocket

Package: client

The package client contains the class WebSharkGWT.java.

class WebSharkGWT: This class is the Entry Point of the GWT project. The function begins always with EntryPoint() and jump to the function OnModuleLoad().

attributes:

◦ String SERVER_ERROR: This string contains the error code which is set when an error occur.

methods:

- **void onModuleLoad():** This method is loaded if an GWT project is started. Here a new instance of the LoginPage is created and is added to the Rootpanel.

Package: client gui

The package client.gui contains the classes LoginPage.java and PacketDisplayTable.java and the responding xml files: LoginPage.ui.xml and PacketDisplayTable.ui.xml.

class LoginPage: This class is the conatins the Login/Logout function on the client side.s

attributes:

- **Button loginButton:** This Button represents the login Button.

methods:

- **void login():** This function creates a new loginService and send a loginrequest to the server, which builds a requestbuilder and send the request of this builder. Then set the text of the loginButton to „logout“ and creates a new instance of the PacketDisplayTable and added this to the Rootpanel.
- **void logout():** This function sets at first the loginbutton text to „login“, clears the rootpanel and adds only the loginbutton to the rootpanel.

class PacketDisplayTable:
attributes:
- PacketDisplayTableUiBinder uiBinder
- ListDataProvider<IMyPcapPacket> smallQueueSizePacketProvider
- List<IMyPcapPacket> localPacketListStore
+ SingleSelectionModel<IMyPcapPacket> smallTablesingleSelectionModel
- CellTable<IMyPcapPacket> smallQueuePacketCellTable
- FileUpload fileUpload
- FormPanel formPanel
- Button submitButton
- TextArea packetDetailsSmallQueueTextArea
- Button startWebsocketDataButton
- Button startPollingDataButton
- SimplePager smallQueueSimplePager
- Button stopPollingDataButton
- IntegerBox delayInMsIntegerBox2

methods:
+ void createFileUploadGUI(): This is a file upload handler, which can select a file and upload this with the function formpanel.submit() to the server. Here is the upload options very important, like:
formPanel.setEncoding(FormPanel.*ENCODING_MULTIPART*);
formPanel.setMethod(FormPanel.*METHOD_POST*);
formPanel.setAction(ServletLocations.*FILE_UPLOAD_SERVLET*);

+ void createSmallQueueGUI(): This function adds to the smallQueuePacketCellTable the time, source, number, destination,.. and other details of all pcap files. The handler of the buttons StartPollingDataButton, StartWebsocketDataButton and StopPollingDataButton are also implemented in this function.

+ void onStartPollingDataButtonClick(): This handler creates a new StartAnalysisService and call the function startAnalyse() of the StartAnaylsisService, which builds a requestbuilder and send a request to the server. Here the 10 seconds delay- time between packets and the queuesize of the packets are also implemented.

+ void onStartWebsocketDataButtonClick(): This handler of the StartWebsocketDataButton creates at first a Websocket and call the function startConnection of the Websocket.java, which builds a websocket connection between server and client.

+ void onStopPollingDataButtonClick (): The handler of the StopPollingDataButton creates at first a new StopAnalysisService and call the function stopAnalyse() of the StopAnalysisService, which stop the packet-analyse. At last the handler stops the websocket connection, if the analyse is finished.

Package: client gson

The package client.gson contains the interface MyAutobeanFactory.java the class MyBeanFactory.java.

interface MyAutobeanFactory: This interface
methods:
+ AutoBean<IMyEthernetFrame> getEthernetFrame():
+ AutoBean<IMyIPv4Packet> getIpv4Packet():
+ AutoBean<IMyUdpPacket> getUdpPacket():
+ AutoBean<IMyTcpPacket> getTcpPacket():
+ AutoBean<IMyIcmpPacket> getIcmpPacket():
+ AutoBean<IMyHttpPacket> getHttpPacket():
+ AutoBean<IMyDhcpPacket> getDhcpPacket():
+ AutoBean<IPacketCommand> getCommand():

class MyBeanFactory: This class
attributes:
- MyBeanFactory myPacketFactory:
- MyAutobeanFactory autoBeanFactory:
methods:
+ IMyPcapPacket getPacket(String currentClass, String jsonString): This method returns a Paket(Java Object) for a given json String.
+ IPacketCommand getPacketCommand():

Package: client services

The package client.services contains the classes FileUploadService.java, LoginService.java, StartAnalysisService.java and StopAnalysis.java.

class FileUploadService: This class
attributes:
+ String SERVER_ERROR
+ String baseUrl
+ String serviceUrl

methods:
+ void sendLoginRequest(): The function call only the function initConnection().

+void InitConnection(): This function builds at first a Requestbuilder and sends a request. If the server recieves this request succesfully, so it sets the statuscode to 200. it's meaning you can just controls in InitConnection(), if response.statuscode is 200 to check if the connection was successfully.

class LoginService: This class
attributes:
+ String SERVER_ERROR
+ String baseUrl

+ String serviceUrl
+ String sessionID

methods:
+ void sendLoginRequest(): The function call only the function initConnection().

+ void initConnection(): This function builds at first a Requestbuilder and sends a request. If the server recieves this request succesfully, so it sets the statuscode to 200. it's meaning you can just controls in InitConnection(), if response.statuscode is 200 to check if the connection was successfully.

class StartAnalysisService:
attributes:
+ String SERVER_ERROR
+ String baseUrl
+ String serviceUrl
- Vector<IMyPcapPacket> packetLists

methods:
+ void startAnalyse(int delayInMsBetweenPackets, int queueSize, final ListDataProvider<IMyPcapPacket> smallQueueSizePacketProvider): This function builds at first a Requestbuilder, set header of this builder and sends a request.
To set the header of this builder:

```
builder.setHeader("Analyse", "start");
builder.setHeader("delayTime",
String.valueOf(delayInMsBetweenPackets));
builder.setHeader("queueSize", String.valueOf(queueSize));
```

If the server receives this request successfully, so it sets the statuscode to 200. it's meaning you can just controls the statuscode in InitConnection(). If response.statuscode is 200, so the connection was successfully. The other possible statuscodes are: “204” for not started analyze and “302” for not finished analyze.

class StopAnalysisService:
attributes:
+ String SERVER_ERROR
+ String baseUrl
+ String serviceUrl

methods:
+ void stopAnalyse(): This function builds at first a Requestbuilder and sends a request. If the server recieves this request succesfully, so it sets the statuscode to 200. it's meaning you can just controls in InitConnection() the response.statuscode. If the statuscode is 200, so the connection was successfully.

Package: client websocket

The package client.websockets contains the class Websocket.java.

class WebSocket: This class represents the client side of the WebSocket connection.
attributes:
- Connection webSocketClient: This Connection represents the connection tot he server.
- Vector<IMyPcapPacket> packetList: This vector contains the packets which are received from the server.

methods:
+ void testIfWebsocketsAvaliable(): This function tests if the used browser can use WebSockets. It will throw an error if the client's browser is not able.

+ void startConnection(): In this function the clients sends the session ID to the server to initiate a connection. This session ID is needed to return the right packets from the packetStore of the uploaded pcap-File. The vector packetList is filled up with the packages, which are received, from the server.

Server

The server contains several packages which are:

- at.iaik.websharkgwt.server
- at.iaik.websharkgwt.server.kraken
- at.iaik.websharkgwt.server.kraken.packetconstructor
- at.iaik.websharkgwt.server.servlets
- at.iaik.websharkgwt.server.utils
- at.iaik.websharkgwt.server.webserverstorage
- at.iaik.websharkgwt.server.websocket

Package: server

It contains HTTPDemoServlet.java and StaticStore.jave classes.

class HTTPDemoServlet:
attributes:
~ List<String> nonStaticStore

methods:
~ void doGet()*: This function controls at first, if a greeting from the client is available. If a greeting message is available, so the greeting message will be stored in the storedGreeting list, the server send also a greeting message to the client and set the statusnumber of the connection. The status will be set to 200, if the connection was successfully and to 501 if the connection was not OK.*

class StaticStore: This is a singleton class.
attributes:
- List<String> greetings

methods:
+ *void* getInstance(): This function checks if this is available. If the class will be created for the first time, so a new object of StaticStore will be created, otherwise the available object will be used. Here the object StaticStore can be created only one time, because of singleton class.

+ *void* addGreeting():The function add the supplied greeting to the gretings list.

+ *void* getGreetings(): This function returns the greetings list.

Package: server.kraken

It contains many classes, which are generally the same. They are decoder and processer classes for all available packet protocols.

class MyDHCPDecoder:
attributes:
~ string sessionID
~ boolean addToResults

methods:
+ void process(): This function call the base class.

class MyDHCPProcessor:
attributes:
~ string sessionID
~ boolean addToResults

methods:
+ void process(): This function adds the new created packet into the packetStore.

class MyEthernetDecoder:
attributes:
~ string sessionID
~ boolean addToResults

methods:
+ void decode(): This function calls the base class and adds the packet into the packetStore.

class MyHTTPDecoder:
attributes:
~ string sessionID
~ boolean addToResults

class MyHTTPProcessor:
attributes:
~ string sessionID
~ boolean addToResults

methods:
+ void onMultipartData(): This function printf only the text: "HTTP on MultipartData".

+ void onRequest(): This function adds the packet into the packetStore.

+ void onResponse(): This function adds the packet into the packetStore.

class MyICMPDecoder:
attributes:
~ string sessionID
~ boolean addToResults

methods:
+ void process(): This function calls only the base class.

class MyICMPProcessor:
attributes:
~ string sessionID
~ boolean addToResults

methods:
+ void process(): This function adds the packet into the packetStore.

class MyIPv4Decoder:
attributes:
~ string sessionID
~ boolean addToResults

methods:
+ void process(): This function adds the packet into the packetstore.

class MyTCPDecoder:
attributes:
~ string sessionID
~ boolean addToResults

methods:
+ void process(Ipv6Packet ipv6Packet): This function calls the base class.

+ void process(Ipv4Packet ipv4Packet): This function calls the base class.

+ void dispatchNewTcpSegment():This function adds the packet into the packetstore and calls the base class of the class dispatchNewTcpSegment.

class MyTCPProcessor:
attributes:
~ string sessionID
~ boolean addToResults

methods:
+ void handleRx(): This function calls the base class of the handleRx class.

+ *void handleTx():* This function calls the base class of the handleTx class.

+ *void onEstablish():* This function calls the base class of the onEstablish class.

+ *void onFinish():* This function calls the base class of the onFinish class.

+ *void onReset():* This function calls the base class of the onReset class.

class MyTCPProcessor:
attributes:
~ string sessionID
~ boolean addToResults

methods:
+ *void onReceive():* This function adds the tcp packet into the packet store.

class MyUDPDecoder:
attributes:
~ string sessionID
~ boolean addToResults

methods:
+ *void process(Ipv4Packet packet):* This function calls the base class of the process class.

+ *void process(Ipv6Packet p):* This function calls the base class of the process class.

class MyUDPProcessor:
attributes:
~ string sessionID
~ boolean addToResults

methods:
+ *void process(Ipv4Packet packet):* This function adds the udp packet into the packetStore.

Package: server.kraken.packetconstructor
Here is only the class MyPacketConstructor.java available.

class MyPacketConstructor:
attributes:
+ int counter

methods: There are many functions with the same name, but they have different supplied packet protocols(DHCPPacket, HTTPPacket, ICMPPacket, ...).
+ void MyPcapPacket conctruct(packet): This function checks at first, if the packet is available. If the packet is not available, so a packet of the same protocol will be created. Then the function sets all variables of this packet in new packet, it's meaning the packet will be formatted into new format and at last the package will be returned.

Package: server.servlets
This contains the classes:
- FileUploadServlet
- LoginServlet
- PCAPAnalysisThread
- PCAPFileUploadServlet
- PollingPacketServlet
- SessionManager
- StartAnalysisServlet
- StopAnalysisServlet

class FileUploadServlet:
attributes:
~ List<String> nonStaticStore

methods:
+ void doGet: This function creates a new cookie and adds it into the cookieStore, if no cookie is available. If a cookie is available, so the sessionID of the cookie will be stored in sessionID variable. If a greeting from client is available, so the status will be set to 200 and the status will be set to 501, if the connection was not successfully.

+ void doPost: This function read at first the HttpServletRequest and saves it into a ByteArrayOutputStream if no error was catched. Then the new formatted file will be added into the pcapFileStorage and the status will be set to 200.

class LoginServlet:
attributes:
~ List<String> nonStaticStore

methods:

+ void doGet: Here the function creates at first a new cookie, which is implemented in cookieGenerator.createNewCookie() class and saves the new cookie into the cookieStore. Than sets the header options (setCookie, setSessionID, setGreeting) and sets the status to 200.

class PCAPAnalysisThread:
attributes:
~ List<String> nonStaticStore
~int msDelayBetweenPacket

methods:
+ void run: This function creates at first a packetStore and checks with the function clearPacketListOfSessionID(), if the sessionID is unique. If the sessionID is available, so this packet will deleted. Than the input stream in the PCAPFileStorage of this sessionID will be saved in a new variable and the function controls also, if this variable is empty. If the inputstream is empty, so this packet will be deleted from taskController and the programm jump out of this class. The taskController controls all thread of this programm, it is also a scheduler of this project. The function creates at next many decoders for several packet protocols. This decoders formats the inputstreams in a new format, like MyEthernetDecoder, MyIPv4Decoder, …etc.
This operation will be repeated in a endless while-loop. If the final packet is achieved, so the final packet will also processed and the program jumps out of this endless while loop. If the delaytimeBetweenPacket is bigger than 10 sec, so the thread should be sleep and If a thread will be executed, so the thread will be removed from the task controller.

class PCAPFileUploadServlet:
attributes:
~ List<String> nonStaticStore
~int msDelayBetweenPacket

methods:
+ void doPost(): This function prints only the SessionID.

class PollingPacketServlet:
attributes:
~ Gson gson

methods:
+ void init(): This function create at first a GsonBuilder and initialize it with the function super.init().

+ void doGet(): This function controls at first, if there are more than 5 packets in packetStore, than the packets will be sends to the client. If the final packet will be send, so the status will be set to 200 and if the packet are available, so the pcap files will be converted in gson format. Than the contentType and header will be set and the status will be set to 200.

class SessionManager:
attributes:

methods:
+ void sessionControl(): This function get all cookies and saves the sessionId of the cookie in sessionId variable.

+ boolean isSessionIdValid: This function check at first, if the sessionID is available in cookie store. If the SessionID is available, so the function checks, if the timestamp of the session smaller than 10 seconds. If the timestamp is smaller than 10 seconds, so true will be returns and otherwise false will be returns.

class StartAnalysisServlet:
attributes:
~ List<String> nonstaticStore
int msDelayBetweenpacket
int queuesize
~ Gson gson

methods:
+ void doGet(): This function checks if the running task is in taskstore available. If the running task isn't available in this store, so a new PCAPAnalysisThread will be creates, the function saves this thread into taskstore and sets the status to 200.

class StopAnalysisServlet:
attributes:
~ List<String> nonstaticStore
int msDelayBetweenpacket
int queuesize
~ Gson gson

methods:
+ void doGet(): This function stops the task and sets the status to 200.

Package: server.utils

The package server.utils contains the classes :
- CookieGenerator.java
- CookieStore.java
- MyCookie.java

- Utils.java

class CookieGenerator: This class represents
methods:
+ MyCookie createNewCookie(): This method creates a unique sessionID and returns it.

class CookieStore: This class represents
attributes:
- CookieStore cookieStore:
- Vector<MyCookie> cookies:

methods:
+ CookieStore getInstance(): This class is a singleton class, here the function creates only one cookieStore Object and returns it.

+ void addCookie(MyCookie cookie): This Method adds the supplied cookie to the cookie store.

+ List<MyCookie> getCookie(): This function return the list of all available cookies.

+ MyCookie isCookieInStore(String session_id): This function returns a cookie, if this is available in cookieStore, otherwise it will returns null.

+ void deleteCookie(String session_id): This function deletes supplied cookie from the cookieStore.

class MyCookie: This class represents
attributes:
- String session_id_:
- String comment_:
- String max_age_:
- String time_stamp_:

methods:
+ String getComment_(): It returns the comment.

+ String getMaxAge(): It returns the max age.

+ String getSessionId(): It returns sessionID.

+ String getCookieString(): It returns a new generated string.

+ String getTimeStamp(): It returns the time stamp.

class Utils: This class represents
methods:
+ void copyStream(InputStream inputStream, OutputStream outputStream):
This function converts the input in a new output format.

Package: server.webserverstorage

The package server.webserverstorage contains the classes:
- PacketStore.java
- PacketStoreSingelton.java
- PCAPFileStoreSingelton.java
- TaskControllerSingelton.java.

class PacketStore: This class represents
attributes:
- HashMap<String,Vector<MyPcapPacket>> sessionIDToPacketListMap
- int queueSize

methods:
+ int getQueueSize(): It returns the queuesize.

+ void setQueueSize(int queueSize): It set the queuesize.

+ Vector<MyPcapPacket> getPacketList(String sessionID): It returns the packetlist.

+ MyPcapPacket getNextPacket(String sessionID): It returns the next packet.

+ void addPacket(String sessionID,MyPcapPacket packet): It adds a new packet.

+ void clearPacketListOfSessionID(String sessionID): It controls if sessionID is exists.

+ void clearAllPacketLists(): It deletes the Packetlist, which contains all packets.

class PacketStoreSingelton: This class represents
attributes:
- PacketStore packetStore;

methods:
+ PacketStore getInstance(): This class is a singleton class, it checks if this class is available. If the class isn't available, so the function creates a packetStore class.

class PCAPFileStorageSingelton: This class represents
attributes:
- PCAPFilesStorageSingelton pcapFileStore:
- HashMap<String, byte[]> sessionIdToFileDataMapper:

methods:
+ PCAPFileStoreSingelton getInstance(): This class is a singleton class, it checks if this class is available. If the class isn't available, so the function creates a PCAPFileStore class.

+ void addFile(String sessionID,byte[] fileData): It adds a filedata and sessionID to sessionIDToFileDataMapper.

+ InputStream getInputstream(String sessionID): It returns the input stream.

class TaskControllerSingelton: This class represents
attributes:
- TaskControllerSingelton taskStore:
- HashSet<String> runningTasks:
- HashSet<String> tasksCalledToStop:

methods:
+ TaskControllerSingelton getInstance(): This class is a singleton class, it checks if this class is available. If the class isn't available, so the function creates a TaskControllerSingletons class.

+ boolean isTaskRunning(String sessionID): It return true, if the task of the supplied sessionID running and returns false if the task doesn't running.

+ boolean isTaskBeingStopped(String sessionID): This checks, if task being stopped.

+ boolean runTask(String sessionID,Thread threadToRun): It returns false, if the task of the supplied sessionID running and return true, if the task doesn't exist. So the task will be creates before to return true.

+ boolean stopTask(String sessionID): It return true and stops the task, if the task is running and returns false, if the task does't exists.

+ void removeTask(String sessionID): It removes a task from the tastStore and stops it.

Package: server.websockets

The package server.websockets contains the class WebSocketImplServlet.java.

class WebSocketImplServlet: This class represents the server side of the WebSocket connection.
attributes:
- WebSocket webSocket;
- WebSocket.Connection webSocketConnection;
- Gson gson;
- int queueSize = 10;
- int msDelayBetweenPacket = 100;

methods:
+ void init(): This function creates a gsonBuilder and initializes it with the function super.init().

+ WebSocket doWebSocketConnect(HttpServletRequest arg0, String arg1):

Shared

The server contains several packages which are:

- at.iaik.websharkgwt.shared
- at.iaik.websharkgwt.shared.packet.implementation
- at.iaik.websharkgwt.shared.packet.interfaces
- at.iaik.websharkgwt.shared.utils

- ***Package: shared.packet.implementat ion***
-
- The package shared.packet.implementations contains the classes MyDhcpPacke.java, MyEthernetFrame.java, MyFinalPacket.java, MyHTTPPacket.java, MyICMPPacket.java, MyIPv4Packet.java, MyPcapPacket.java, MyTCPPacket.java and MyUDPPacket.java.
-
- **class IMyDHCPPacke:** This class represents a Dhcp Packet.
- **attributes:**
- **-** long messageType;
- **-** String hardwareType;
- **-** long hardwareAddressLength;
- **-** long hops;
- **-** String transactionID;
- **-** long secondsElapsed;
- **-** String bootpFlags;
- **-** String clientIPAddress;
- **-** String yourClientIPAddress;
- **-** String nextServerIPAddress;
- **-** String relayAgentIPAddress;
- **-** String clientMacAddress;
- **-** long clientHardwareAddressPadding;
- **-** String serverHostname;
- **-** String bootFilename;
- **-** String magicCookie;
- **-** Vector<String> options;
- **methods:**
- **+** long getMessageType();
- **+** void setMessageType(long messageType);
- **+** String getHardwareType();
- **+** void setHardwareType(String hardwareType);
- **+** long getHardwareAddressLength();
- **+** void setHardwareAddressLength(long hardwareAddressLength);
- **+** long getHops();
- **+** void setHops(long hops);
- **+** String getTransactionID();
- **+** void setTransactionID(String transactionID);
- **+** long getSecondsElapsed();
- **+** void setSecondsElapsed(long secondsElapsed);
- **+** String getBootpFlags();
- **+** void setBootpFlags(String bootpFlags);

- **+** String getClientIPAddress();
- **+** void setClientIPAddress(String clientIPAddress);
- **+** String getYourClientIPAddress();
- **+** void setYourClientIPAddress(String yourClientIPAddress);
- **+** String getNextServerIPAddress();
- **+** void setNextServerIPAddress(String nextServerIPAddress);
- **+** String getRelayAgentIPAddress();
- **+** void setRelayAgentIPAddress(String relayAgentIPAddress);
- **+** String getClientMacAddress();
- **+** void setClientMacAddress(String clientMacAddress);
- **+** long getClientHardwareAddressPadding();
- **+** void setClientHardwareAddressPadding(long clientHardwareAddressPadding);
- **+** String getServerHostname();
- **+** void setServerHostname(String serverHostname);
- **+** String getBootFilename();
- **+** void setBootFilename(String bootFilename);
- **+** String getMagicCookie();
- **+** void setMagicCookie(String magicCookie);
- **+** Vector<String> getOptions();
- **+** void setOptions(Vector<String> options);
-
- **class MyEthernetFrame:** This class represents a EthernetFrame Packet.
- **attributes:**
- **-** String arrivalTime;
- **-** double epochTime;
- **-** long frameNumber;
- **-** long frameLength;
- **-** long captureLength;
- **-** String sourceAddress;
- **-** String destinationAddress;
- **-** long type;
- **methods:**
- **+** long getType();
- **+** void setType(long type);
- **+** String getArrivalTime();
- **+** void setArrivalTime(String arrivalTime);
- **+** double getEpochTime();
- **+** void setEpochTime(double epochTime);
- **+** long getFrameNumber();
- **+** void setFrameNumber(long frameNumber);
- **+** long getFrameLength();
- **+** void setFrameLength(long frameLenght);
- **+** long getCaptureLength();
- **+** void setCaptureLength(long captureLength);
- **+** String getDestinationAddress();
- **+** void setDestinationAddress(String destinationAddress);
- **+** String getSourceAddress();
- **+** void setSourceAddress(String sourceAddress);
-
- **class MyFinalePacket:** This class represents the last Packet.

-
- **class MyHttpPacket:** This class represents a Http Packet.
- **attributes:**
- **-** String requestVersion;
- **-** long statusCode;
- **-** String responsePhrase;
- **-** String date;
- **-** String server;
- **-** String location;
- **-** String vary;
- **-** String contentEncoding;
- **-** long contentLength;
- **-** long keepAlive;
- **-** String connection;
- **-** String contentType;
- **-** String contentEncodedEntityBody;
- **methods:**
- **+** String getRequestVersion();
- **+** void setRequestVersion(String requestVersion);
- **+** long getStatusCode();
- **+** void setStatusCode(long statusCode);
- **+** String getResponsePhrase();
- **+** void setResponsePhrase(String responsePhrase);
- **+** String getDate();
- **+** void setDate(String date);
- **+** String getServer();
- **+** void setServer(String server);
- **+** String getLocation();
- **+** void setLocation(String location);
- **+** String getVary();
- **+** void setVary(String vary);
- **+** String getContentEncoding();
- **+** void setContentEncoding(String contentEncoding);
- **+** long getContentLength();
- **+** void setContentLength(long contentLength);
- **+** long getKeepAlive();
- **+** void setKeepAlive(long keepAlive);
- **+** String getConnection();
- **+** void setConnection(String connection);
- **+** String getContentType();
- **+** void setContentType(String contentType);
- **+** String getContentEncodedEntityBody();
- **+** void setContentEncodedEntityBody(String contentEncodedEntityBody);
-
-
- **class MyIcmpPacket:** This class represents a Icmp Packet.
- **attributes:**
- **-** long type;
- **-** long code;
- **-** String checksum;
- **-** long identifierBE;

- **-** long identifierLE;
- **-** long sequenceNumberBE;
- **-** long sequenceNumberLE;
- **-** String data;
- **methods:**
- **+** long getType();
- **+** void setType(long type);
- **+** long getCode();
- **+** void setCode(long code);
- **+** String getChecksum();
- **+** void setChecksum(String checksum);
- **+** long getIdentifierBE();
- **+** void setIdentifierBE(long identifierBE);
- **+** long getIdentifierLE();
- **+** void setIdentifierLE(long identifierLE);
- **+** long getSequenceNumberBE();
- **+** void setSequenceNumberBE(long sequenceNumberBE);
- **+** long getSequenceNumberLE();
- **+** void setSequenceNumberLE(long sequenceNumberLE);
- **+** String getData();
- **+** void setData(String data);
-
- **class MyIPv4Packet:** This class represents a Ipv4 Packet.
- **attributes:**
- **-** long version;
- **-** long headerLength;
- **-** String differentiatedServiceField;
- **-** long totalLength;
- **-** String identification;
- **-** String flags;
- **-** long fragmentOffset;
- **-** long timeToLive;
- **-** String protocolIP;
- **-** String headerChecksum;
- **-** String sourceIPAddress;
- **-** String destinationIPAddress;
- **methods:**
- **+** long getVersion();
- **+** void setVersion(long version);
- **+** long getHeaderLength();
- **+** void setHeaderLength(long headerLength);
- **+** String getDifferentiatedServiceField();
- **+** void setDifferentiatedServiceField(String differentiatedServiceField);
- **+** long getTotalLength();
- **+** void setTotalLength(long totalLength);
- **+** String getIdentification();
- **+** void setIdentification(String identification);
- **+** String getFlags();
- **+** void setFlags(String flags);
- **+** long getFragmentOffset();
- **+** void setFragmentOffset(long fragmentOffset);

- **+** long getTimeToLive();
- **+** void setTimeToLive(long timeToLive);
- **+** String getProtocol();
- **+** void setProtocol(String protocol);
- **+** String getHeaderChecksum();
- **+** void setHeaderChecksum(String headerChecksum);
- **+** String getSourceIPAddress();
- **+** void setSourceIPAddress(String sourceIPAddress);
- **+** String getDestinationIPAddress();
- **+** void setDestinationIPAddress(String destinationIPAddress);
-
- **class MyPcapPacket:** This class represents a Pcap Packet.
- **attributes:**
- **-** int number;
- **-** String protocol;
- **methods:**
- **+** int getNumber();
- **+** String getProtocol();
-
- **class MyTCPPacket:** This class represents a Tcp Packet.
- **attributes:**
- **-** long sourcePort;
- **-** long destinationPort;
- **-** long sequenceNumber;
- **-** long acknowlegmentNumber;
- **-** long windowSizeValue;
- **-** String checksum;
- **methods:**
- **+** long getSourcePort();
- **+** void setSourcePort(long sourcePort);
- **+** long getDestinationPort();
- **+** void setDestinationPort(long destinationPort);
- **+** long getSequenceNumber();
- **+** void setSequenceNumber(long sequenceNumber);
- **+** long getAcknowlegmentNumber();
- **+** void setAcknowlegmentNumber(long acknowlegmentNumber);
- **+** long getHeaderLength();
- **+** void setHeaderLength(long headerLength);
- **+** String getFlags();
- **+** void setFlags(String flags);
- **+** long getWindowSizeValue();
- **+** void setWindowSizeValue(long windowSizeValue);
- **+** String getChecksum();
- **+** void setChecksum(String checksum);
-
- **class MyUDPPacket:** This class represents a Udp Packet.
- **attributes:**
- **-** long sourcePort;
- **-** long destinationPort;
- **-** long length;
- **-** String checksum;

- **methods:**
- **+** long getSourcePort();
- **+** void setSourcePort(long sourcePort);
- **+** long getDestinationPort();
- **+** void setDestinationPort(long destinationPort);
- **+** long getLength();
- **+** void setLength(long length);
- **+** String getChecksum();
- **+** void setChecksum(String checksum);
-
-
- ***Package: shared.packet.interfaces***
-
- The package shared.packet.interfaces contains the interfaces IMyDhcpPacke.java, IMyEthernetFrame.java, IMyFinalPacket.java, IMyHttpPacket.java, IMyIcmpPacket.java, IMyIPv4Packet.java, IMyPcapPacket.java, IMyTcpPacket.java, IMyUdpPacket.java and IPacketCommand.java.
-
- **interface IMyDhcpPacke:** This interface represents a Dhcp Packet.
- **methods:**
- **+** long getMessageType();
- **+** void setMessageType(long messageType);
- **+** String getHardwareType();
- **+** void setHardwareType(String hardwareType);
- **+** long getHardwareAddressLength();
- **+** void setHardwareAddressLength(long hardwareAddressLength);
- **+** long getHops();
- **+** void setHops(long hops);
- **+** String getTransactionID();
- **+** void setTransactionID(String transactionID);
- **+** long getSecondsElapsed();
- **+** void setSecondsElapsed(long secondsElapsed);
- **+** String getBootpFlags();
- **+** void setBootpFlags(String bootpFlags);
- **+** String getClientIPAddress();
- **+** void setClientIPAddress(String clientIPAddress);
- **+** String getYourClientIPAddress();
- **+** void setYourClientIPAddress(String yourClientIPAddress);
- **+** String getNextServerIPAddress();
- **+** void setNextServerIPAddress(String nextServerIPAddress);
- **+** String getRelayAgentIPAddress();
- **+** void setRelayAgentIPAddress(String relayAgentIPAddress);
- **+** String getClientMacAddress();
- **+** void setClientMacAddress(String clientMacAddress);
- **+** long getClientHardwareAddressPadding();
- **+** void setClientHardwareAddressPadding(long clientHardwareAddressPadding);
- **+** String getServerHostname();
- **+** void setServerHostname(String serverHostname);
- **+** String getBootFilename();

- **+** void setBootFilename(String bootFilename);
- **+** String getMagicCookie();
- **+** void setMagicCookie(String magicCookie);
- **+** Vector<String> getOptions();
- **+** void setOptions(Vector<String> options);
-
- **interface IMyEthernetFrame:** This interface represents a EthernetFrame Packet.
- **methods:**
- **+** long getType();
- **+** void setType(long type);
- **+** String getArrivalTime();
- **+** void setArrivalTime(String arrivalTime);
- **+** double getEpochTime();
- **+** void setEpochTime(double epochTime);
- **+** long getFrameNumber();
- **+** void setFrameNumber(long frameNumber);
- **+** long getFrameLength();
- **+** void setFrameLength(long frameLenght);
- **+** long getCaptureLength();
- **+** void setCaptureLength(long captureLength);
- **+** String getDestinationAddress();
- **+** void setDestinationAddress(String destinationAddress);
- **+** String getSourceAddress();
- **+** void setSourceAddress(String sourceAddress);
-
- **interface IMyFinalePacket:** This interface represents the last Packet.
-
- **interface IMyHttpPacket:** This interface represents a Http Packet.
- **methods:**
- **+** String getRequestVersion();
- **+** void setRequestVersion(String requestVersion);
- **+** long getStatusCode();
- **+** void setStatusCode(long statusCode);
- **+** String getResponsePhrase();
- **+** void setResponsePhrase(String responsePhrase);
- **+** String getDate();
- **+** void setDate(String date);
- **+** String getServer();
- **+** void setServer(String server);
- **+** String getLocation();
- **+** void setLocation(String location);
- **+** String getVary();
- **+** void setVary(String vary);
- **+** String getContentEncoding();
- **+** void setContentEncoding(String contentEncoding);
- **+** long getContentLength();
- **+** void setContentLength(long contentLength);
- **+** long getKeepAlive();
- **+** void setKeepAlive(long keepAlive);
- **+** String getConnection();

- **+** void setConnection(String connection);
- **+** String getContentType();
- **+** void setContentType(String contentType);
- **+** String getContentEncodedEntityBody();
- **+** void setContentEncodedEntityBody(String contentEncodedEntityBody);
-
-
- **interface IMyIcmpPacket:** This interface represents a Icmp Packet.
- **methods:**
- **+** long getType();
- **+** void setType(long type);
- **+** long getCode();
- **+** void setCode(long code);
- **+** String getChecksum();
- **+** void setChecksum(String checksum);
- **+** long getIdentifierBE();
- **+** void setIdentifierBE(long identifierBE);
- **+** long getIdentifierLE();
- **+** void setIdentifierLE(long identifierLE);
- **+** long getSequenceNumberBE();
- **+** void setSequenceNumberBE(long sequenceNumberBE);
- **+** long getSequenceNumberLE();
- **+** void setSequenceNumberLE(long sequenceNumberLE);
- **+** String getData();
- **+** void setData(String data);
-
- **interface IMyIPv4Packet:** This interface represents a Ipv4 Packet.
- **methods:**
- **+** long getVersion();
- **+** void setVersion(long version);
- **+** long getHeaderLength();
- **+** void setHeaderLength(long headerLength);
- **+** String getDifferentiatedServiceField();
- **+** void setDifferentiatedServiceField(String differentiatedServiceField);
- **+** long getTotalLength();
- **+** void setTotalLength(long totalLength);
- **+** String getIdentification();
- **+** void setIdentification(String identification);
- **+** String getFlags();
- **+** void setFlags(String flags);
- **+** long getFragmentOffset();
- **+** void setFragmentOffset(long fragmentOffset);
- **+** long getTimeToLive();
- **+** void setTimeToLive(long timeToLive);
- **+** String getProtocol();
- **+** void setProtocol(String protocol);
- **+** String getHeaderChecksum();
- **+** void setHeaderChecksum(String headerChecksum);
- **+** String getSourceIPAddress();
- **+** void setSourceIPAddress(String sourceIPAddress);
- **+** String getDestinationIPAddress();

- **+** void setDestinationIPAddress(String destinationIPAddress);
-
- **interface IMyPcapPacket:** This interface represents a Pcap Packet.
- **methods:**
- **+** int getNumber();
- **+** String getProtocol();
-
- **interface IMyTcpPacket:** This interface represents a Tcp Packet.
- **methods:**
- **+** long getSourcePort();
- **+** void setSourcePort(long sourcePort);
- **+** long getDestinationPort();
- **+** void setDestinationPort(long destinationPort);
- **+** long getSequenceNumber();
- **+** void setSequenceNumber(long sequenceNumber);
- **+** long getAcknowlegmentNumber();
- **+** void setAcknowlegmentNumber(long acknowlegmentNumber);
- **+** long getHeaderLength();
- **+** void setHeaderLength(long headerLength);
- **+** String getFlags();
- **+** void setFlags(String flags);
- **+** long getWindowSizeValue();
- **+** void setWindowSizeValue(long windowSizeValue);
- **+** String getChecksum();
- **+** void setChecksum(String checksum);
-
- **interface IMyUdpPacket:** This interface represents a Udp Packet.
- **methods:**
- **+** long getSourcePort();
- **+** void setSourcePort(long sourcePort);
- **+** long getDestinationPort();
- **+** void setDestinationPort(long destinationPort);
- **+** long getLength();
- **+** void setLength(long length);
- **+** String getChecksum();
- **+** void setChecksum(String checksum);
-
- **interface IPacketCommand:** This interface represents a PacketCommand.
- **methods:**
- **+** String getPacketType();
- **+** void setPacketType(String packetType);

Package: shared

at.iaik.websharkgwt.shared.utils

This contains classes:
- Fieldverifier.java

- PacketCommand.java
- ServletLocations.java

class Fieldverifier:
attributes:

methods:
+ boolean isValidName(): This function returns false, if the name is null and otherwise it returns true.

class PacketCommand: This class represents
methods:
+ void setPacketType(String packetType): set the supplied packettype to the variable packetType.

+ String getPacketType(): It returns the packet type.

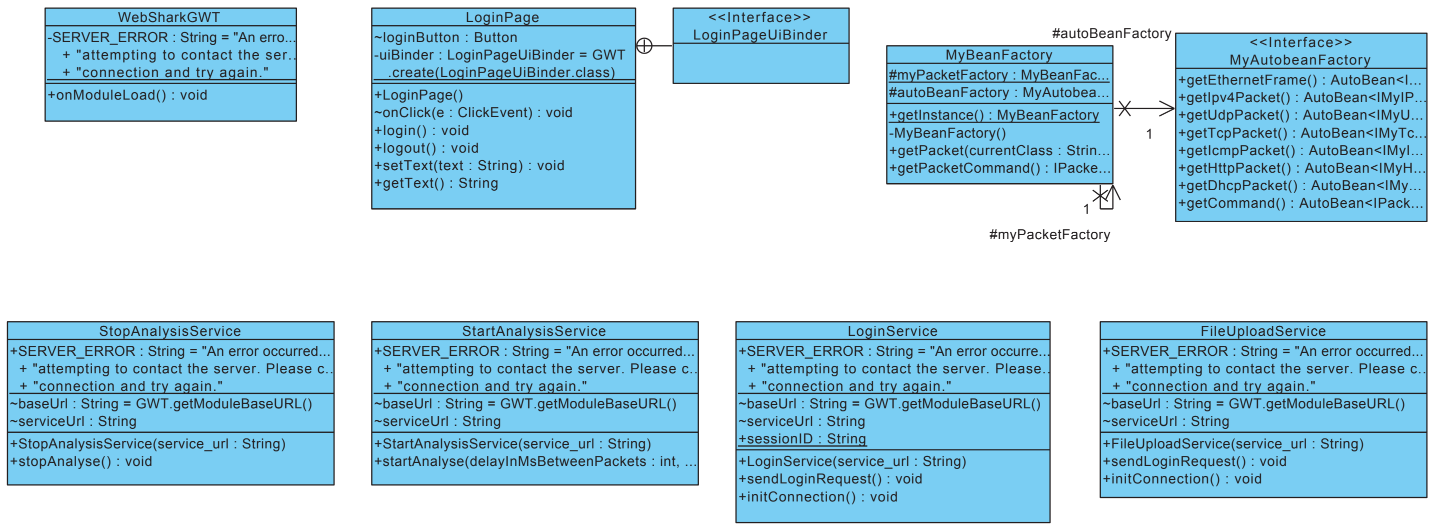
class ServletLocations: This class represents some static string to the Locations of the servlests and the file upload directory.
attributes:
+ static String BASE:
+ static String PCAP_FILE_UPLOAD_SERVLET:
+ static String FILE_UPLOAD_SERVLET:

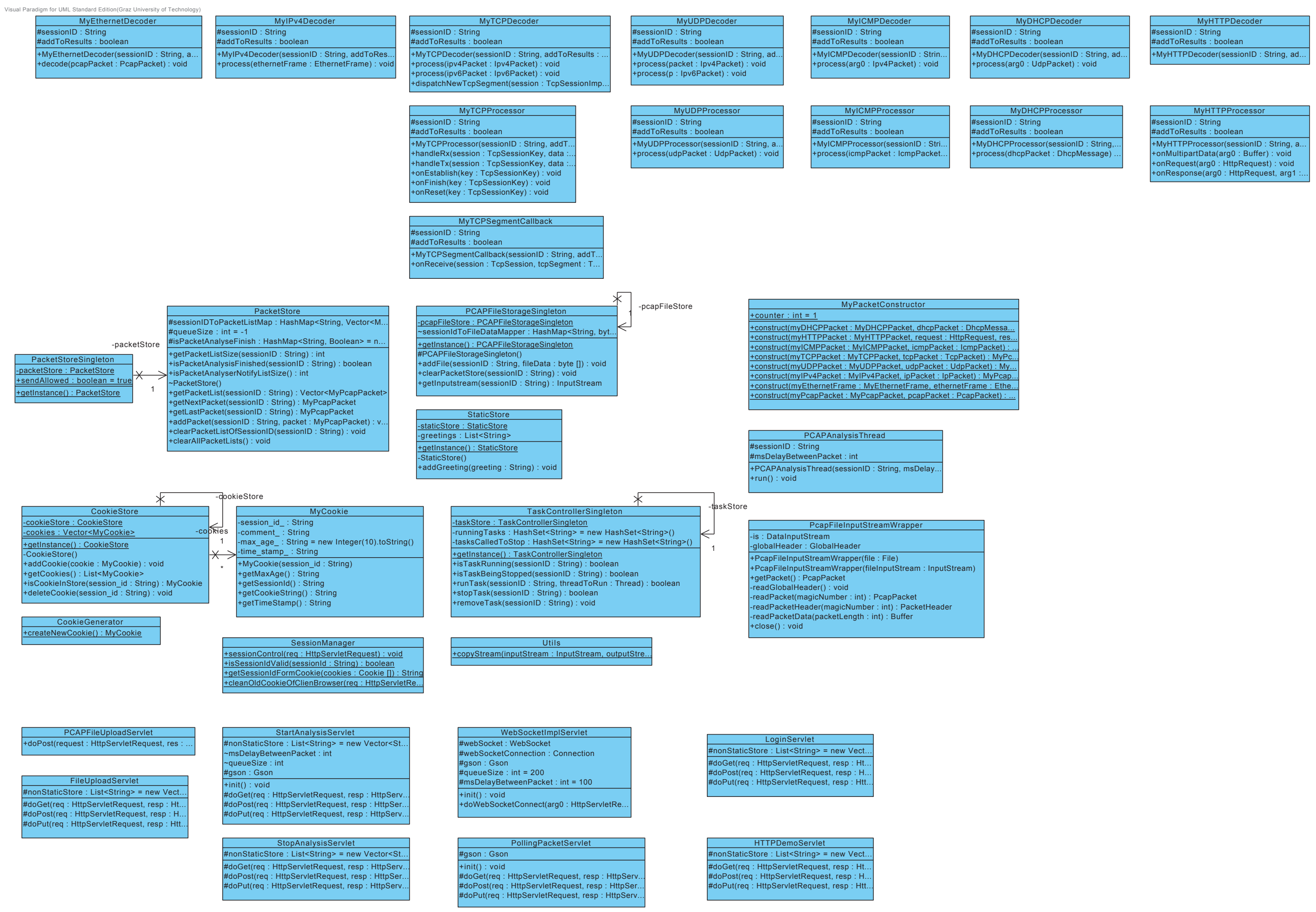
Package: shared.utils

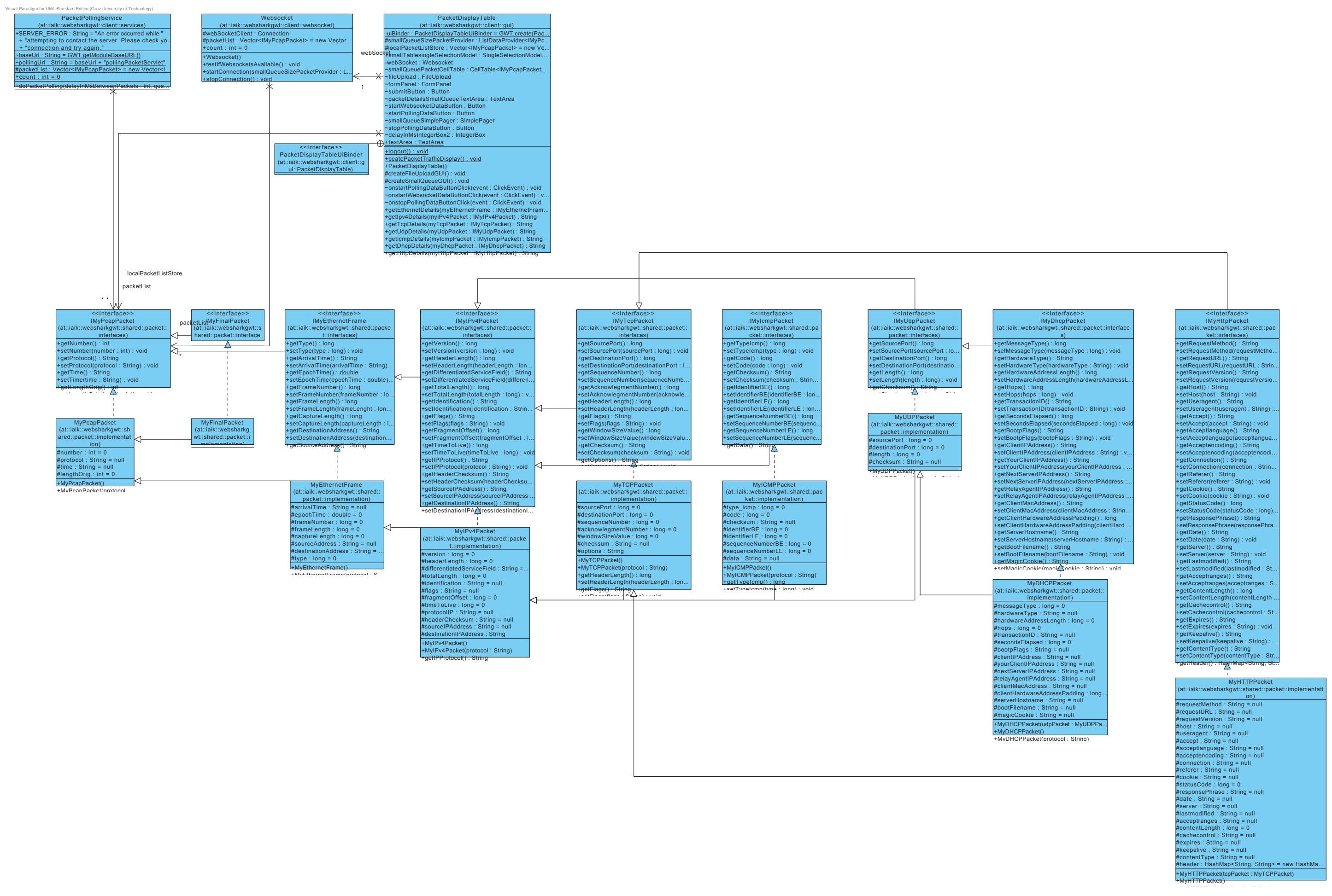
The package shared.utils contains the classes HttpRequestDebugger.java and HttpResponseDebugger.java.

class HttpRequestDebugger: This class represents a Debugger for Http Requests.
methods:
+ void printRequest(HttpServletRequest req)

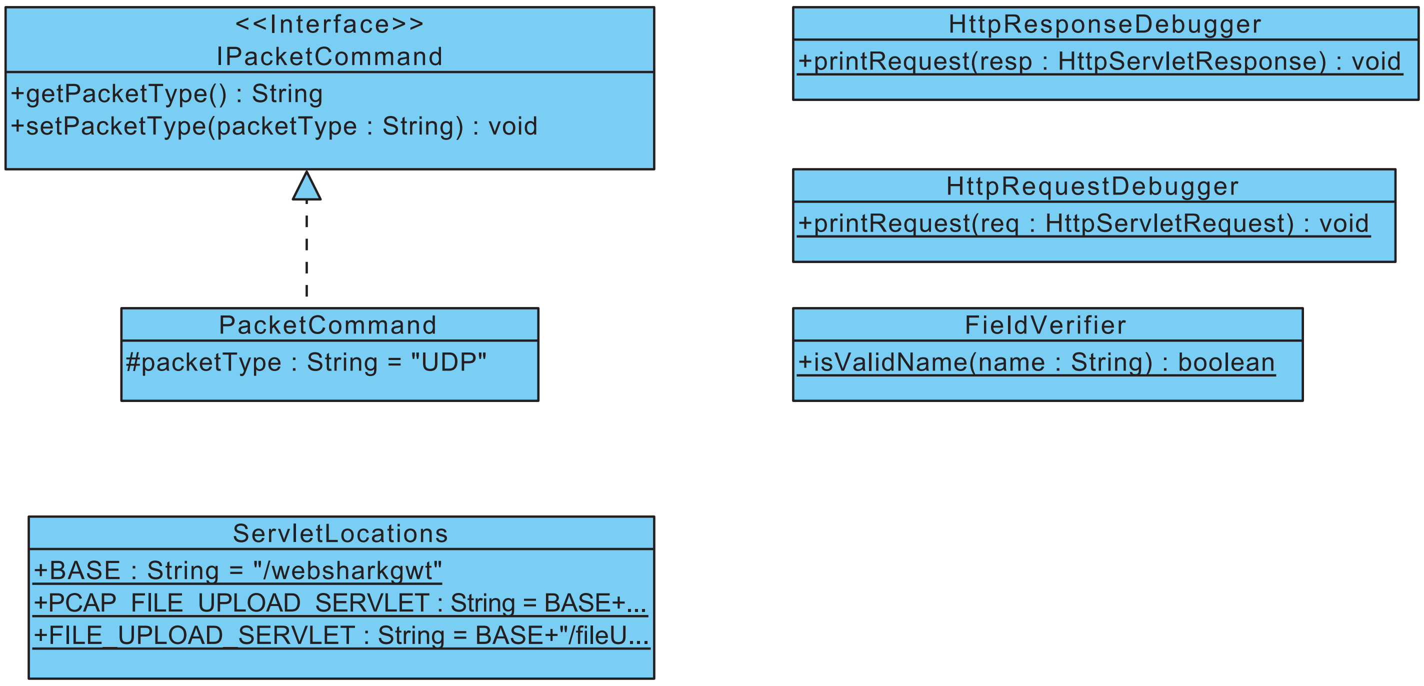
class HttpResponseDebugger: This class represents a Debugger for Http Response.
methods:
+ void printRequest(HttpServletResponse resp)


WebSharkGWT
-SERVER_ERROR : String = "An erro...
+ "attempting to contact the ser...
+ "connection and try again."
+onModuleLoad() : void
LoginPage
~loginButton : Button
-uiBinder : LoginPageUiBinder = GWT
.create(LoginPageUiBinder.class)
+LoginPage()
~onClick(e : ClickEvent) : void
+login() : void
+logout() : void
+setText(text : String) : void
+getText() : String
<<Interface>>
LoginPageUiBinder
#autoBeanFactory
MyBeanFactory
#myPacketFactory : MyBeanFac...
#autoBeanFactory : MyAutobea...
+getInstance() : MyBeanFactory
-MyBeanFactory()
+getPacket(currentClass : Strin...
+getPacketCommand() : IPacke...
1
1
#myPacketFactory
<<Interface>>
MyAutobeanFactory
+getEthernetFrame() : AutoBean<I...
+getIpv4Packet() : AutoBean<IMyIP...
+getUdpPacket() : AutoBean<IMyU...
+getTcpPacket() : AutoBean<IMyTc...
+getIcmpPacket() : AutoBean<IMyI...
+getHttpPacket() : AutoBean<IMyH...
+getDhcpPacket() : AutoBean<IMy...
+getCommand() : AutoBean<IPack...
StopAnalysisService
+SERVER_ERROR : String = "An error occurred...
+ "attempting to contact the server. Please c...
+ "connection and try again."
~baseUrl : String = GWT.getModuleBaseURL()
~serviceUrl : String
+StopAnalysisService(service_url : String)
+stopAnalyse() : void
StartAnalysisService
+SERVER_ERROR : String = "An error occurred...
+ "attempting to contact the server. Please c...
+ "connection and try again."
~baseUrl : String = GWT.getModuleBaseURL()
~serviceUrl : String
+StartAnalysisService(service_url : String)
+startAnalyse(delayInMsBetweenPackets : int, ...
LoginService
+SERVER_ERROR : String = "An error occurre...
+ "attempting to contact the server. Please...
+ "connection and try again."
~baseUrl : String = GWT.getModuleBaseURL()
~serviceUrl : String
+sessionID : String
+LoginService(service_url : String)
+sendLoginRequest() : void
+initConnection() : void
FileUploadService
+SERVER_ERROR : String = "An error occurred...
+ "attempting to contact the server. Please c...
+ "connection and try again."
~baseUrl : String = GWT.getModuleBaseURL()
~serviceUrl : String
+FileUploadService(service_url : String)
+sendLoginRequest() : void
+initConnection() : void


Visual Paradigm for UML Standard Edition(Graz University of Technology)
MyEthernetDecoder
#sessionID : String
#addToResults : boolean
+MyEthernetDecoder(sessionID : String, a...
+decode(pcapPacket : PcapPacket) : void
MyIPv4Decoder
#sessionID : String
#addToResults : boolean
+MyIPv4Decoder(sessionID : String, addToRes...
+process(ethernetFrame : EthernetFrame) : void
MyTCPDecoder
#sessionID : String
#addToResults : boolean
+MyTCPDecoder(sessionID : String, addToResults : ...
+process(ipv4Packet : Ipv4Packet) : void
+process(ipv6Packet : Ipv6Packet) : void
+dispatchNewTcpSegment(session : TcpSessionImp...
MyUDPDecoder
#sessionID : String
#addToResults : boolean
+MyUDPDecoder(sessionID : String, ad...
+process(packet : Ipv4Packet) : void
+process(p : Ipv6Packet) : void
MyICMPDecoder
#sessionID : String
#addToResults : boolean
+MyICMPDecoder(sessionID : Strin...
+process(arg0 : Ipv4Packet) : void
MyDHCPDecoder
#sessionID : String
#addToResults : boolean
+MyDHCPDecoder(sessionID : String, ad...
+process(arg0 : UdpPacket) : void
MyHTTPDecoder
#sessionID : String
#addToResults : boolean
+MyHTTPDecoder(sessionID : String, ad...
MyTCPProcessor
#sessionID : String
#addToResults : boolean
+MyTCPProcessor(sessionID : String, addT...
+handleRx(session : TcpSessionKey, data :...
+handleTx(session : TcpSessionKey, data :...
+onEstablish(key : TcpSessionKey) : void
+onFinish(key : TcpSessionKey) : void
+onReset(key : TcpSessionKey) : void
MyUDPProcessor
#sessionID : String
#addToResults : boolean
+MyUDPProcessor(sessionID : String, a...
+process(udpPacket : UdpPacket) : void
MyICMPProcessor
#sessionID : String
#addToResults : boolean
+MyICMPProcessor(sessionID : Stri...
+process(icmpPacket : IcmpPacket...
MyDHCPProcessor
#sessionID : String
#addToResults : boolean
+MyDHCPProcessor(sessionID : String,...
+process(dhcpPacket : DhcpMessage) ...
MyHTTPProcessor
#sessionID : String
#addToResults : boolean
+MyHTTPProcessor(sessionID : String, a...
+onMultipartData(arg0 : Buffer) : void
+onRequest(arg0 : HttpRequest) : void
+onResponse(arg0 : HttpRequest, arg1 :...
MyTCPSegmentCallback
#sessionID : String
#addToResults : boolean
+MyTCPSegmentCallback(sessionID : String, addT...
+onReceive(session : TcpSession, tcpSegment : T...
-packetStore
PacketStoreSingleton
-packetStore : PacketStore
+sendAllowed : boolean = true
+getInstance() : PacketStore
1
PacketStore
#sessionIDToPacketListMap : HashMap<String, Vector<M...
#queueSize : int = -1
#isPacketAnalyseFinish : HashMap<String, Boolean> = n...
+getPacketListSize(sessionID : String) : int
+isPacketAnalysisFinished(sessionID : String) : boolean
+isPacketAnalyserNotifyListSize() : int
~PacketStore()
+getPacketList(sessionID : String) : Vector<MyPcapPacket>
+getNextPacket(sessionID : String) : MyPcapPacket
+getLastPacket(sessionID : String) : MyPcapPacket
+addPacket(sessionID : String, packet : MyPcapPacket) : v...
+clearPacketListOfSessionID(sessionID : String) : void
+clearAllPacketLists() : void
-pcapFileStore
1
PCAPFileStorageSingleton
-pcapFileStore : PCAPFileStorageSingleton
~sessionIdToFileDataMapper : HashMap<String, byt...
+getInstance() : PCAPFileStorageSingleton
#PCAPFileStorageSingleton()
+addFile(sessionID : String, fileData : byte []) : void
+clearPacketStore(sessionID : String) : void
+getInputstream(sessionID : String) : InputStream
StaticStore
-staticStore : StaticStore
-greetings : List<String>
+getInstance() : StaticStore
-StaticStore()
+addGreeting(greeting : String) : void
MyPacketConstructor
+counter : int = 1
+construct(myDHCPPacket : MyDHCPPacket, dhcpPacket : DhcpMessa...
+construct(myHTTPPacket : MyHTTPPacket, request : HttpRequest, res...
+construct(myICMPPacket : MyICMPPacket, icmpPacket : IcmpPacket) : ...
+construct(myTCPPacket : MyTCPPacket, tcpPacket : TcpPacket) : MyPc...
+construct(myUDPPacket : MyUDPPacket, udpPacket : UdpPacket) : My...
+construct(myIPv4Packet : MyIPv4Packet, ipPacket : IpPacket) : MyPcap...
+construct(myEthernetFrame : MyEthernetFrame, ethernetFrame : Ethe...
+construct(myPcapPacket : MyPcapPacket, pcapPacket : PcapPacket) : ...
PCAPAnalysisThread
#sessionID : String
#msDelayBetweenPacket : int
+PCAPAnalysisThread(sessionID : String, msDelay...
+run() : void
-cookieStore
CookieStore
-cookieStore : CookieStore
-cookies : Vector<MyCookie>
+getInstance() : CookieStore
-CookieStore()
+addCookie(cookie : MyCookie) : void
+getCookies() : List<MyCookie>
+isCookieInStore(session_id : String) : MyCookie
+deleteCookie(session_id : String) : void
-cookies
1
*
MyCookie
-session_id_ : String
-comment_ : String
-max_age_ : String = new Integer(10).toString()
-time_stamp_ : String
+MyCookie(session_id : String)
+getMaxAge() : String
+getSessionId() : String
+getCookieString() : String
+getTimeStamp() : String
-taskStore
1
TaskControllerSingleton
-taskStore : TaskControllerSingleton
-runningTasks : HashSet<String> = new HashSet<String>()
-tasksCalledToStop : HashSet<String> = new HashSet<String>()
+getInstance() : TaskControllerSingleton
+isTaskRunning(sessionID : String) : boolean
+isTaskBeingStopped(sessionID : String) : boolean
+runTask(sessionID : String, threadToRun : Thread) : boolean
+stopTask(sessionID : String) : boolean
+removeTask(sessionID : String) : void
PcapFileInputStreamWrapper
-is : DataInputStream
-globalHeader : GlobalHeader
+PcapFileInputStreamWrapper(file : File)
+PcapFileInputStreamWrapper(fileInputStream : InputStream)
+getPacket() : PcapPacket
-readGlobalHeader() : void
-readPacket(magicNumber : int) : PcapPacket
-readPacketHeader(magicNumber : int) : PacketHeader
-readPacketData(packetLength : int) : Buffer
+close() : void
CookieGenerator
+createNewCookie() : MyCookie
SessionManager
+sessionControl(req : HttpServletRequest) : void
+isSessionIdValid(sessionId : String) : boolean
+getSessionIdFormCookie(cookies : Cookie []) : String
+cleanOldCookieOfClienBrowser(req : HttpServletRe...
Utils
+copyStream(inputStream : InputStream, outputStre...
PCAPFileUploadServlet
+doPost(request : HttpServletRequest, res : ...
FileUploadServlet
#nonStaticStore : List<String> = new Vect...
#doGet(req : HttpServletRequest, resp : Ht...
#doPost(req : HttpServletRequest, resp : H...
#doPut(req : HttpServletRequest, resp : Htt...
StartAnalysisServlet
#nonStaticStore : List<String> = new Vector<St...
~msDelayBetweenPacket : int
~queueSize : int
#gson : Gson
+init() : void
#doGet(req : HttpServletRequest, resp : HttpServ...
#doPost(req : HttpServletRequest, resp : HttpSer...
#doPut(req : HttpServletRequest, resp : HttpServ...
StopAnalysisServlet
#nonStaticStore : List<String> = new Vector<St...
#doGet(req : HttpServletRequest, resp : HttpServ...
#doPost(req : HttpServletRequest, resp : HttpSer...
#doPut(req : HttpServletRequest, resp : HttpServ...
WebSocketImplServlet
#webSocket : WebSocket
#webSocketConnection : Connection
#gson : Gson
#queueSize : int = 200
#msDelayBetweenPacket : int = 100
+init() : void
+doWebSocketConnect(arg0 : HttpServletRe...
PollingPacketServlet
#gson : Gson
+init() : void
#doGet(req : HttpServletRequest, resp : HttpServ...
#doPost(req : HttpServletRequest, resp : HttpSer...
#doPut(req : HttpServletRequest, resp : HttpServ...
LoginServlet
#nonStaticStore : List<String> = new Vect...
#doGet(req : HttpServletRequest, resp : Ht...
#doPost(req : HttpServletRequest, resp : H...
#doPut(req : HttpServletRequest, resp : Htt...
HTTPDemoServlet
#nonStaticStore : List<String> = new Vect...
#doGet(req : HttpServletRequest, resp : Ht...
#doPost(req : HttpServletRequest, resp : H...
#doPut(req : HttpServletRequest, resp : Htt...



Visual Paradigm for UML Standard Edition(Graz University of Technology)
PacketPollingService
(at::iaik::websharkgwt::client::services)
+SERVER_ERROR : String = "An error occurred while "
+ "attempting to contact the server. Please check yo...
+ "connection and try again."
~baseUrl : String = GWT.getModuleBaseURL()
~pollingUrl : String = baseUrl + "pollingPacketServlet"
#packetList : Vector<IMyPcapPacket> = new Vector<I...
+count : int = 0
+doPacketPolling(delayInMsBetweenPackets : int, que...
Websocket
(at::iaik::websharkgwt::client::websocket)
#webSocketClient : Connection
#packetList : Vector<IMyPcapPacket> = new Vector...
+count : int = 0
+Websocket()
+testIfWebsocketsAvaliable() : void
+startConnection(smallQueueSizePacketProvider : L...
+stopConnection() : void
webSocket
1
PacketDisplayTable
(at::iaik::websharkgwt::client::gui)
-uiBinder : PacketDisplayTableUiBinder = GWT.create(Pac...
#smallQueueSizePacketProvider : ListDataProvider<IMyPc...
#localPacketListStore : Vector<IMyPcapPacket> = new Ve...
+smallTablesingleSelectionModel : SingleSelectionModel...
-webSocket : Websocket
~smallQueuePacketCellTable : CellTable<IMyPcapPacket...
~fileUpload : FileUpload
~formPanel : FormPanel
~submitButton : Button
~packetDetailsSmallQueueTextArea : TextArea
~startWebsocketDataButton : Button
~startPollingDataButton : Button
~smallQueueSimplePager : SimplePager
~stopPollingDataButton : Button
~delayInMsIntegerBox2 : IntegerBox
+textArea : TextArea
+logout() : void
+ceatePacketTrafficDisplay() : void
+PacketDisplayTable()
#createFileUploadGUI() : void
#createSmallQueueGUI() : void
~onstartPollingDataButtonClick(event : ClickEvent) : void
~onstartWebsocketDataButtonClick(event : ClickEvent) : v...
~onstopPollingDataButtonClick(event : ClickEvent) : void
+getEthernetDetails(myEthernetFrame : IMyEthernetFram...
+getIpv4Details(myIPv4Packet : IMyIPv4Packet) : String
+getTcpDetails(myTcpPacket : IMyTcpPacket) : String
+getUdpDetails(myUdpPacket : IMyUdpPacket) : String
+getIcmpDetails(myIcmpPacket : IMyIcmpPacket) : String
+getDhcpDetails(myDhcpPacket : IMyDhcpPacket) : String
+getHttpDetails(myHttpPacket : IMyHttpPacket) : String
<<Interface>>
PacketDisplayTableUiBinder
(at::iaik::websharkgwt::client::gui::PacketDisplayTable)
localPacketListStore
packetList
* *
*
<<Interface>>
IMyPcapPacket
(at::iaik::websharkgwt::shared::packet::interfaces)
+getNumber() : int
+setNumber(number : int) : void
+getProtocol() : String
+setProtocol(protocol : String) : void
+getTime() : String
+setTime(time : String) : void
+getLengthOrig() : int
packetList
<<Interface>>
IMyFinalPacket
(at::iaik::websharkgwt::shared::packet::interface
MyPcapPacket
(at::iaik::websharkgwt::shared::packet::implementation)
#number : int = 0
#protocol : String = null
#time : String = null
#lengthOrig : int = 0
+MyPcapPacket()
MyFinalPacket
(at::iaik::websharkgwt::shared::packet::implementation)
<<Interface>>
IMyEthernetFrame
(at::iaik::websharkgwt::shared::packet::interfaces)
+getType() : long
+setType(type : long) : void
+getArrivalTime() : String
+setArrivalTime(arrivalTime : String)...
+getEpochTime() : double
+setEpochTime(epochTime : double)...
+getFrameNumber() : long
+setFrameNumber(frameNumber : lo...
+getFrameLength() : long
+setFrameLength(frameLenght : lon...
+getCaptureLength() : long
+setCaptureLength(captureLength : l...
+getDestinationAddress() : String
+setDestinationAddress(destination...
+getSourceAddress() : String
MyEthernetFrame
(at::iaik::websharkgwt::shared::packet::implementation)
#arrivalTime : String = null
#epochTime : double = 0
#frameNumber : long = 0
#frameLength : long = 0
#captureLength : long = 0
#sourceAddress : String = null
#destinationAddress : String = ...
#type : long = 0
+MyEthernetFrame()
<<Interface>>
IMyIPv4Packet
(at::iaik::websharkgwt::shared::packet::interfaces)
+getVersion() : long
+setVersion(version : long) : void
+getHeaderLength() : long
+setHeaderLength(headerLength : lon...
+getDifferentiatedServiceField() : String
+setDifferentiatedServiceField(differen...
+getTotalLength() : long
+setTotalLength(totalLength : long) : v...
+getIdentification() : String
+setIdentification(identification : Strin...
+getFlags() : String
+setFlags(flags : String) : void
+getFragmentOffset() : long
+setFragmentOffset(fragmentOffset : l...
+getTimeToLive() : long
+setTimeToLive(timeToLive : long) : void
+getIPProtocol() : String
+setIPProtocol(protocol : String) : void
+getHeaderChecksum() : String
+setHeaderChecksum(headerChecksu...
+getSourceIPAddress() : String
+setSourceIPAddress(sourceIPAddress ...
+getDestinationIPAddress() : String
+setDestinationIPAddress(destinationI...
MyIPv4Packet
(at::iaik::websharkgwt::shared::packet::implementation)
#version : long = 0
#headerLength : long = 0
#differentiatedServiceField : String =...
#totalLength : long = 0
#identification : String = null
#flags : String = null
#fragmentOffset : long = 0
#timeToLive : long = 0
#protocolIP : String = null
#headerChecksum : String = null
#sourceIPAddress : String = null
#destinationIPAddress : String
+MyIPv4Packet()
+MyIPv4Packet(protocol : String)
+getIPProtocol() : String
<<Interface>>
IMyTcpPacket
(at::iaik::websharkgwt::shared::packet::interfaces)
+getSourcePort() : long
+setSourcePort(sourcePort : long) : void
+getDestinationPort() : long
+setDestinationPort(destinationPort : l...
+getSequenceNumber() : long
+setSequenceNumber(sequenceNumb...
+getAcknowlegmentNumber() : long
+setAcknowlegmentNumber(acknowle...
+getHeaderLength() : long
+setHeaderLength(headerLength : lon...
+getFlags() : String
+setFlags(flags : String) : void
+getWindowSizeValue() : long
+setWindowSizeValue(windowSizeValu...
+getChecksum() : String
+setChecksum(checksum : String) : void
+getOptions() : String
MyTCPPacket
(at::iaik::websharkgwt::shared::packet::implementation)
#sourcePort : long = 0
#destinationPort : long = 0
#sequenceNumber : long = 0
#acknowlegmentNumber : long = 0
#windowSizeValue : long = 0
#checksum : String = null
#options : String
+MyTCPPacket()
+MyTCPPacket(protocol : String)
+getHeaderLength() : long
+setHeaderLength(headerLength : lon...
+getFlags() : String
<<Interface>>
IMyIcmpPacket
(at::iaik::websharkgwt::shared::packet::interfaces)
+getTypeIcmp() : long
+setTypeIcmp(type : long) : void
+getCode() : long
+setCode(code : long) : void
+getChecksum() : String
+setChecksum(checksum : Strin...
+getIdentifierBE() : long
+setIdentifierBE(identifierBE : lon...
+getIdentifierLE() : long
+setIdentifierLE(identifierLE : lon...
+getSequenceNumberBE() : long
+setSequenceNumberBE(sequenc...
+getSequenceNumberLE() : long
+setSequenceNumberLE(sequenc...
+getData() : String
MyICMPPacket
(at::iaik::websharkgwt::shared::packet::implementation)
#type_icmp : long = 0
#code : long = 0
#checksum : String = null
#identifierBE : long = 0
#identifierLE : long = 0
#sequenceNumberBE : long = 0
#sequenceNumberLE : long = 0
#data : String = null
+MyICMPPacket()
+MyICMPPacket(protocol : String)
+getTypeIcmp() : long
+setTypeIcmp(type : long) : void
<<Interface>>
IMyUdpPacket
(at::iaik::websharkgwt::shared::packet::interfaces)
+getSourcePort() : long
+setSourcePort(sourcePort : lo...
+getDestinationPort() : long
+setDestinationPort(destinatio...
+getLength() : long
+setLength(length : long) : void
+getChecksum() : String
MyUDPPacket
(at::iaik::websharkgwt::shared::packet::implementation)
#sourcePort : long = 0
#destinationPort : long = 0
#length : long = 0
#checksum : String = null
+MyUDPPacket()
<<Interface>>
IMyDhcpPacket
(at::iaik::websharkgwt::shared::packet::interfaces)
+getMessageType() : long
+setMessageType(messageType : long) : void
+getHardwareType() : String
+setHardwareType(hardwareType : String) : void
+getHardwareAddressLength() : long
+setHardwareAddressLength(hardwareAddressL...
+getHops() : long
+setHops(hops : long) : void
+getTransactionID() : String
+setTransactionID(transactionID : String) : void
+getSecondsElapsed() : long
+setSecondsElapsed(secondsElapsed : long) : void
+getBootpFlags() : String
+setBootpFlags(bootpFlags : String) : void
+getClientIPAddress() : String
+setClientIPAddress(clientIPAddress : String) : v...
+getYourClientIPAddress() : String
+setYourClientIPAddress(yourClientIPAddress : ...
+getNextServerIPAddress() : String
+setNextServerIPAddress(nextServerIPAddress :...
+getRelayAgentIPAddress() : String
+setRelayAgentIPAddress(relayAgentIPAddress :...
+getClientMacAddress() : String
+setClientMacAddress(clientMacAddress : Strin...
+getClientHardwareAddressPadding() : long
+setClientHardwareAddressPadding(clientHard...
+getServerHostname() : String
+setServerHostname(serverHostname : String) : ...
+getBootFilename() : String
+setBootFilename(bootFilename : String) : void
+getMagicCookie() : String
+setMagicCookie(magicCookie : String) : void
MyDHCPPacket
(at::iaik::websharkgwt::shared::packet::implementation)
#messageType : long = 0
#hardwareType : String = null
#hardwareAddressLength : long = 0
#hops : long = 0
#transactionID : String = null
#secondsElapsed : long = 0
#bootpFlags : String = null
#clientIPAddress : String = null
#yourClientIPAddress : String = null
#nextServerIPAddress : String = null
#relayAgentIPAddress : String = null
#clientMacAddress : String = null
#clientHardwareAddressPadding : long...
#serverHostname : String = null
#bootFilename : String = null
#magicCookie : String = null
+MyDHCPPacket(udpPacket : MyUDPPa...
+MyDHCPPacket()
+MyDHCPPacket(protocol : String)
<<Interface>>
IMyHttpPacket
(at::iaik::websharkgwt::shared::packet::interfaces)
+getRequestMethod() : String
+setRequestMethod(requestMetho...
+getRequestURL() : String
+setRequestURL(requestURL : Strin...
+getRequestVersion() : String
+setRequestVersion(requestVersio...
+getHost() : String
+setHost(host : String) : void
+getUseragent() : String
+setUseragent(useragent : String) :...
+getAccept() : String
+setAccept(accept : String) : void
+getAcceptlanguage() : String
+setAcceptlanguage(acceptlangua...
+getAcceptencoding() : String
+setAcceptencoding(acceptencodi...
+getConnection() : String
+setConnection(connection : Strin...
+getReferer() : String
+setReferer(referer : String) : void
+getCookie() : String
+setCookie(cookie : String) : void
+getStatusCode() : long
+setStatusCode(statusCode : long)...
+getResponsePhrase() : String
+setResponsePhrase(responsePhra...
+getDate() : String
+setDate(date : String) : void
+getServer() : String
+setServer(server : String) : void
+getLastmodified() : String
+setLastmodified(lastmodified : St...
+getAcceptranges() : String
+setAcceptranges(acceptranges : S...
+getContentLength() : long
+setContentLength(contentLength ...
+getCachecontrol() : String
+setCachecontrol(cachecontrol : St...
+getExpires() : String
+setExpires(expires : String) : void
+getKeepalive() : String
+setKeepalive(keepalive : String) : ...
+getContentType() : String
+setContentType(contentType : Str...
+getHeader() : HashMap<String, St...
MyHTTPPacket
(at::iaik::websharkgwt::shared::packet::implementation)
#requestMethod : String = null
#requestURL : String = null
#requestVersion : String = null
#host : String = null
#useragent : String = null
#accept : String = null
#acceptlanguage : String = null
#acceptencoding : String = null
#connection : String = null
#referer : String = null
#cookie : String = null
#statusCode : long = 0
#responsePhrase : String = null
#date : String = null
#server : String = null
#lastmodified : String = null
#acceptranges : String = null
#contentLength : long = 0
#cachecontrol : String = null
#expires : String = null
#keepalive : String = null
#contentType : String = null
#header : HashMap<String, String> = new HashMa...
+MyHTTPPacket(tcpPacket : MyTCPPacket)
+MyHTTPPacket()



<<Interface>>
IPacketCommand
+getPacketType() : String
+setPacketType(packetType : String) : void
PacketCommand
#packetType : String = "UDP"
HttpResponseDebugger
+printRequest(resp : HttpServletResponse) : void
HttpRequestDebugger
+printRequest(req : HttpServletRequest) : void
FieldVerifier
+isValidName(name : String) : boolean
ServletLocations
+BASE : String = "/websharkgwt"
+PCAP_FILE_UPLOAD_SERVLET : String = BASE+...
+FILE_UPLOAD_SERVLET : String = BASE+"/fileU...