# Manual de desarrollo

**jWebSocket**

**Librería cliente C#**

**Versión 1.0**

# Control de versiones

|  |  |  |  |
| --- | --- | --- | --- |
| **Fecha** | **Versión** | **Descripción** | **Autor** |
| 13/05/12 | 1.0 | Creación del documento | Rolando Betancourt Toucet |

## Visión general

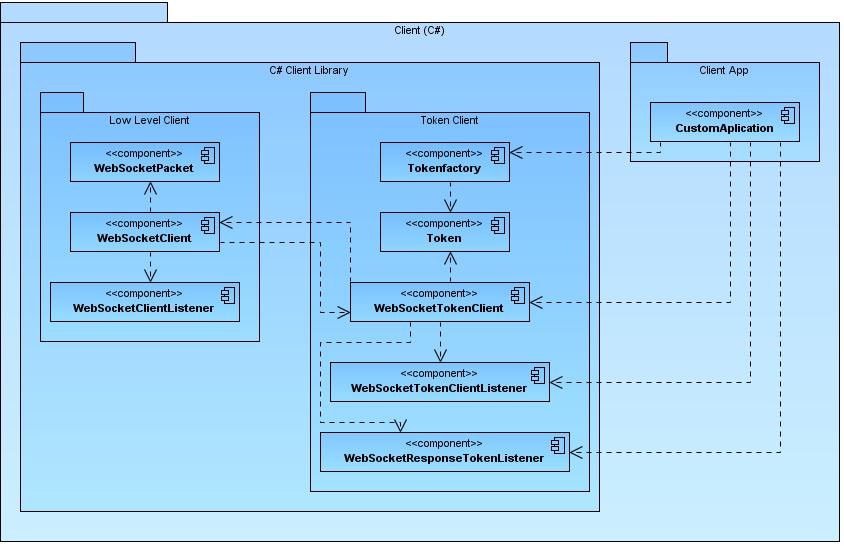
La Librería Cliente C# no es más que una implementación del protocolo WebSocket en el lenguaje de programación C# que brinda una API bien definida mediante la cual se puede lograr la integración entre las aplicaciones clientes desarrolladas con este lenguaje y el marco de trabajo jWebSocket. La utilización de dicha librería les permite a los desarrolladores centrarse completamente en la lógica de la aplicación así como un ahorro considerable de tiempo en la implementación de las mismas.

## Infraestructura, Modelo, Enfoque

En el desarrollo de esta librería se siguió una arquitectura en capas la cual brinda la posibilidad de dividir la implementación en dos capas principales. Una capa de bajo nivel la cual se encarga de realizar la conexión y transferencia de datos con cualquier servidor que utilice el protocolo WebSocket, y otra capa de alto nivel que cumple con las especificaciones del servidor jWebSocket para lograr la comunicación con el mismo.

Con el objetivo de reutilizar la mayor cantidad de código posible se utilizó un modelo de componentes para desarrollar las distintas funcionalidades de la librería. Esto trae consigo una mayor organización e independencia del código.

La siguiente imagen muestra un diagrama de componentes para ayudar a entender mejor dicha solución.

**

**Ilustración 1:** Diagrama de componentes de la librería cliente C#.

**Patrón de Diseño**

Debido a que esta librería se encarga de enviar y recibir información constantemente se hace conveniente la utilización del patrón *Observador* para lograr una mayor fluidez y notificación de los eventos. Este patrón resuelve la necesidad de mantener la consistencia entre objetos relacionados sin requerir que las clases se encuentren altamente acopladas.

**Tecnologías Utilizadas**

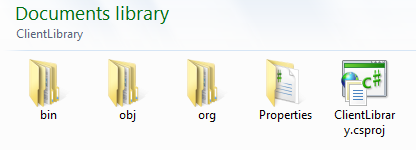
Para su desarrollo se utilizó el lenguaje de programación C# con .NET Framework 4.0 y el entorno de desarrollo integrado (IDE) Visual Studio 2010.

## Requisitos y Prerrequisitos

No son necesarios

## Módulos, Estructura

|  |  |
| --- | --- |
| **Nombre del proyecto** | Librería Cliente C# |
| **Localizacion de las fuentes en el servidor** | [*https://jwsdev.org:9443/svn/jWebSocket/branches/jWebSocket-1.0/jWebSocketCSClient*](https://jwsdev.org:9443/svn/jWebSocket/branches/jWebSocket-1.0/jWebSocketCSClient) |
| **SVN branch** | jWebSocket-1.0 |
| **Archivo .zip** | SourceCodeCSharpClientLibrary.zip |
| **Estructura de directorios** |  |
| ***org.jwebsocket.client.common:***  Contiene todas las implementaciones que son utilizadas por el cliente a bajo nivel y el cliente Token. | |
| ***org.jwebsocket.client.csharp.api:***  Contiene todas las interfaces que son implementadas por el cliente de bajo nivel. | |
| ***org.jwebsocket.client.csharp.cbase:***  Contiene la implementacion de las clases base para el cliente de bajo nivel. | |
| ***org.jwebsocket.client.csharp.kit:***  Contiene la implementacion de las herramientas utilizadas por el cliente de bajo nivel. | |
| ***org.jwebsocket.client.token.api:***  Contiene todas las interfaces que son implementadas por el cliente de Token. | |
| ***org.jwebsocket.client.token.kit:***  Contiene la implementacion de las herramientas utilizadas por el cliente de Token. | |
| ***org.jwebsocket.client.token.processor:***  Contiene la implementacion referente al procesamiento de un Token. | |
| ***org.jwebsocket.client.token.tbase:***  Contiene la implementacion de las clases base para el cliente de Token. | |

****

**bin:** Este directorio almacena el código fuente compilado.

**obj:** Este directorio almacena temporalmente el código fuente compilado, su contenido no se incluye en el control de versiones.

**org:** Este directorio contiene todo el código fuente de las clases y librerías de la solución.

**Properties:** Este directorio contiene los datos dl ensamblado de la solución, su contenido no se incluye en el control de versiones.

## Código fuente

|  |  |  |
| --- | --- | --- |
| **Paquete** | Low Level Client | |
| **Componente** | WebSocketClient | |
| **Clase** | WebSocketBaseClient.cs | |
| **Extiende** | WebSocketClient.cs | |
| **Descripción** | Esta clase implementa el protocolo WebSocket y hace función de cliente a bajo nivel. | |
| **Dependencias** | using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Net.Sockets;  using System.Net.Security;  using System.Security.Authentication;  using System.Security.Cryptography.X509Certificates;  using System.IO;  using System.Threading;  using System.Diagnostics;  using log4net;  using log4net.Config;  using ClientLibrary.org.jwebsocket.client.csharp.api;  using ClientLibrary.org.jwebsocket.client.csharp.kit;  using ClientLibrary.org.jwebsocket.client.common; | |
| **Atributos** | | |
| mURI | Uri**:** Objeto que almacena la Uri del servidor que se desea conectar. | |
| mSocket | TcpClient: Objeto que brinda una API para realizar la conexión a una aplicación servidor a través de un canal TCP. | |
| mNetStream | NetworkStream: Objeto que permite realizar una conexión a través de un socket utilizando TcpClient. | |
| mVersion | Int: Almacena la versión del protocolo WebSocket que se utilizara por defecto en la conexión con el servidor. | |
| mListeners | List<WebSocketClientListener>: Contiene un listado de los escuchadores de la aplicación. | |
| mSubProtocols | List<WebSocketSubProtocol>: Contiene un listado de los subprotocolos utilizados para establecer la conexión con el servidor. | |
| mNegotiatedSubProtocol | WebSocketSubProtocol: Este objeto representa el protocolo que se negocia con el servidor para establecer la conexión. | |
| mEncoding | WebSocketEncoding: | |
| mStatus | WebSocketStatus: Representa el estado en que se encuentra la conexión con el servidor. | |
| mReliabilityOptions | WebSocketReliabilityOptions: | |
| mHeaders | WebSocketHeaders: | |
| mCLose | WebSocketCloseReason | |
| mIsRunning | Bool: Toma valor true cuando la aplicación se encuentra ejecutándose y false en caso contrario. | |
| **Métodos** | | |
| public WebSocketBaseClient() | | Constructor de la clase |
| public void Open(string aURI) | | Establece la conexión con el servidor especificando su dirección. |
| public void SendText(string aUTF8String) | | Envía un token de texto hacia el servidor |
| public void SendBinary(byte[] aBinaryData) | | Envía un token binario hacia el servidor |
| public void SendText(string aUTF8String, int aFragmentSize) | | Envía un fragmento de token de texto hacia el servidor |
| public void SendBinary(byte[] aBinaryData, int aFragmentSize) | | Envía un fragmento de token binario hacia el servidor |
| public void Close() | | Finaliza la conexión con el servidor |
| public bool IsRunning() | | Determina si la aplicación está en ejecución |
| public Dictionary<string, string> GetRequestHeader() | | Retorna la cabecera de la solicitud realizada al servidor |
| public Dictionary<string, string> GetResponseHeader() | | Retorna la cabecera de la respuesta del servidor |
| public void AddListener(WebSocketClientListener aListener) | |  |
| public void RemoveListener(WebSocketClientListener aListener) | |  |
| public void OnOpen(WebSocketHeaders aHeader) | |  |
| public void OnClose(WebSocketCloseReason aCloseReason) | |  |
| public void OnError(WebSocketError aError) | |  |
| public virtual void OnTextMessage(WebSocketPacket aDataPacket) | |  |
| public void OnBinaryMessage(WebSocketPacket aDataPacket)  public void OnPing() | |  |

|  |  |  |
| --- | --- | --- |
| **Paquete** | Low Level Client | |
| **Componente** | WebSocketClientListener | |
| **Clase** | WebSocketClientListener.cs | |
| **Descripción** |  | |
| **Dependencias** | using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using ClientLibrary.org.jwebsocket.client.csharp.kit; | |
| **Métodos** | | |
| void ProcessOnTextMessage(WebSocketPacket aDataPacket) | |  |
| void ProcessOnBinaryMessage(WebSocketPacket aDataPacket) | |  |
| void ProcessOnFragment(WebSocketPacket aFragment, int aIndex, int aTotal) | |  |
| void ProcessOnOpen(WebSocketHeaders aHeader) | |  |
| void ProcessOnClose(WebSocketCloseReason aCloseReason) | |  |
| void ProcessOnError(WebSocketError aError) | |  |
| void ProcessOnPing() | |  |
| void ProcessOnPong() | |  |

|  |  |  |
| --- | --- | --- |
| **Paquete** | Low Level Client | |
| **Componente** | WebSocketPacket | |
| **Clase** | WebSocketRawPacket.cs | |
| **Extiende** | WebSocketPacket.cs | |
| **Descripción** |  | |
| **Dependencias** | using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using ClientLibrary.org.jwebsocket.client.csharp.api;  using ClientLibrary.org.jwebsocket.client.csharp.kit;  using ClientLibrary.org.jwebsocket.client.common; | |
| **Atributos** | | |
| mByteArray | byte[]: | |
| mFragments | string[]: | |
| mFragmentsLoaded | Int: | |
| mFragmentsExpected | Int: | |
| mIsFragmented | Bool: | |
| mIsComplete | Bool: | |
| mCreationDate | DateTime: | |
| mTimeout | Long: | |
| mFrameType | WebSocketFrameType: | |
| **Métodos** | | |
| public WebSocketRawPacket(int aInitialSize) | |  |
| public WebSocketRawPacket(byte[] aByteArray) | |  |
| public WebSocketRawPacket(WebSocketFrameType aFrameType, byte[] aByteArray) | |  |
| public WebSocketRawPacket(string aString) | |  |
| public WebSocketRawPacket(WebSocketFrameType aFrameType, string aString) | |  |
| public WebSocketRawPacket(string aString, WebSocketTypeEncoding aEncoding) | |  |
| public void InitFragmented(int aTotal) | |  |
| public void SetFragment(string aString, int aIdx) | |  |
| public void SetString(string aString) | |  |
| public void SetString(string aString, WebSocketTypeEncoding aEncoding) | |  |
| public void SetUTF8(string aString) | |  |
| public void SetASCII(string aString) | |  |
| public string GetString() | |  |
| public string GetString(WebSocketTypeEncoding aEncoding) | |  |
| public string GetUTF8() | |  |
| public string GetASCII() | |  |

|  |  |  |
| --- | --- | --- |
| **Paquete** | Token Client | |
| **Componente** | WebSocketTokenClient | |
| **Clase** | WebSocketBaseTokenClient.cs | |
| **Extiende** | WebSocketBaseClient.cs | |
| **Descripción** |  | |
| **Dependencias** | using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using ClientLibrary.org.jwebsocket.client.token.kit;  using ClientLibrary.org.jwebsocket.client.csharp.csbase;  using ClientLibrary.org.jwebsocket.client.csharp.kit;  using ClientLibrary.org.jwebsocket.client.csharp.api;  using ClientLibrary.org.jwebsocket.client.token.api;  using ClientLibrary.org.jwebsocket.client.common;  using ClientLibrary.org.jwebsocket.client.token.processor; | |
| **Atributos** | | |
| CURRENT\_TOKEN\_ID | Int: | |
| mPendingResponseQueue | Dictionary<int, PendingResponseQueueItem>: | |
| **Métodos** | | |
| public WebSocketBaseTokenClient() | |  |
| public WebSocketBaseTokenClient(WebSocketReliabilityOptions aReliabilityOptions)  public void SendTokenText(Token aToken) | |  |
| public void SendTokenText(Token aToken, WebSocketResponseTokenListener aResponseListener) | |  |
| private void SendTokenBinary(Token aToken) | |  |
| private void SendTokenBinary(Token aToken, WebSocketResponseTokenListener aResponseListener) | |  |
| public void AddTokenClientListener(WebSocketClientTokenListener aTokenListener) | |  |
| public void RemoveTokenClientListener(WebSocketClientTokenListener aTokenListener) | |  |

|  |  |  |
| --- | --- | --- |
| **Paquete** | Token Client | |
| **Componente** | WebSocketTokenClientListener | |
| **Clase** | WebSocketClientTokenListener.cs | |
| **Extiende** | WebSocketClientListener.cs | |
| **Descripción** |  | |
| **Dependencias** | using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using ClientLibrary.org.jwebsocket.client.csharp.api; | |
| **Métodos** | | |
| void ProcessOnTokenText(Token aToken) | |  |

|  |  |  |
| --- | --- | --- |
| **Paquete** | Token Client | |
| **Componente** | WebSocketResponseTokenListener | |
| **Clase** | WebSocketResponseTokenListener.cs | |
| **Descripción** |  | |
| **Dependencias** | using System;  using System.Collections.Generic;  using System.Linq;  using System.Text; | |
| **Métodos** | | |
| void OnTimeout(Token aToken) | |  |
| void OnResponse(Token aToken) | |  |
| void OnSuccess(Token aToken) | |  |
| void OnFailure(Token aToken) | |  |

|  |  |  |
| --- | --- | --- |
| **Paquete** | Token Client | |
| **Componente** | Token | |
| **Clase** | DictionaryToken.cs | |
| **Extiende** | Token.cs | |
| **Descripción** |  | |
| **Dependencias** | using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using ClientLibrary.org.jwebsocket.client.token.api;  using ClientLibrary.org.jwebsocket.client.common; | |
| **Atributos** | | |
| mData | Dictionary<string, object>: | |
| mBinary | Bool: | |
| **Métodos** | | |
| public DictionaryToken() | |  |
| public DictionaryToken(string aType) | |  |
| public DictionaryToken(Dictionary<string, object> aDictionary) | |  |
| public DictionaryToken(string aNS, string aType) | |  |
| public void SetDictionary(Dictionary<string, object> aDictionary) | |  |
| public object GetObject(string aKey) | |  |
| public string GetString(string aKey) | |  |
| public void SetString(string aKey, string aValue) | |  |
| public int GetInt(string aKey) | |  |
| public void SetInt(string aKey, int aValue) | |  |
| public double GetDouble(string aKey) | |  |
| public void SetDouble(string aKey, double aValue) | |  |
| public bool GetBool(string aKey) | |  |
| public void SetBool(string aKey, bool aValue) | |  |
| public List<object> GetList(string aKey) | |  |
| public void SetList(string aKey, List<object> aList) | |  |
| public void SetToken(string aKey, Token aToken) | |  |
| public Token GetToken(string aKey) | |  |
| public Dictionary<string, object> GetDictionary() | |  |
| public Dictionary<string, object> GetDictionary(string aKey) | |  |
| public void SetDictionary(string aKey, Dictionary<string, object> aDictionary) | |  |
| public string GetType() | |  |
| public void SetType(string aType) | |  |
| public string GetNS() | |  |
| public void SetNS(string aNS) | |  |
| public void Clear() | |  |
| public void Remove(string aKey) | |  |

|  |  |  |
| --- | --- | --- |
| **Paquete** | Token Client | |
| **Componente** | Tokenfactory | |
| **Clase** | TokenFactory.cs | |
| **Descripción** |  | |
| **Dependencias** | using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using ClientLibrary.org.jwebsocket.client.token.api;  using ClientLibrary.org.jwebsocket.client.common;  using ClientLibrary.org.jwebsocket.client.csharp.api;  using ClientLibrary.org.jwebsocket.client.token.processor; | |
| **Métodos** | | |
| public static Token CreateToken() | |  |
| public static Token CreateToken(string aType) | |  |
| public static Token CreateToken(string aNS, string aType) | |  |
| public static Token PacketToToken(string aFormat, WebSocketPacket aDataPacket) | |  |
| public static WebSocketPacket TokenToPacket(string aFormat, Token aToken) | |  |

**Reusabilidad**

La solución desarrollada está estructurada fundamentalmente en 2 capas bien definidas e independientes. Una de ellas es La capa de bajo nivel, que se encarga de gestionar la comunicación entre el cliente y el servidor utilizando el protocolo WebSocket .Esta capa es completamente reutilizable por cualquier otra aplicación que desee establecer conexión mediante el protocolo WebSocket.

**El patrón Observador**

El patrón observador define dependencias de uno a muchos entre objetos, así que cuando un objeto cambia su estado, todos sus escuchadores son notificados automáticamente. Este patrón resuelve la necesidad de mantener la consistencia entre objetos relacionados sin requerir que las clases se encuentren altamente acopladas.

Con la utilización del patrón Observador se obtienen implementaciones fáciles de leer, alta flexibilidad y mínimo acoplamiento entre las clases de un sistema basado en POO.

La implementación del patrón Observador en la Librería Cliente C# se rige por las siguientes interfaces:

**Interfaz WebSocketClientpara los objetos escuchables (sujetos):**

public interface WebSocketClient

{

void Open(string aURL);

void Open(string aURL, string aSubProtocol);

void Open(string aURL, string aSubProtocol, int aTimeout);

void SendText(string aUTF8String);

void SendText(string aUTF8String, int aFragmentSize);

void SendBinary(byte[] aBinaryData);

void SendBinary(byte[] aBinaryData, int aFragmentSize);

void Ping();

void OnTextMessage(WebSocketPacket aDataPacket);

void OnBinaryMessage(WebSocketPacket aDataPacket);

void OnFragment(WebSocketPacket aFragment, int aIndex, int aTotal);

void OnOpen(WebSocketHeaders aHeader);

void OnClose(WebSocketCloseReason aCloseReason);

void OnError(WebSocketError aError);

void OnPing();

void OnPong();

void AddListener(WebSocketClientListener aListener);

void RemoveListener(WebSocketClientListener aListener);

}

**Interfaz WebSocketClientListener para los objetos que desean escuchar eventos de otros objetos (escuchadores)**

public interface WebSocketClientListener

{

void ProcessOnTextMessage(WebSocketPacket aDataPacket);

void ProcessOnBinaryMessage(WebSocketPacket aDataPacket);

void ProcessOnFragment(WebSocketPacket aFragment, int aIndex, int aTotal);

void ProcessOnOpen(WebSocketHeaders aHeader);

void ProcessOnClose(WebSocketCloseReason aCloseReason);

void ProcessOnError(WebSocketError aError);

void ProcessOnPing();

void ProcessOnPong();

}

**Interfaz WebSocketClientTokenListener** **para los objetos que desean escuchar eventos de otros objetos (escuchadores)**

public interface WebSocketClientTokenListener : WebSocketClientListener

{

void ProcessOnTokenText(Token aToken);

}

**Interfaz WebSocketResponseTokenListener para los objetos que desean escuchar eventos de otros objetos (escuchadores)**

public interface WebSocketResponseTokenListener

{

void OnTimeout(Token aToken);

void OnResponse(Token aToken);

void OnSuccess(Token aToken);

void OnFailure(Token aToken);

}

**Ejemplo de utilización del patrón Observador en la Librería Cliente C#:**

|  |
| --- |
| class Program  {  static void Main(string[] args)  {  WebSocketBaseTokenClient mClient = new WebSocketBaseTokenClient(  new WebSocketReliabilityOptions(true, 3000, 3000) );  mClient.AddListener(new MyListener(mClient));  mClient.Open("ws://localhost:8787//jWebSocket//jWebSocket");  }  }  public class MyListener : WebSocketClientTokenListener  {  private WebSocketBaseTokenClient mClient;  public MyListener(WebSocketBaseTokenClient aClient)  {  this.mClient = aClient;  }  public void ProcessOnTokenText(Token aToken)  {  Console.WriteLine("Token Text: " + aToken.GetString("data"));  }  public void ProcessOnBinaryMessage(WebSocketPacket aDataPacket) { }  public void ProcessOnClose(WebSocketCloseReason aCloseReason)  {  Console.WriteLine(aCloseReason.ToString());  }  public void ProcessOnError(WebSocketError aError)  {  Console.WriteLine(aError.Reason);  }  public void ProcessOnFragment(WebSocketPacket aFragment, int aIndex,  int aTotal) { }  public void ProcessOnOpen(WebSocketHeaders aHeader)  {  Token lMyToken = TokenFactory.CreateToken(  WebSocketMessage.NS\_SYSTEM\_PLUGIN, WebSocketMessage.ECHO);  lMyToken.SetString("data", "tito");  mClient.SendTokenText(lMyToken, new MyResponse());  }  public void ProcessOnPing() { }  public void ProcessOnPong() { }  public void ProcessOnTextMessage(WebSocketPacket aDataPacket) { }  }  public class MyResponse : WebSocketResponseTokenListener  {  public void OnFailure(Token aToken)  {  Console.WriteLine("Failure");  }  public void OnResponse(Token aToken) { }  public void OnSuccess(Token aToken)  {  Console.WriteLine("Success");  }  public void OnTimeout(Token aToken) { }  } |

## Interfaces

La solución no contiene interfaces.

## Marcos de trabajo, librerías y herramientas

**Marco de trabajo:**

*.NET Framework 4.0:* Contiene todas las librerías necesarias para desarrollar y ejecutar las aplicaciones basadas en esta tecnología.

**Librerías:**

*JSON bajo la licencia GPL:* Esta librería se utilizó para serializar y desrealizar los datos que se intercambian con el servidor a través del protocolo WebSocket.

*Log4net bajo la licencia de apache:* Esta librería se utilizó para mostrar todos los logs de la solución desarrollada.

## Base de datos y persistencia de datos

Esta solución no contiene persistencia de datos.

## Hardware

No se necesita ningún hardware específico para el desarrollo de esta solución.

## Seguridad

Se utiliza SSL para la transferencia segura.

## Pruebas, aseguramiento de la calidad e Integración Continúa

*[En este epígrafe se describe la ejecución de pruebas automatizadas y manuales para el aseguramiento de la calidad. Se incluirán también los elementos de integración continúa cuando estén definidos por el proyecto. Se mencionan a continuación los elementos a tener en cuenta para la descripción de este epígrafe:*

* *Describir los casos de pruebas, funcionales (JUnit, Jasmine) y de interfaz de usuario (iMacros, pendiente)*
* *Qué pruebas automatizadas están disponibles, cómo son estas invocadas, cómo están integradas estas pruebas dentro de la suite de pruebas de desarrollo de jWebSocket. Dónde se pueden encontrar informes o reportes al respecto.*
* *Diga si existen algunos problemas identificados durante el proceso de pruebas, cuáles son las razones de estos errores, están solucionados los mismos o qué soluciones alternativas se pueden hacer ante estos casos.*
* *Describa el manejo de los errores antes situaciones potenciales cómo, la base de datos no está disponible, la conexión de red falló, hay carencia de algún recurso necesario, entre otros casos.*
* *Cómo se manejan los errores de manera general y el tratamiento de excepciones.*
* *En caso de realizar pruebas manuales, especifique si estas puede realizarse antes de ejecutar la solución, especificar los detalles necesarios.*

## Mejora continúa

*No se proponen métodos ni estrategias para la mejora continua de la solución.*

## Referencias

|  |
| --- |
| <?xml version="1.0"?>  <doc>  <assembly>  <name>ClientLibrary</name>  </assembly>  <members>  <member name="T:ClientLibrary.org.jwebsocket.client.token.kit.WebSocketTokenListener">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>3/26/2012</lastUpdate>  <summary>    </summary>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.token.api.WebSocketClientTokenListener">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>3/26/2012</lastUpdate>  <summary>    </summary>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketClientListener">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>3/26/2012</lastUpdate>  <summary>  Interface for the low level WebSocket listeners.  </summary>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketClientListener.ProcessOnTextMessage(ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketPacket)">  <summary>  Invoked when a data packet as text message from a client is received.  </summary>  <param name="aDataPacket">Data packet received.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketClientListener.ProcessOnBinaryMessage(ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketPacket)">  <summary>  Invoked when a data packet as binary message from a client is received.  </summary>  <param name="aDataPacket">Data packet received.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketClientListener.ProcessOnFragment(ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketPacket,System.Int32,System.Int32)">  <summary>  Invoked when a fragment from client is received.  </summary>  <param name="aFragment">Data packet fragment received.</param>  <param name="aIndex">Index of fragment.</param>  <param name="aTotal">Total size of fragment.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketClientListener.ProcessOnOpen(ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketHeaders)">  <summary>  Invoked when a new client connects to the Client.  </summary>  <param name="aHeader">Header fields from the handshake.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketClientListener.ProcessOnClose(ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketCloseReason)">  <summary>  Invoked when a client was disconnted to the Client.  </summary>  <param name="aCloseReason">Represent the reason of the disconnect.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketClientListener.ProcessOnError(ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketError)">  <summary>  Invoked when any error occurs.  </summary>  <param name="aError">Description of the error.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketClientListener.ProcessOnPing">  <summary>  Invoked when client sent a ping.  </summary>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketClientListener.ProcessOnPong">  <summary>  Invoked when server sent a pong.  </summary>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketClient">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>3/26/2012</lastUpdate>  <summary>  API for low level client  </summary>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketClient.Open(System.String)">  <summary>  Establish a connection to a websocket server.  </summary>  <param name="aURL">Server URL.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketClient.Open(System.String,System.String)">  <summary>  Establish a connection to a websocket server.  </summary>  <param name="aURL">Server URL.</param>  <param name="aSubProtocol">WebSocket protocol specification.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketClient.Open(System.String,System.String,System.Int32)">  <summary>  Establish a connection to a websocket server.  </summary>  <param name="aURL">Server URL.</param>  <param name="aSubProtocol">WebSocket protocol specification.</param>  <param name="aTimeout">Timeout for close the conection.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketClient.SendText(System.String)">  <summary>  Send a complete packet as UTF8 string.  </summary>  <param name="aUTF8String">Packet as UTF8 String.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketClient.SendText(System.String,System.Int32)">  <summary>  Send packet UTF8 string as multiple fragments.  </summary>  <param name="aUTF8String">Packet as UTF8 String.</param>  <param name="aFragmentSize">Maximum fragment size.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketClient.SendBinary(System.Byte[])">  <summary>  Send a complete packet as binary data.  </summary>  <param name="aBinaryData">Binary data.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketClient.SendBinary(System.Byte[],System.Int32)">  <summary>  Send packet binary Data as multiple fragments.  </summary>  <param name="aBinaryData">Binary data.</param>  <param name="aFragmentSize">Maximum fragment size.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketClient.Ping">  <summary>  Send a ping frame to the server and starts a timeout observer.  </summary>  <param name="aTimeout">Timeout for close the conection.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketClient.OnTextMessage(ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketPacket)">  <summary>  Called when a text message has been received.  </summary>  <param name="aDataPacket">Data packet received .</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketClient.OnBinaryMessage(ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketPacket)">  <summary>  Called when a binary message has been received.  </summary>  <param name="aDataPacket">Data packet received.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketClient.OnFragment(ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketPacket,System.Int32,System.Int32)">  <summary>  Called when a fragment has been received.  </summary>  <param name="aFragment">Data packet fragment received.</param>  <param name="aIndex">Index of fragment.</param>  <param name="aTotal">Total size of fragment.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketClient.OnOpen(ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketHeaders)">  <summary>  Called when connection to a websocket server has been Establish.  </summary>  <param name="aHeader">Header fields from the handshake.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketClient.OnClose(ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketCloseReason)">  <summary>  Called when connection to a websocket server has been closed.  </summary>  <param name="aCloseReason">Represent the reason of the disconnect.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketClient.OnError(ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketError)">  <summary>  Called when occurs any error.  </summary>  <param name="aError">Represents the error occurred.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketClient.OnPing">  <summary>  Called when client sent a ping.  </summary>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketClient.OnPong">  <summary>  Called when server sent a pong.  </summary>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketClient.AddListener(ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketClientListener)">  <summary>  Adds the listener.  </summary>  <param name="aListener">A listener.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketClient.RemoveListener(ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketClientListener)">  <summary>  Removes the listener.  </summary>  <param name="aListener">A listener.</param>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketStatus">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>3/26/2012</lastUpdate>  <summary>  WebSocket Status.  </summary>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.common.WebSocketTypeEncoding">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>3/26/2012</lastUpdate>  <summary>  Type encoding  </summary>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.common.WebSocketConvert">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>3/26/2012</lastUpdate>  <summary>  Convert strings to bytes and viceversa.  </summary>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.common.WebSocketConvert.StringToBytes(System.String,ClientLibrary.org.jwebsocket.client.common.WebSocketTypeEncoding)">  <summary>  Strings to bytes.  </summary>  <param name="aString">String.</param>  <param name="aEncoding">Encoding.</param>  <returns>Array bytes</returns>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.common.WebSocketConvert.BytesToString(System.Byte[],ClientLibrary.org.jwebsocket.client.common.WebSocketTypeEncoding)">  <summary>  Bytes to strings.  </summary>  <param name="aBytes">Bytes.</param>  <param name="aEncoding">Encoding.</param>  <returns>String</returns>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.token.tbase.DictionaryToken">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>3/26/2012</lastUpdate>  <summary>    </summary>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.token.api.Token">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>3/26/2012</lastUpdate>  <summary>    </summary>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.token.api.Token.SetDictionary(System.Collections.Generic.Dictionary{System.String,System.Object})">  <summary>  Copies all fields from a Map into the Token. A check has to be made  by the corresponding implementations that only such data types are  passed that are supported by the Token abstraction.  </summary>  <param name="aDictionary">Dictionary.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.token.api.Token.GetObject(System.String)">  <summary>  Gets the object.  </summary>  <param name="aKey">key.</param>  <returns></returns>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.token.api.Token.GetString(System.String)">  <summary>  Gets the string.  </summary>  <param name="aKey">key.</param>  <returns></returns>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.token.api.Token.SetString(System.String,System.String)">  <summary>  Sets the string.  </summary>  <param name="aKey">key.</param>  <param name="aValue">Value.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.token.api.Token.GetInt(System.String)">  <summary>  Gets the int.  </summary>  <param name="aKey">key.</param>  <returns></returns>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.token.api.Token.SetInt(System.String,System.Int32)">  <summary>  Sets the int.  </summary>  <param name="aKey">key.</param>  <param name="aValue">Value.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.token.api.Token.GetDouble(System.String)">  <summary>  Gets the double.  </summary>  <param name="aKey">key.</param>  <returns></returns>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.token.api.Token.SetDouble(System.String,System.Double)">  <summary>  Sets the double.  </summary>  <param name="aKey">key.</param>  <param name="aValue">Value.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.token.api.Token.GetBool(System.String)">  <summary>  Gets the bool.  </summary>  <param name="aKey">key.</param>  <returns></returns>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.token.api.Token.SetBool(System.String,System.Boolean)">  <summary>  Sets the bool.  </summary>  <param name="aKey">key.</param>  <param name="aValue">if set to <c>true</c> [a value].</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.token.api.Token.GetList(System.String)">  <summary>  Gets the list.  </summary>  <param name="aKey">key.</param>  <returns>List.</returns>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.token.api.Token.SetList(System.String,System.Collections.Generic.List{System.Object})">  <summary>  Sets the list.  </summary>  <param name="aKey">key.</param>  <param name="aList">List.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.token.api.Token.SetToken(System.String,ClientLibrary.org.jwebsocket.client.token.api.Token)">  <summary>  Sets the token.  </summary>  <param name="aKey">key.</param>  <param name="aToken">Token.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.token.api.Token.GetToken(System.String)">  <summary>  Gets the token.  </summary>  <param name="aKey">key.</param>  <returns></returns>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.token.api.Token.GetDictionary">  <summary>  Gets the dictionary.  </summary>  <returns></returns>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.token.api.Token.GetDictionary(System.String)">  <summary>  Gets the dictionary.  </summary>  <param name="aKey">key.</param>  <returns>Dictionary.</returns>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.token.api.Token.SetDictionary(System.String,System.Collections.Generic.Dictionary{System.String,System.Object})">  <summary>  Sets the dictionary.  </summary>  <param name="aKey">key.</param>  <param name="aDictionary">Dictionary.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.token.api.Token.GetType">  <summary>  Gets the type.  </summary>  <returns>Type.</returns>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.token.api.Token.SetType(System.String)">  <summary>  Sets the type.  </summary>  <param name="aType">Type.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.token.api.Token.GetNS">  <summary>  Gets the NS.  </summary>  <returns>NS.</returns>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.token.api.Token.SetNS(System.String)">  <summary>  Sets the NS.  </summary>  <param name="aNS">NS.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.token.api.Token.Clear">  <summary>  Resets all fields of the token. After this operation the token is empty.  </summary>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.token.api.Token.Remove(System.String)">  <summary>  Removes the specified a key.  </summary>  <param name="aKey">key.</param>  </member>  <member name="P:ClientLibrary.org.jwebsocket.client.token.api.Token.IsBinary">  <summary>  Determines whether this instance is binary.  </summary>  <returns><c>true</c> if this instance is binary; otherwise, <c>false</c>.</returns>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketFrameType">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>3/26/2012</lastUpdate>  <summary>  Defines an frame type.  </summary>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketCloseReason">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>3/26/2012</lastUpdate>  <summary>  1000 indicates a normal closure, meaning that the purpose for  which the connection was established has been fulfilled.    1001 indicates that an endpoint is "going away", such as a server  going down or a browser having navigated away from a page.    1002 indicates that an endpoint is terminating the connection due  to a protocol error.    1004 Reserved. The specific meaning might be defined in the future.    1005 is a reserved value and MUST NOT be set as a status code in a  Close control frame by an endpoint. It is designated for use in  applications expecting a status code to indicate that no status  code was actually present.    1006 is a reserved value and MUST NOT be set as a status code in a  Close control frame by an endpoint. It is designated for use in  applications expecting a status code to indicate that the  connection was closed abnormally, e.g., without sending or  receiving a Close control frame.  </summary>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.csharp.csbase.WebSocketBaseClient">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>3/26/2012</lastUpdate>  <summary>  Base WebSocket implementation.  This uses thread model for handling WebSocket connection which is defined  by the WebSocket protocol specification.  http://www.whatwg.org/specs/web-socket-protocol/  http://www.w3.org/TR/websockets/  </summary>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.common.WebSocketMessage">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>3/26/2012</lastUpdate>  <summary>  All messages of the client.  </summary>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketPacket">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>3/26/2012</lastUpdate>  <summary>  Specifies the API for low level data packets which are interchanged between  client and server. Data packets do not have a special format at this  communication level.  </summary>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketPacket.InitFragmented(System.Int32)">  <summary>  Inits the fragmented.  </summary>  <param name="aTotal">Total size.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketPacket.SetFragment(System.String,System.Int32)">  <summary>  Sets the fragment.  </summary>  <param name="aString">String fragment.</param>  <param name="aIdx">Idx.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketPacket.PackFragments">  <summary>  Packs the fragments.  </summary>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketPacket.SetString(System.String)">  <summary>  Sets the value of the data packet to the given string by using  default encoding.  </summary>  <param name="aString">String value.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketPacket.SetString(System.String,ClientLibrary.org.jwebsocket.client.common.WebSocketTypeEncoding)">  <summary>  Sets the value of the data packet to the given string by using  the passed encoding.  </summary>  <param name="aString">String value.</param>  <param name="aEncoding">Encoding type.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketPacket.SetUTF8(System.String)">  <summary>  Sets the value of the data packet to the given string by using  UTF-8 encoding.  </summary>  <param name="aString">String value.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketPacket.SetASCII(System.String)">  <summary>  Sets the value of the data packet to the given string by using  7 bit US-ASCII encoding.  </summary>  <param name="aString">String value.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketPacket.GetString">  <summary>  Returns the content of the data packet as a string using default  encoding.  </summary>  <returns>Raw Data packet as string with default encoding</returns>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketPacket.GetString(ClientLibrary.org.jwebsocket.client.common.WebSocketTypeEncoding)">  <summary>  Returns the content of the data packet as a string using the passed  encoding.  </summary>  <param name="aEncoding">Encoding type.</param>  <returns>Raw Data packet as string using passed encoding</returns>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketPacket.GetUTF8">  <summary>  Interprets the data packet as a UTF8 string and returns the string  in UTF-8 encoding.If an exception occurs "null" is returned.  </summary>  <returns>Data packet as UTF-8 string or <c>null</c> if not convertible.</returns>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketPacket.GetASCII">  <summary>  Interprets the data packet as a US-ASCII string and returns the string  in US-ASCII encoding. If an exception occurs "null" is returned.  </summary>  <returns>Data packet as US-ASCII string or <c>null</c> if not convertible.</returns>  </member>  <member name="P:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketPacket.FrameType">  <summary>  Gets or sets data packet.  </summary>  <value>The type of the frame.</value>  </member>  <member name="P:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketPacket.ByteArray">  <summary>  Gets or sets the byte array.  </summary>  <value>The byte array.</value>  </member>  <member name="P:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketPacket.Fragments">  <summary>  Gets or sets the fragments.  </summary>  <value>The fragments.</value>  </member>  <member name="P:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketPacket.IsFragmented">  <summary>  Gets or sets a value indicating whether this instance is fragmented.  </summary>  <value><c>true</c> if this instance is fragmented; otherwise, <c>false</c>.</value>  </member>  <member name="P:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketPacket.IsComplete">  <summary>  Gets or sets a value indicating whether this instance is complete.  </summary>  <value><c>true</c> if this instance is complete; otherwise, <c>false</c>.</value>  </member>  <member name="P:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketPacket.CreationDate">  <summary>  Gets or sets the creation date.  </summary>  <value>The creation date.</value>  </member>  <member name="P:ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketPacket.Timeout">  <summary>  Gets or sets the timeout.  </summary>  <value>The timeout.</value>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketExceptionType">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>3/26/2012</lastUpdate>  <summary>  Defines an type of exception.  </summary>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.token.kit.PendingResponseQueueItem">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>3/26/2012</lastUpdate>  <summary>    </summary>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketRuntimeException">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>3/26/2012</lastUpdate>  <summary>  Define Runtime Exception for Websocket.  </summary>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketTimeout">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>3/26/2012</lastUpdate>  <summary>    </summary>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketTimeout.CallWithTimeout(System.Action{System.Net.Sockets.NetworkStream},System.Int32,System.Net.Sockets.NetworkStream)">  <summary>  Calls the with timeout.  </summary>  <param name="aAction">Action.</param>  <param name="aTimeoutMilliseconds">Timeout milliseconds.</param>  <param name="aIn">Network Stream.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketTimeout.CallWithTimeout(System.Action{ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketPacket},System.Int32,ClientLibrary.org.jwebsocket.client.csharp.api.WebSocketPacket)">  <summary>  Calls the with timeout.  </summary>  <param name="aAction">Action.</param>  <param name="aTimeoutMilliseconds">Timeout milliseconds.</param>  <param name="lPacket">WebSocket Packet.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketTimeout.CallWithTimeout(System.Action{ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketHeaders},System.Int32,ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketHeaders)">  <summary>  Calls the with timeout.  </summary>  <param name="aAction">Action.</param>  <param name="aTimeoutMilliseconds">Timeout milliseconds.</param>  <param name="lHeaders"> WebSocket Headers.</param>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketError">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>3/26/2012</lastUpdate>  <summary>  Defines an error  </summary>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketProtocolAbstraction">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>3/26/2012</lastUpdate>  <summary>  Utility class for packetizing WebSocketPacket into web socket protocol packet or packets (with fragmentation) and  vice versa.    Web socket protocol packet specification  http://tools.ietf.org/html/draft-ietf-hybi-thewebsocketprotocol-03)    0 1 2 3  0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1  +-+-+-+-+-------+-+-------------+-------------------------------------------+  |M|R|R|R| opcode|R| Payload len | Extended payload length |  |O |S|S|S | (4) |S| (7) | (16/63) |  |R |V|V|V | |V| | (if payload len==126/127) |  |E |1 |2|3 | |4| | |  +-+-+-+-+-------+-+-------------+ - - - - - - - - ---------------- - - - - - - +  | Extended payload length continued, if payload len == 127 |  + - - - - - - - - - - - - - - - +--------------------------------------------------+  | | Extension data |  +-------------------------------+ - - - - - - - - - - --------------------- - - - - +  : :  +---------------------------------------------------------------------------------+  : Application data :  +---------------------------------------------------------------------------------+  </summary>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketException">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>3/26/2012</lastUpdate>  <summary>  Defines an exception  </summary>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketStateOfStatus">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>3/26/2012</lastUpdate>  <summary>  Status of the connection.  </summary>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.token.api.WebSocketResponseTokenListener">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>3/26/2012</lastUpdate>  <summary>    </summary>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.token.api.WebSocketResponseTokenListener.OnTimeout(ClientLibrary.org.jwebsocket.client.token.api.Token)">  <summary>  Is fired when the given response timeout is exceeded.  </summary>  <param name="aToken"></param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.token.api.WebSocketResponseTokenListener.OnResponse(ClientLibrary.org.jwebsocket.client.token.api.Token)">  <summary>  Is fired on any response to a send token.  </summary>  <param name="aToken"></param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.token.api.WebSocketResponseTokenListener.OnSuccess(ClientLibrary.org.jwebsocket.client.token.api.Token)">  <summary>  Is fired if token.code equals 0 (zero).  </summary>  <param name="aToken"></param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.token.api.WebSocketResponseTokenListener.OnFailure(ClientLibrary.org.jwebsocket.client.token.api.Token)">  <summary>  Is fired if token.code does not equal 0 (zero).  </summary>  <param name="aToken"></param>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketHandshake">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>3/26/2012</lastUpdate>  <summary>  Utility class for all the handshake.  </summary>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketHandshake.GenerateC2SRequest">  <summary>  Generates the initial Handshake from a Client to the WebSocket.  </summary>  <returns>Handshake as byte array.</returns>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketHandshake.VerifyS2CResponse(ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketHeaders)">  <summary>  Verify that the server's response is correct.  </summary>  <param name="aHeaders">Header with the response data.</param>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.token.processor.JSONTokenProcessor">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>3/26/2012</lastUpdate>  <summary>    </summary>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.csharp.csbase.WebSocketRawPacket">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>3/26/2012</lastUpdate>  <summary>  Implements the low level data packets which are interchanged between  client and server. Data packets do not have a special format at this  communication level.  </summary>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.token.tbase.WebSocketBaseTokenClient">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>3/26/2012</lastUpdate>  <summary>  Token based implementation of <c>WebSocketBaseClient</c>.  </summary>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketSubProtocol">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>3/26/2012</lastUpdate>  <summary>  Define WebSocket sub protocol.  </summary>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>3/26/2012</lastUpdate>  <summary>  Contains all constants of the client.  </summary>  </member>  <member name="F:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants.mLog">  <summary>  Logger.  </summary>  </member>  <member name="F:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants.WS\_ENCODING\_DEFAULT">  <summary>  Use text format as default encoding for WebSocket Packets if not explicitly specified.  </summary>  </member>  <member name="F:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants.WS\_VERSION\_DEFAULT">  <summary>  WebSocket default protocol version.  </summary>  </member>  <member name="F:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants.WS\_SUBPROT\_PREFIX">  <summary>  jWebSocket sub protocol prefix.  </summary>  </member>  <member name="F:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants.WS\_SUBPROT\_JSON">  <summary>  jWebSocket JSON sub protocol.  </summary>  </member>  <member name="F:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants.WS\_SUBPROT\_CSV">  <summary>  jWebSocket CSV sub protocol.  </summary>  </member>  <member name="F:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants.WS\_SUBPROT\_XML">  <summary>  jWebSocket XML sub protocol.  </summary>  </member>  <member name="F:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants.WS\_SUBPROT\_TEXT">  <summary>  jWebSocket custom specific text sub protocol.  </summary>  </member>  <member name="F:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants.WS\_SUBPROT\_BINARY">  <summary>  jWebSocket custom specific binary sub protocol.  </summary>  </member>  <member name="F:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants.WS\_FORMAT\_JSON">  <summary>  JSON sub protocol format.  </summary>  </member>  <member name="F:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants.WS\_FORMAT\_CSV">  <summary>  CSV sub protocol format.  </summary>  </member>  <member name="F:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants.WS\_FORMAT\_XML">  <summary>  XML sub protocol format.  </summary>  </member>  <member name="F:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants.WS\_FORMAT\_BINARY">  <summary>  Binary sub protocol format.  </summary>  </member>  <member name="F:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants.WS\_FORMAT\_TEXT">  <summary>  Custom specific sub protocol format.  </summary>  </member>  <member name="F:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants.WS\_SUBPROT\_DEFAULT">  <summary>  Default protocol.  </summary>  </member>  <member name="F:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants.HOST">  <summary>  Host header from handshake.  </summary>  </member>  <member name="F:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants.UPGRADE">  <summary>  Upgrade header from handshake.  </summary>  </member>  <member name="F:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants.CONNECTION">  <summary>  Connection header from handshake.  </summary>  </member>  <member name="F:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants.SEC\_WEBSOCKET\_KEY">  <summary>  Sec-WebSocket-Key header from handshake.  </summary>  </member>  <member name="F:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants.ORIGIN">  <summary>  Sec-WebSocket-Origin header from handshake.  </summary>  </member>  <member name="F:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants.SEC\_WEBSOCKET\_PROTOCOL">  <summary>  Sec-WebSocket-Protocol header from handshake.  </summary>  </member>  <member name="F:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants.SEC\_WEBSOCKET\_VERSION">  <summary>  Sec-WebSocket-Version header from handshake.  </summary>  </member>  <member name="F:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants.SEC\_WEBSOCKET\_ACCEPT">  <summary>  Sec-WebSocket-Accept header from handshake.  </summary>  </member>  <member name="F:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants.SCHEME\_WS">  <summary>  Scheme ws for connection.  </summary>  </member>  <member name="F:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants.SCHEME\_WSS">  <summary>  Scheme wss for connection.  </summary>  </member>  <member name="F:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants.CONFIG">  <summary>  XML configuration file.  </summary>  </member>  <member name="F:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants.DEFAULT\_MAX\_FRAME\_SIZE">  <summary>  The default maximum frame size if not configured.  </summary>  </member>  <member name="F:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants.DEFAULT\_OPEN\_TIMEOUT">  <summary>  Default Session Timeout for client connections.  </summary>  </member>  <member name="F:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants.DEFAULT\_PING\_TIMEOUT">  <summary>  Default ping Timeout for client connections.  </summary>  </member>  <member name="F:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants.DEFAULT\_PING\_DELAY">  <summary>  Default ping delay for client connections.  </summary>  </member>  <member name="F:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants.WS\_SUPPORTED\_HYBI\_VERSIONS">  <summary>  WebSocket supported hixie versions.  </summary>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants.#cctor">  <summary>  Initializes the <see cref="T:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants"/> class.  </summary>`  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.common.WebSocketConstants.InitialiceXML">  <summary>  Initialices the XML config.  </summary>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.token.tbase.TokenFactory">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>3/26/2012</lastUpdate>  <summary>    </summary>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketEncoding">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>3/26/2012</lastUpdate>  <summary>  Type of encoding for data packet.  </summary>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketCookieManager">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>4/26/2012</lastUpdate>  <summary>    </summary>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.csharp.api.ICookiesManager">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>4/26/2012</lastUpdate>  <summary>  API for Cookie Manage  </summary>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.ICookiesManager.AddCookies(System.Collections.Generic.List{System.String},System.Uri)">  <summary>    </summary>  <param name="aCookies"></param>  <param name="aUri"></param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.ICookiesManager.GetCookies(System.Uri)">  <summary>    </summary>  <param name="aUri"></param>  <returns></returns>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.api.ICookiesManager.ProcessCookies(System.Web.HttpCookieCollection)">  <summary>    </summary>  <param name="aCookies"></param>  <returns></returns>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketReliabilityOptions">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>3/26/2012</lastUpdate>  <summary>  Reliability Options for connection.  </summary>  </member>  <member name="T:ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketHeaders">  <author>Rolando Betancourt Toucet</author>  <lastUpdate>3/26/2012</lastUpdate>  <summary>  Implementation of the request and response headers  </summary>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketHeaders.ReadResponseFromStream(System.Net.Sockets.NetworkStream)">  <summary>  Reads the response from stream.  </summary>  <param name="aSR">Data stream from server.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketHeaders.ReadRequestFromBuffer(System.Byte[])">  <summary>  Reads the request from buffer.  </summary>  <param name="aBuff">Data buffer from client.</param>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketHeaders.ToStringRequest">  <summary>  Gets request as string.  </summary>  <returns>String request</returns>  </member>  <member name="M:ClientLibrary.org.jwebsocket.client.csharp.kit.WebSocketHeaders.ToStringResponse">  <summary>  Gets response as string.  </summary>  <returns>String response</returns>  </member>  </members>  </doc> |