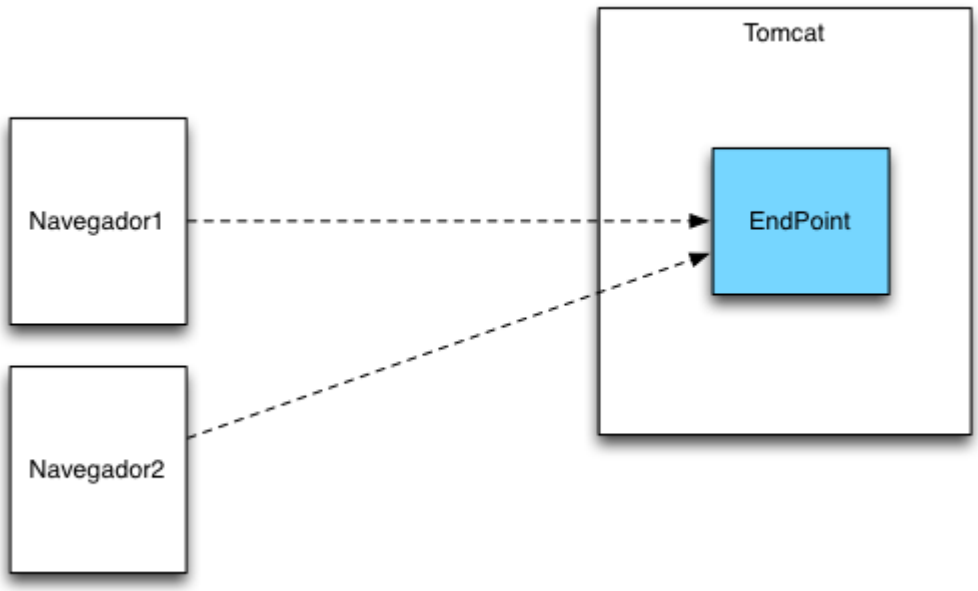
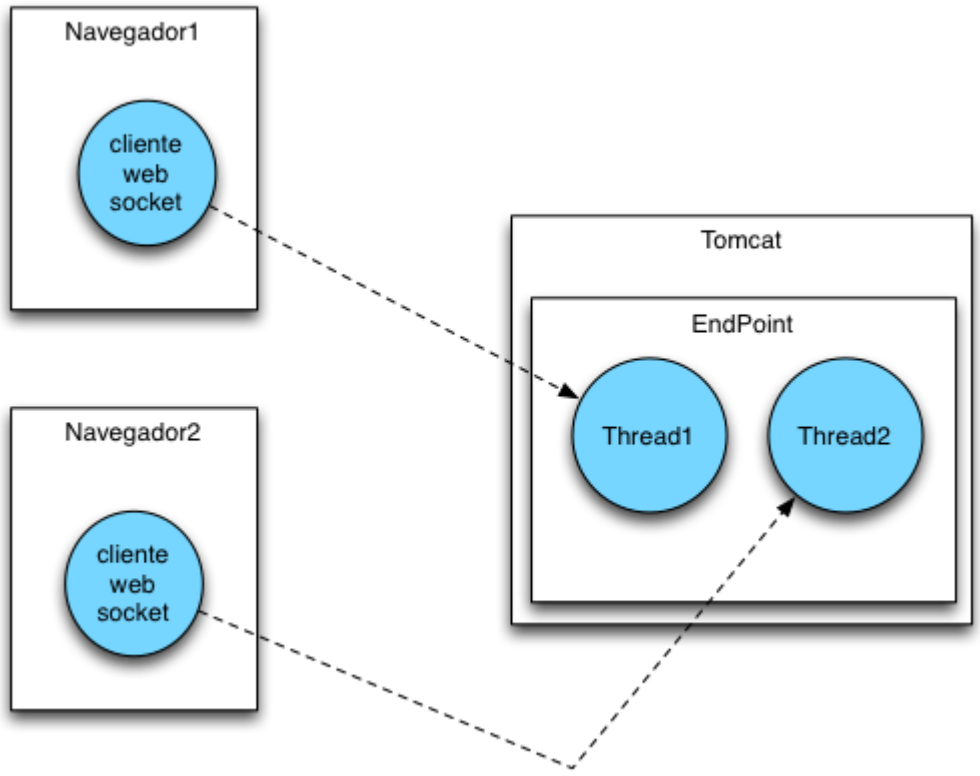
En el artículo anterior de introducción a WebSocket vimos como construir un WebSocket sencillo con Java y como cada navegador se convertía en un cliente para el Servidor que contenía un EndPoint de WebSocket. Ahora bien ¿cómo funciona exactamente este sistema a nivel de programación?. Si echamos un vistazo a lo que hemos construido nos dirá poco ya que simplemente tenemos un navegador y un EndPoint.



Threads y WebSockets

Ahora bien ¿como se construye esta relación realmente? . La realidad es que Tomcat asigna un Thread para cada una de las conexiones de WebSocket que generemos. Así pues cada uno de los clientes de Websocket están ligados a un Thread concreto. El diagrama sería mas cercano a lo siguiente.



Esto parece más lógico ya que cada Thread se encargará de estar pendiente de los mensajes que llegan por cada conexión de WebSoket.

WebSocket Programáticos y Threads

En el artículo anterior de WebSockets hemos construido un WebSocket orientado a anotaciones. Vamos a construir ahora un WebSocket sin anotaciones para cubrir el otro estilo de programación. Eso sí en este caso vamos a construir una arquitectura de WebSocket que trabaje con Threads y nos permita enviar desde el servidor mensajes de forma continua al cliente una vez que este haya abierto una conexión.

```
package com.arquitecturajava;

import java.io.IOException;

import javax.websocket.Endpoint;
import javax.websocket.EndpointConfig;
import javax.websocket.Session;

public final class MiWebSocketEndPoint extends Endpoint {

@Override
public void onOpen(Session session, EndpointConfig configuracion) {

final Session misesion = session;

Thread hilo = new Thread(new Runnable() {

@Override
public void run() {
try {
while(true) {

misesion.getBasicRemote().sendText("datos de la bolsa");
Thread.sleep(5000);
}

} catch (IOException e) {
// TODO Auto-generated catch block
e.printStackTrace();
} catch (InterruptedException e) {
// TODO Auto-generated catch block
```

```
e.printStackTrace();
}

}

});
hilo.start();
}

}
```

En este caso el WebSocket no usa anotaciones sino que extiende de WebSocket. En este caso simplemente sobrecargamos el método open. Como ya sabemos que cada conexión de WebSocket es gestionada por un Thread, abrimos otro nuevo Thread que envíe cada 5 segundos un mensaje al cliente escribiendo el texto de “datos de la bolsa” . Para que este WebSocket funcione correctamente tendremos que indicar un mapeo de URL. Esto se hace con la siguiente clase que lo configura y registra.

```
package com.arquitecturajava;

import java.util.HashSet;
import java.util.Set;

import javax.websocket.Endpoint;
import javax.websocket.server.ServerApplicationConfig;
import javax.websocket.server.ServerEndpointConfig;
```

```
public class ConfiguradorWebSocket implements ServerApplicationConfig
{

@Override
public Set<Class<?>> getAnnotatedEndpointClasses(Set<Class<?>> arg0) {
// TODO Auto-generated method stub
return null;
}

@Override
public Set<ServerEndpointConfig> getEndpointConfigs(
Set<Class<? extends Endpoint>> arg0) {
return new HashSet<ServerEndpointConfig>() {
{
add(ServerEndpointConfig.Builder.create(
MiWebSocketEndPoint.class, "/miwebsocket").build());
}
};
}

}
```

Realizadas ambas operaciones hemos registrado un nuevo EndPoint que por cada WebSocket que se conecte emitirá un mensaje cada 5 segundos al cliente.En último lugar podemos ver el código del cliente que se conecta al 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/WebSocketHolaMundo2/miwebsocket";
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>
```

Los mensajes se van guardando en la consola cada 5 segundos: