# Development Guide

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

**1.0**

**Version Control**

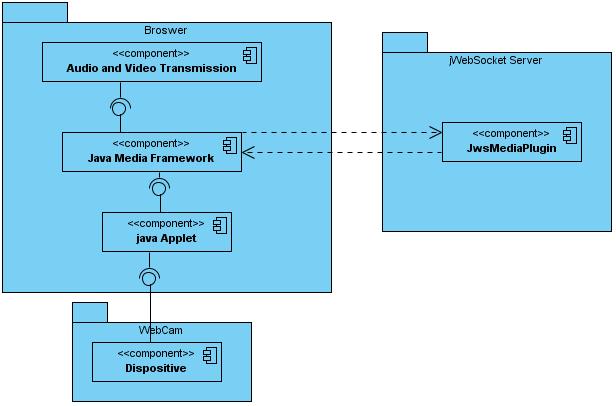
|  |  |  |  |
| --- | --- | --- | --- |
| **Date** | **Version** | **Description** | **Author** |
| 24/04/2012 | 1.0 | Document Creation | Alexander Rojas Hernández |

# 

# 1. General Vision.

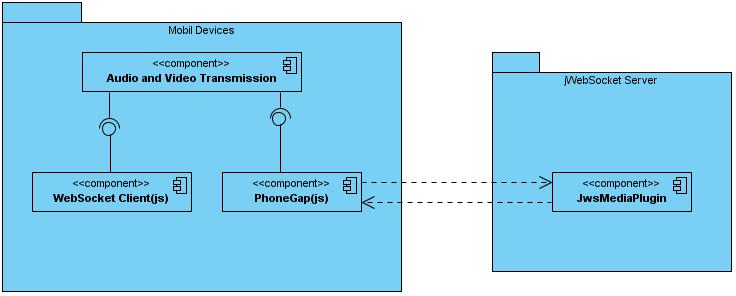
The Project has as main objetive increase the jWebSocket potentialities, developing new functionalities that allows to create web applications in real time with the beneficts that offers the Streaming technology. For these purposes there are 2 applications to work with the basics functionalities of this technology. An stationary applications using Java Media Framework and a native application using PhoneGap.

# 2. Solution Infrastructure.

****

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

Java Media Framework is used as API to handle the capture process, low level encode and sends the packets to the server.



**Figure2: Structure for the native application**

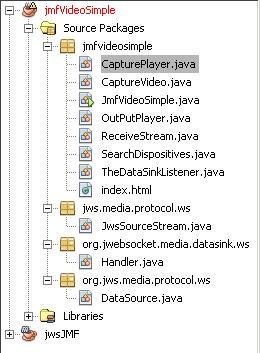
PhoneGap framework is the point Access to the differents SmartPhone controllers allowing the capture and the processing for the image, audio and video. It can be integrated with other libraries that work with similar devices.

The use of jWebSocket server to potential the streaming technology provides to the users create applications with high levels of security in the transmition process, allowing a confiability in the service in enviorments where needs a lot of users traffic and high speed in the communication. Also it demonstrate the high prestations that offers WebSocket protocol for this purposes.

# 3. Modularization of the extention

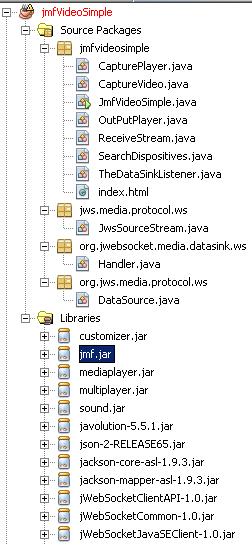
# 3.1 Stationary Application

The main player where execute the capture methods is inside the jmfVideoSimple packet, in the figure 3 it can appreciate the packets structure.



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

The jmfvideosimple has the main classes that starts the capture process of the audio and video signal where firstly make a search to find de controllers installed.

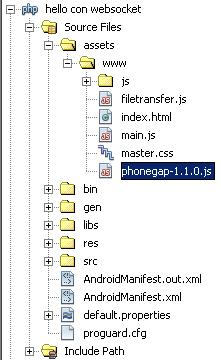


**Figura4: Structure of the java packets for the stationary application with the JMF libraries.**

As shows in the figure 4 , it appears the Java Media Framework libraries, being jmf.jar the most important packets.

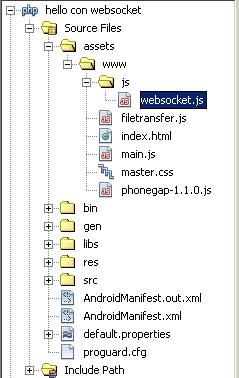
**3.1 Aplicación Nativa**

The main file that must to be include it’s the PhoneGap framework to can interact with the Smartphone controllers.



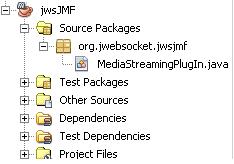
**Figure5: Structure of the native application with the PhoneGap library.**

Other importat file to be include is the jWebSocket client to JavaScript to establish the communication with the server. Its shows in the 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.

**

**Figura7: Structure for the server.**

# 4. Structure of source code

**4.1 Maven configuration**

El módulo **jwsJMF** se configura a través del fichero pom.xml, una representación XML del proyecto de Maven dentro de la carpeta del proyecto que permite configurar las dependencias y propiedades básicas del proyecto.

The jwsJMF module configuration is located in the file pom.xml, an XML representation for the project is inside the folder that allows 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**

**4.2.1 Packets structure for the stationary application**

The **jmfvideosimple** packetis composed for 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 information captured. |
| CaptureVideo | Capture the Medialocators and create the DataSources to send it to the player |
| OutPutPlayer | Shows in the other client the information transmitted. |
| ReceiveStream | Received the data flow send from the server and create the output DataSource to the player. |
| SearchDispositives | Search the audio and video controllers installed in the pc |
| TheDataSinkListener | Controls the DataSink states |

The **jws.media.protocol.ws** packet is composed by the following library:

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

The **org.jwebsocket.media.datasink.ws** packet is composed by the following library:

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

The **org.jws.media.protocol.ws** packet is composed by the following library:

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

**4.2.2 Packets structure for native application**

The **www** packet packet is composed by the following libraries:

|  |  |
| --- | --- |
| Library | Description |
| Websocket.js | jWebSocket client to establish the communication with the server. |
| Filetransfer.js | Send 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 | Se realizan los distintos estilos de la aplicación  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:

CapturePlayer

|  |  |  |
| --- | --- | --- |
| Type | Method | Description |
| void | [**init**](file:///D:\TESIS\sourceCode\ijwssvn\branches\jWebSocket-1.0\javadocs\jWebSocketServer\apidocs\org\jwebsocket\grizzly\GrizzlyConnector.html#generateUID())() | Initialize the application in general, thus as the connection to the jWebSocket server. |
| void | **processToken(WebSocketClientEvent wsce, Token token)** | Listener that analize and process the token that is capturing |
| void | **processOpening(WebSocketClientEvent wsce)** | Listener that excecute 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)** | Process 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 will process in the player. |
| MediaLocator | **getVideoLocator** () | Search in the video controllers and capture the first available for the webcam |
| MediaLocator | **getAudioLocator** () | Search in the audio controllers and capture 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 transmition 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())() | Start the application in general, verify 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 Process to the player. |
|  |  |  |
|  |  |  |

SearchDispositives

|  |  |  |
| --- | --- | --- |
| Type | Method | Description |
| void | **Scaner**() | Search 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 transmition. |
| void | **waitEndOfStream(long checkTimeMs)** | Controls the threads of the different transmition 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 transmition. |

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 transmition |
| void | **open()** | Open the conecction and verify if the state has been initialized |
| void | **start()** | Begins the transmition 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 transmition 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 transmition. |
| 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 |
| MediaLocator | **getLocator**() | Returns the medialocator that belong to parent element |
| void | **initCheck()** | Ckeck if has some MediaLocator from the server |
| void | **connect()** | Create the jWebSocket to connect and capture the streaming the server via token |
| void | **processToken(WebSocketClientEvent wsce, Token aToken)** | Receive the token with the data flow codify. |
| void | **processOpening(WebSocketClientEvent wsce)** | Listener that excecute when the conecction Process is opening |
| void | **processOpened(WebSocketClientEvent wsce)** | Fired when the conection Process is opened |
| void | **processPacket(WebSocketClientEvent wsce, WebSocketPacket wsp)** | Process the incoming data packets . |
| void | **processClosed(WebSocketClientEvent wsce)** | Fires when the connection is stoped. |
| 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 | Descripción |
| void | **init()** | This method is called when start the application and verify that the móbile device is ready. |
| void | **openWebSocket**() | Realize the connection to the server and control the states. |
| void | **initApp()** | Its execute when the connection is opened |
| void | **captureImage()** | Access to the capture image controller from the movil device and controls the success and failure callbacks |
| void | **captureVideogetPicture()** | Search in the stored files(in this case videos) to send it to the server. |
| void | **captureVideo(**) | Access to the smartphone video controllers and controls the success and failure callbacks. |
| void | **captureAudio**() | Access to the smartphone audio controllers and controls the success and failure callbacks. |
| void | **uploadFile(aimageData)** | Encoded in base64 the data to send to the server |
| void | **uploadAudioFile**(amediaFile) | Establish the parameters of the audio file captured to send to the server. |
| void | **uploadVideoFile**(amediaFile) | Establish the parameters of the audio file captured to 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. Conclusiones

Después de haber leído este manual el desarrollador se debe haber familiarizado con la estructura de las aplicaciones que hacen uso de la tecnología streaming utilizando el marco de trabajo jWebSocket, sabiendo los detalles de implementación principales, además, tendrá un conocimiento avanzado de las dependencias de las mismas, así como los ficheros de necesarios para configurar el mismo.

El hecho de que jWebSocket hoy sea utilizado como marco de trabajo para el empleo de la tecnología streaming hace que sea mucho más extensible y confiable.