# Developer Guide

**jWebSocket  
Audio and Video transmission using jWebSocket framework.**

**1.0**

**Version Control**

|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 24/04/2012 | 1.0 | Document Creation | Alexander Rojas Hernández |

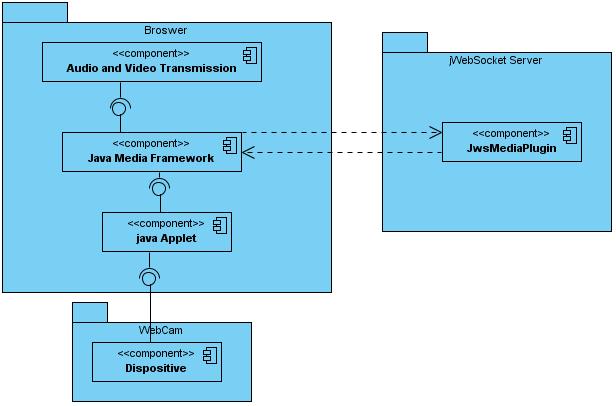
# 

# 1. General Vision.

The Project has as main objetive to increase the jWebSocket potentialities, developing new functionalities that allow to create web applications in real time with the benefics offered by the streaming technology. For these purposes there are 2 applications to work with the basics functionalities of this technology. A stationary application using the Java Media Framework (JMF) and a native application using PhoneGap.

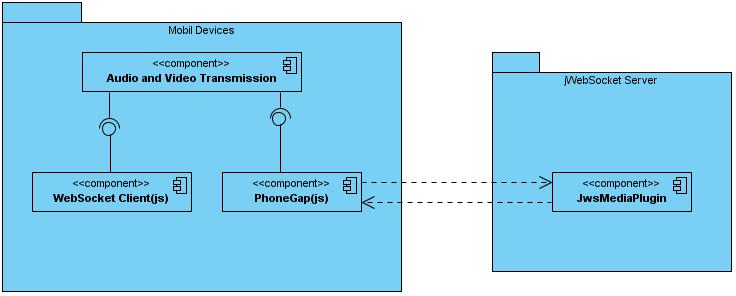
# 2. Solution Infrastructure.

# One introductional sentence here

****

**Figure1: Structure for the stationary application.**

To handle the capture process the Java Media Framework is used as API for low level encoding and sending (better “exchanging”?) the packets to the server.



**Figure2: Structure for the native application**

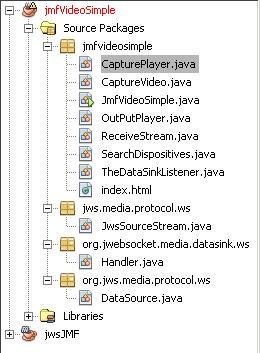
The PhoneGap framework is the access point to the differents SmartPhone controllers allowing the capture and the processing for images, audio and video data. It can be integrated with other libraries that work with similar devices.

The use of the jWebSocket server to potential the streaming technology provides to the users create applications with high levels of security in the transmission process, allowing a confiability (?? Don’t know this word) in the service in enviorments where needs a lot of users traffic and high speed in the communication. Also it demonstrates the high prestations (?? What does mean) that offers WebSocket protocol for this purposes.

# 3. Modularization of the extention

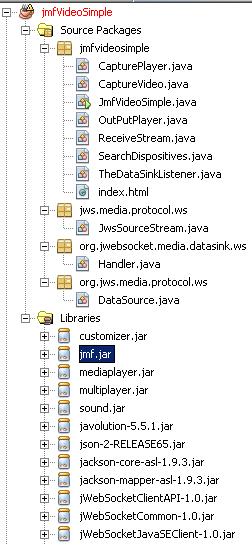
# 3.1 Stationary Application

The main component where the capture methods are executed is inside the jmfVideoSimple packet, figure 3 shows the packets structure.



**Figura3: Structure of the java packets in the stationary application.**

The jmfvideosimple contains the main classes which start the capture process of the audio and video signal. First a search is made to find the installed controllers.

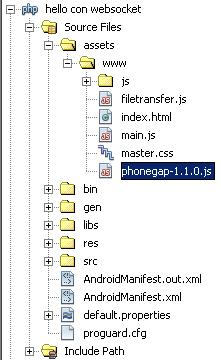


**Figura4: Structure of the java packets for the stationary application with the JMF libraries.**

Figure 4 shows the Java Media Framework library, the jmf.jar is the most important packet.

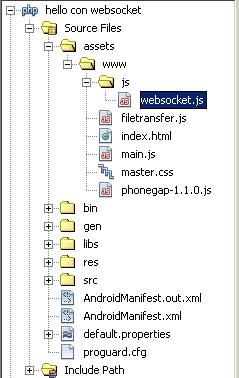
**3.1 Aplicación Nativa (translate to English)**

The main file that must be includes is the PhoneGap framework to interact with the Smartphone controllers.



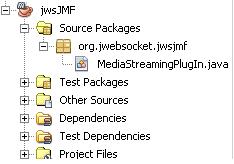
**Figure5: Structure of the native application with the PhoneGap library.**

Another important file to be included is the JavaScript jWebSocket client to establish the communication with the server, shown in figure 6.

****

**Figure 6: Structure of the native application with the jWebSocket client.**

The packet where is jWebSocket server in both applications contains the libraries to make applications based in HTML5 WebSocket, the framework reuse WebSocket implementations in the server side as engines. This sentences is not understandable! Please compare with your spanish version and translate again.

**

**Figura7: Structure for the server.**

# 4. Structure of source code

**4.1 Maven configuration**

The jwsJMF module configuration is located in the file pom.xml, an XML representation for the project is inside the folder that allows to configure the dependencies and basics properties for the project.

<groupId>org.jwebsocket</groupId>

<artifactId>jwsJMF</artifactId>

<version>1.0</version>

<packaging>jar</packaging>

<name>jwsJMF</name>

<url>http://maven.apache.org</url>

<build>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>2.3.2</version>

<configuration>

<source>1.6</source>

<target>1.6</target>

</configuration>

</plugin>

</plugins>

</build>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

</properties>

<dependencies>

<dependency>

<groupId>org.jwebsocket</groupId>

<artifactId>jWebSocketServer</artifactId>

<version>1.0</version>

</dependency>

<dependency>

<groupId>junit</groupId>

<artifactId>junit</artifactId>

<version>3.8.1</version>

<scope>test</scope>

</dependency>

</dependencies>

**4.2 Estructura de paquetes (Translate!)**

**4.2.1 Packets structure for the stationary application**

The **jmfvideosimple** packetis composed of the following libraries:

|  |  |
| --- | --- |
| Libraries | Descriptions |
| [CapturePlayer](file:///D:\TESIS\sourceCode\ijwssvn\branches\jWebSocket-1.0\javadocs\jWebSocketServer\apidocs\org\jwebsocket\grizzly\GrizzlyConnector.html) | Start the capture process and the player creation where shows and controls the events of the captured information. |
| CaptureVideo | Captures the Medialocators and creates the DataSources to be send to the player |
| OutPutPlayer | Distributes the information transmitted to the other clients |
| ReceiveStream | Receives the data flow sent from the server and creates the output DataSource to the player. |
| SearchDispositives | Searches the audio and video controllers installed in the PC |
| TheDataSinkListener | Controls the DataSink states (what is that?) |

The **jws.media.protocol.ws** packet contains the following library:

|  |  |
| --- | --- |
| Library | Description |
| JwsSourceStream | Analize the data flow. |

The **org.jwebsocket.media.datasink.ws** packet contains the following library:

|  |  |
| --- | --- |
| Library | Description |
| Handler | Realize the writer process of the capture data. |

The **org.jws.media.protocol.ws** packet contains the following library:

|  |  |
| --- | --- |
| Library | Description |
| DataSource | Create a DataSource with the received data from the server to send to the player |

**4.2.2 Packet structure for the native application**

The **www** packet packet is contains the following libraries:

|  |  |
| --- | --- |
| Library | Description |
| Websocket.js | jWebSocket client to establish the communication with the server. |
| Filetransfer.js | Sends the packets to the server. |
| Index.html | Shows the main interface of the application in the SmartPhone |
| main.js | Creates the capture methods and processing of the information |
| master.css | Controls the application styles |
| phonegap.js | Framework to work with the Smartphone controllers |

**4.3 Detail Description of the API. Stationary Application**

Detail explanation of the libraries: (please make a complete sentence!)

CapturePlayer

|  |  |  |
| --- | --- | --- |
| Type | Method | Description |
| void | [**init**](file:///D:\TESIS\sourceCode\ijwssvn\branches\jWebSocket-1.0\javadocs\jWebSocketServer\apidocs\org\jwebsocket\grizzly\GrizzlyConnector.html#generateUID())() | Initializes the application in general, e.g. as the connection to the jWebSocket server. |
| void | **processToken(WebSocketClientEvent wsce, Token token)** | Listener that analizes and processes the token that are captured |
| void | **processOpening(WebSocketClientEvent wsce)** | Listener that excecutes when the connection process is opening. |
| void | **processOpened(WebSocketClientEvent wsce)** | This method is fired when the connection process is opened. |
| void | **processPacket(WebSocketClientEvent wsce, WebSocketPacket wsp)** | Processes the incoming data packets |
| void | **processClosed(WebSocketClientEvent wsce)** | Fired when the connection is closed |
| void | **processReconnecting(WebSocketClientEvent wsce)** | Realized the reconnection process . |

CaptureVideo

|  |  |  |
| --- | --- | --- |
| Type | Method | Description |
| void | **CaptureVideo**() | Class constructor where register the “org.jwebsocket” packet. |
| DataSource | **getPlayerDataSource**() | Returns the DataSource that processes the player. |
| MediaLocator | **getVideoLocator** () | Search for the video controllers and captures the first available for the webcam |
| MediaLocator | **getAudioLocator** () | Searches in the audio controllers and captures the first available for the microphone. |
| void | **startCaptureVideo**() | Start all the data process where creates a DataSources to the device and realize the different kinds of the transmission process packets. |

OutPutPlayer

|  |  |  |
| --- | --- | --- |
| Type | Method | Description |
| void | [**init**](file:///D:\TESIS\sourceCode\ijwssvn\branches\jWebSocket-1.0\javadocs\jWebSocketServer\apidocs\org\jwebsocket\grizzly\GrizzlyConnector.html#generateUID())() | Starts the application in general, verifies the server connection to build the DataSource |
|  |  |  |
|  |  |  |

ReceiveStream

|  |  |  |
| --- | --- | --- |
| Type | Method | Description |
| void | **ReceiveStream**() | Constructor of the class where register the packet “org.jws” . |
| DataSource | **getOutputplayerDataSource**() | Returns the DataSource that processes the player. |
|  |  |  |
|  |  |  |

SearchDispositives

|  |  |  |
| --- | --- | --- |
| Type | Method | Description |
| void | **Scaner**() | Searches the audio and video controllers installed in the pc |
|  |  |  |

TheDataSinkListener

|  |  |  |
| --- | --- | --- |
| Type | Method | Description |
| void | **dataSinkUpdate(DataSinkEvent event)** | Controls the flag to verify when the DataSink object end the transmission. |
| void | **waitEndOfStream(long checkTimeMs)** | Controls the threads of the different transmission process of the DataSink. |

JwsSourceStream

|  |  |  |
| --- | --- | --- |
| Type | Method | Description |
| int | **read(byte[] buffer, int offset, int length)** | Read the incoming data flow and returns the bytes reads. |
| ContentDescriptor | **getContentDescriptor** () | Returns the description of kind of content. |
| long | **getContentLength()** | Returns the size of the content |
| Object[] | **getControls** | Returns the object array with the controls of the transmission. |

Handler

|  |  |  |
| --- | --- | --- |
| Type | Method | Description |
| void | **setSource(DataSource ds)** | Verify the type the DataSource that will transfer PullDataSource or PushDataSource |
| void | **setOutputLocator(MediaLocator output)** | Put to the MediaLocator the output Process of the transmission |
| void | **open()** | Open the conecction and verify if the state has been initialized |
| void | **start()** | Begins the transmission of the packets after to check the open state. |
| void | **stop()** | Detiene la transferencia si no ha sido conectado e inicializado.  Stop the transfer if don´t have been connected and initialized. |
| void | **close(String reason)** | Close the process of the data transmission and send to the DataSinkErrorEvent the reason and wake up the writer thread. |
| synchronized void | **transferData(PushSourceStream pss)** | Maneja los procesos de sincronización entre los distintos buffers que se utilizan para la transmisión.  Handle the synchronization Process among differents buffers that use to the transmission. |
| void | **run()** | Method to work asynchrony to the writer Process. |
| void | **write(byte[] aBuffer, long aLocation, int aLength)** | Sends the captured packets to the jWebSocket |

DataSource

|  |  |  |
| --- | --- | --- |
| Type | Method | Description |
| string | **getContentType()** | Returns the content type of what? Examples? |
| MediaLocator | **getLocator**() | Returns the medialocator that belongs to the parent element what means “parent element”? |
| void | **initCheck()** | Ckecks if has some MediaLocator from the server (exists, works or what?) |
| void | **connect()** | Creates the jWebSocket to connect and capture the streaming the server via token |
| void | **processToken(WebSocketClientEvent wsce, Token aToken)** | Receives the token with the data flow codify (codify? What do you mean here?). |
| void | **processOpening(WebSocketClientEvent wsce)** | Listener that excecutes when the connection process is opening |
| void | **processOpened(WebSocketClientEvent wsce)** | Fired when the conection process is opened |
| void | **processPacket(WebSocketClientEvent wsce, WebSocketPacket wsp)** | Processes the incoming data packets . |
| void | **processClosed(WebSocketClientEvent wsce)** | Fires when the connection is stopped. |
| void | **processReconnecting(WebSocketClientEvent wsce)** | Realize the reconnection process . |
| PullSourceStream[] | **getStreams()** | Returns an array of t he captured streams in the ProcessToken method. |

**4.4 Detail explanation of the API. Native application.**

main

|  |  |  |
| --- | --- | --- |
| Tipo | Método (Translate titles!) | Descripción |
| void | **init()** | This method is called on start of the application, it verifies that the mobile device is ready. |
| void | **openWebSocket**() | Realizes the connection to the server and control the states. |
| void | **initApp()** | IExecuted when the connection is opened |
| void | **captureImage()** | Accesses the capture image controller from the movil device and controls the success and failure callbacks |
| void | **captureVideogetPicture()** | Searches in the stored files(in this case videos) to send it to the server. |
| void | **captureVideo(**) | Accesses the smartphone video controllers and controls the success and failure callbacks. |
| void | **captureAudio**() | Accesses the smartphone audio controllers and controls the success and failure callbacks. |
| void | **uploadFile(aimageData)** | Base64 encodes the data to be send to the server |
| void | **uploadAudioFile**(amediaFile) | Establishes the parameters of the audio file captured to be send to the server. |
| void | **uploadVideoFile**(amediaFile) | Establishes the parameters of the video (?) file captured to be send to the server. |
|  |  |  |

Filetransfer

|  |  |  |
| --- | --- | --- |
| Tipo | Método | Descripción |
| void | **FileTransfer.prototype.jwsupload(filePath, server, successCallback, errorCallback, options, debug)** | Método reimplementado de PhoneGap para llevar a cabo el envío al servidor el fichero capturado.  Reimplemented PhoneGap method to send the data to the server. |

# 5. Conclutions

After you have read the development guide you will be familiar with the structure of the applications that uses the streaming technology provided by the jWebSocket framework, knowing the main implementations details, furthermore, you will have advanced knowledge of the dependencies, as well as of the main files needed to configure the applications.

The fact that jWebSocket is used as framework for the streaming technology makes this solution much more extensible and reliable.