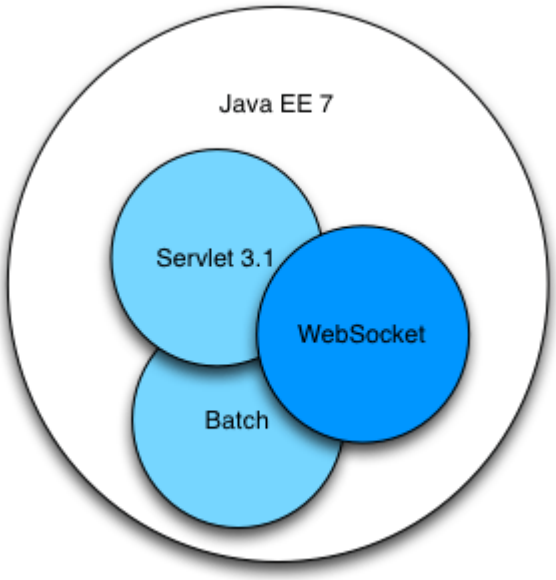
Tabla de Contenidos

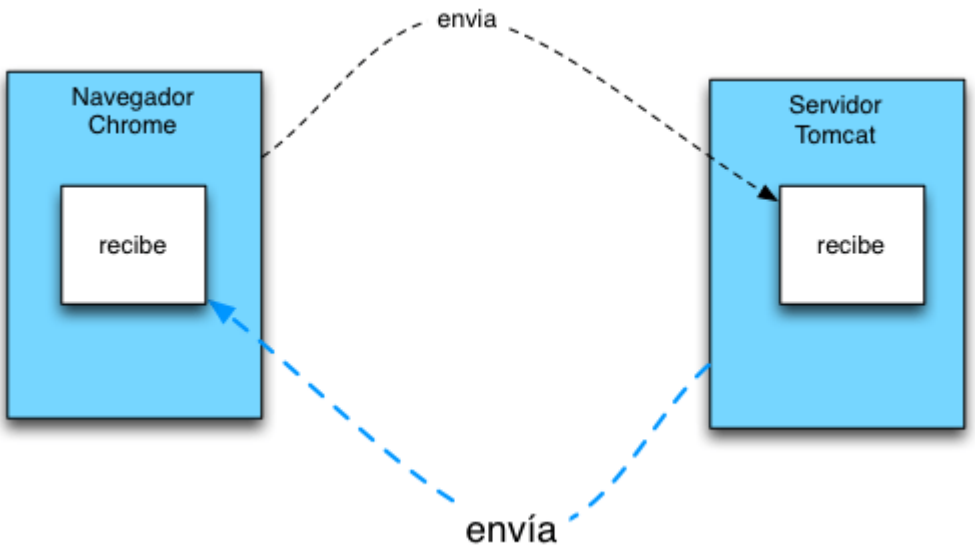
Cada día tenemos una mayor necesidad de utilizar las características que tiene HTML5 en nuestras aplicaciones. Pero quizá si hay una que destaca por encima de todas es el uso de los WebSockets. Java WebSockets es una de las especificaciones de Java EE 7 y vamos a introducirla en este artículo.



¿Que es un WebSocket?

Es una conexión bidireccional punto a punto entre un navegador y su servidor, de tal manera que ambas partes se puedan enviar mensajes. Para la parte del cliente no resulta ninguna innovación pero para la parte del servidor sí ya que permite enviar información al

cliente sin obligar a este a realizar peticiones AJAX cada x segundos.



Java WebSockets

En Java, los WebSockets vienen como estandar a partir de Java EE 7 en la especificación JSR 356. Aunque hay varios servidores que lo implementan en versiones anteriores ya sea cumpliendo el standard o ya sea a traves de una implementación propietaria. Yo en este caso utilizaré un Tomcat 8 que cumple con la especificación y podré utilizar las anotaciones estandard para crear de una forma sencilla un WebSocket.

Java WebSockets HolaMundo

Vamos a construir el ejemplo de HolaMundo para ello nos vamos a apoyar en dos anotaciones:

@ServerEndPoint : Define el punto de acceso al WebSocket en nuestro caso será

@ServerEndPoint(“/mensaje”) que nos define la ruta /mensaje como punto de acceso.

@OnMessage : Se encarga de definir el método del servidor que se ejecutará cuando llegue un mensaje desde el cliente via WebSocket . Vamos a ver ambas anotaciones en código.

```
package com.arquitecturajava;

import javax.websocket.OnMessage;
import javax.websocket.server.ServerEndpoint;

@ServerEndpoint("/mensaje")
public class MiWebSocket {

@OnMessage
public String mensaje(String mensaje) {

return "hola desde el servidor el mensaje es :" +mensaje;
}
}
```

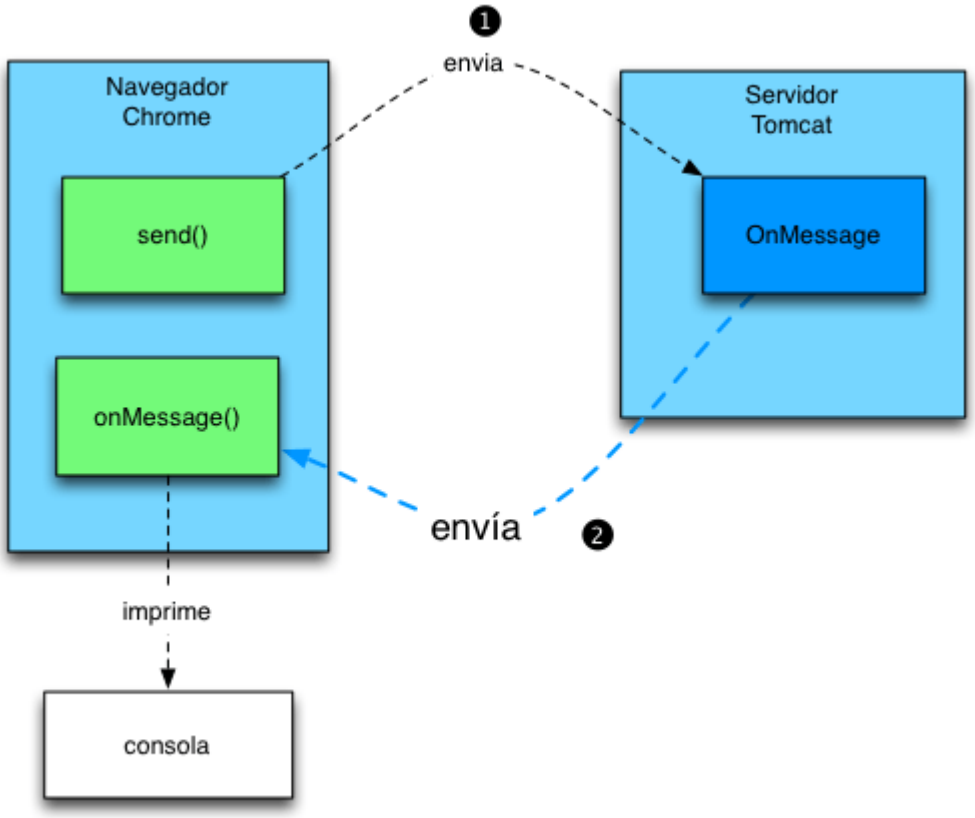
Java WebSocket Cliente

Podemos ver que el código es realmente sencillo .Nos falta por ver como construimos una página de cliente que se conecte a este WebSocket.

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
 pageEncoding="UTF-8"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<script type="text/javascript" src="jquery-1.11.0.js">

</script>
<script type="text/javascript">

$(document).ready(function() {

 var uriWS="ws://localhost:8080/WebSocketHolaMundo/mensaje";
 var miWebsocket= new WebSocket(uriWS);
 console.log (miWebsocket);

 miWebsocket.onopen=function(evento) {
 console.log("abierto");
 miWebsocket.send("hola");
 }

 miWebsocket.onmessage=function(evento) {

 console.log(evento.data);
 }

});
</script>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
```

```
<title>Insert title here</title>
</head>
<body>

</body>
</html>
```

Hemos usado el api de Javascript perteneciente a HTML 5 para la construcción de un Cliente de WebSocket . Ojo este caso es muy similar a los casos clásicos de AJAX y existe una sintaxis concreta para cada tipo de navegador (Chrome,Safari,Firefox) . En nuestro caso nos hemos centrado en Chrome. Hemos creado un WebSocket para él y le hemos solicitado que abra la conexión enviando un mensaje a través del método send(). Realizada esta operación el servidor recibirá el mensaje en su método @OnMessage y nos volverá a enviar una respuesta . Esta respuesta será capturada por el método onmessage del cliente e imprimida por la consola.



La consola nos mostrará un resultado como el siguiente :

```
▶ WebSocket {binaryType: "blob", extensions: "", protocol: "", onclose: null, onerror: null…}
abierto
hola desde el servidor el mensaje es :hola
>
```

Hemos terminado de construir nuestro WebSocket