JSR 356: Building HTML5 WebSocket Apps in Java

Arun Gupta
Java EE & GlassFish Guy
blogs.oracle.com/arungupta, @arungupta

The preceding is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Agenda

- Primer on WebSocket
- JSR 356: Java API for WebSocket

Interactive Web Sites

- HTTP is half-duplex
- HTTP is verbose
- Flavors of Server Push
  - Polling
  - Long Polling
  - Comet/Ajax
- Complex, Inefficient, Wasteful

WebSocket to the Rescue

- TCP based, bi-directional, full-duplex messaging
- Originally proposed as part of HTML5
- IETF-defined **Protocol**: RFC 6455
  - Handshake
  - Data Transfer
- W3C defined **JavaScript API**
  - Candidate Recommendation

What's the basic idea ?

- Establish a connection (Single TCP connection)
- Send messages in both direction (Bi-directional)
- Send message independent of each other (Full Duplex)
- End the connection

What does it add over TCP ?

- Origin-based security model for browsers
- Adds an addressing and protocol naming mechanism to support multiple services on one port
- Layers a framing mechanism on top of TCP
- Includes an additional in-band closing handshake

Establish a connection

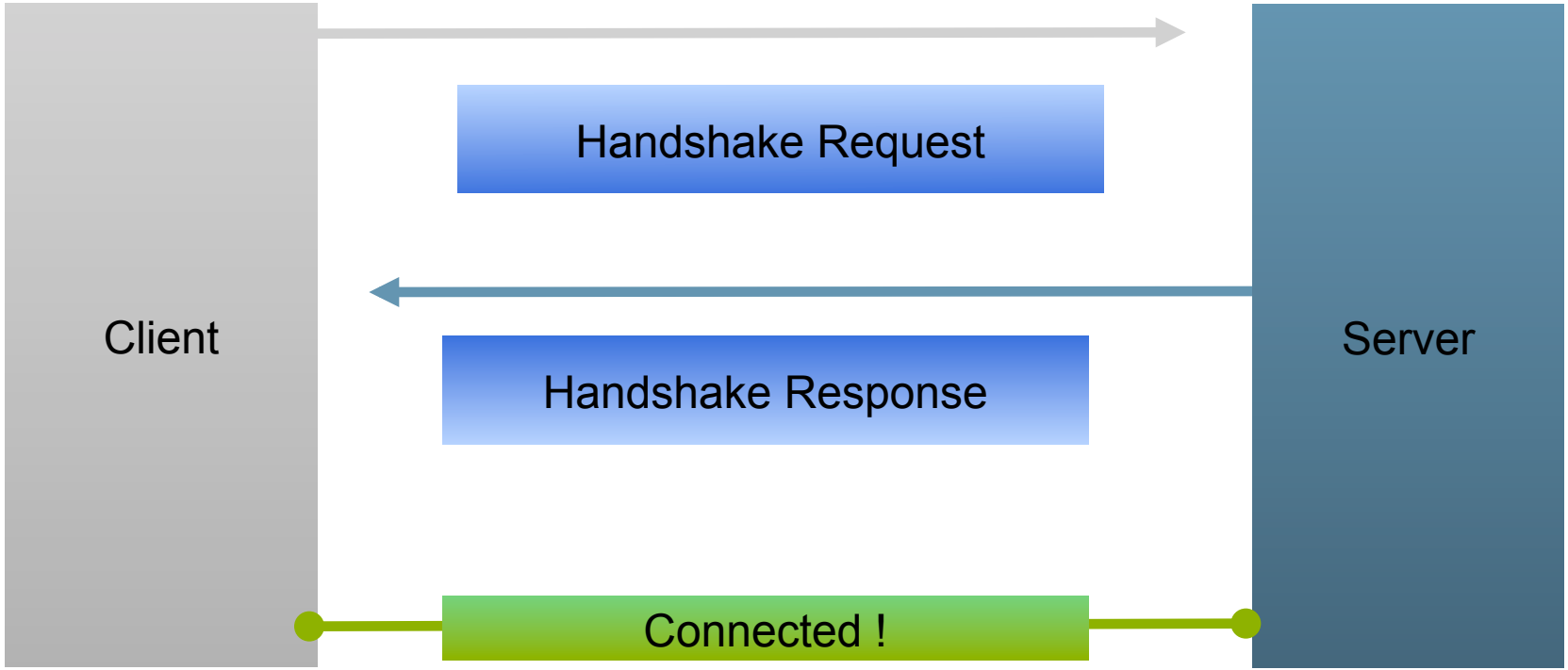
Handshake Request

```
GET /chat HTTP/1.1
Host: server.example.com
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Key: dGhlIHNhbXBsZSBub25jZQ==
Origin: http://example.com
Sec-WebSocket-Protocol: chat, superchat
Sec-WebSocket-Version: 13
```

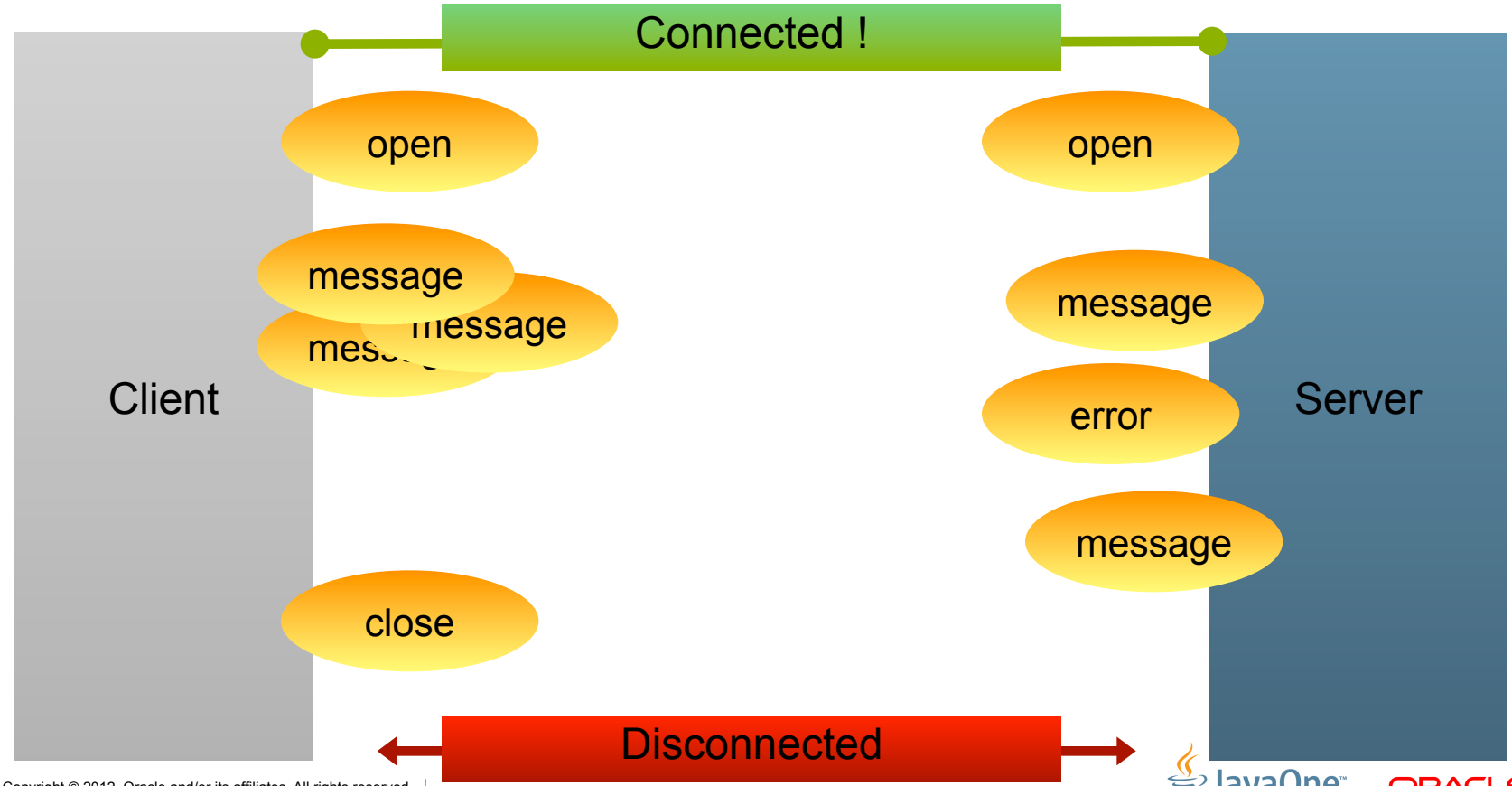
Handshake Response

```
HTTP/1.1 101 Switching Protocols
Upgrade: websocket
Connection: Upgrade
Sec-WebSocket-Accept: s3pPLMBiTxaQ9kYGzzhZRbK+xOo=
Sec-WebSocket-Protocol: chat
```

Establishing a Connection



WebSocket Lifecycle



WebSocket API

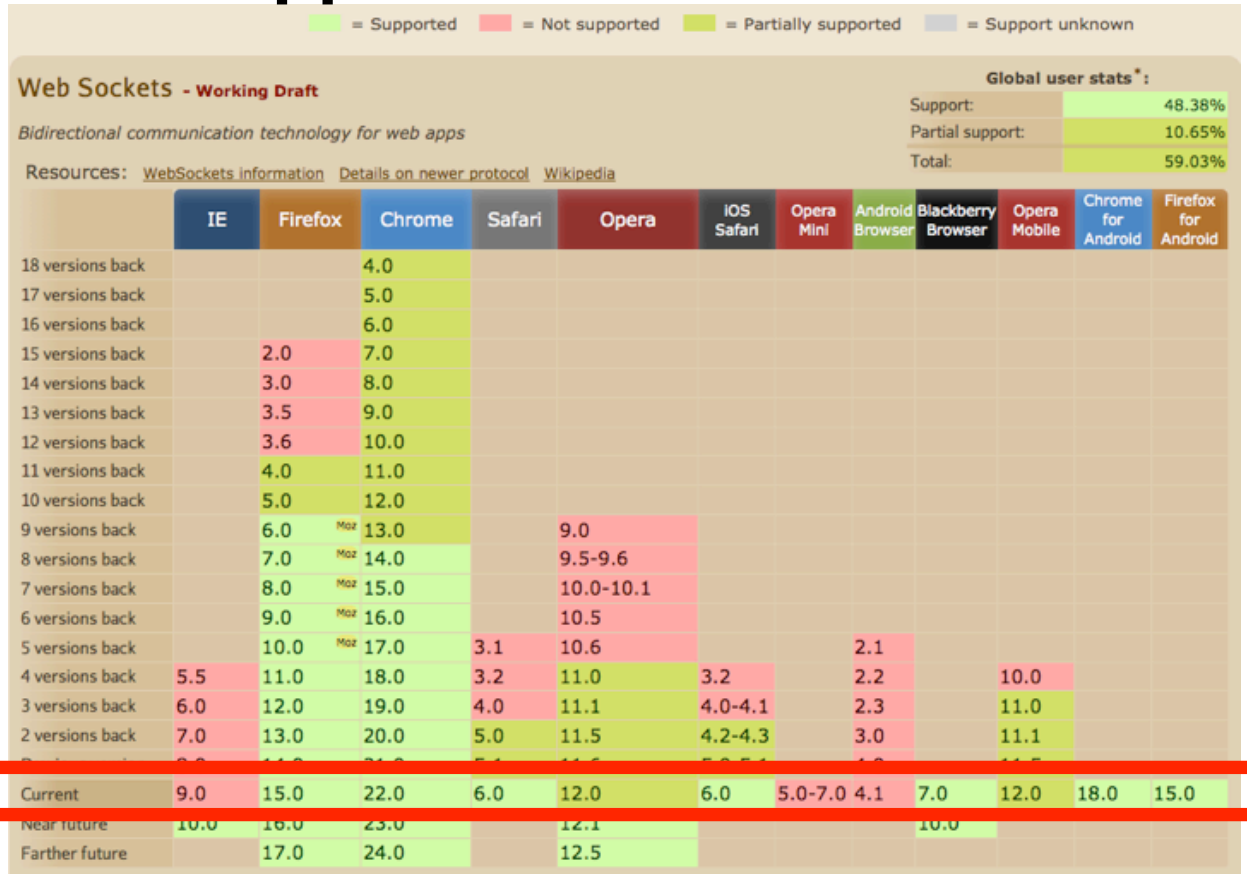
www.w3.org/TR/websockets/

```
[Constructor(DOMString url, optional (DOMString or DOMString[]) protocols)]
interface WebSocket : EventTarget {
  readonly attribute DOMString url;

  // ready state
  const unsigned short CONNECTING = 0;
  const unsigned short OPEN = 1;
  const unsigned short CLOSING = 2;
  const unsigned short CLOSED = 3;
  readonly attribute unsigned short readyState;
  readonly attribute unsigned long bufferedAmount;

  // networking
           attribute EventHandler onopen;
           attribute EventHandler onerror;
           attribute EventHandler onclose;
  readonly attribute DOMString extensions;
  readonly attribute DOMString protocol;
  void close([Clamp] optional unsigned short code, optional DOMString reason);

  // messaging
           attribute EventHandler onmessage;
           attribute DOMString binaryType;
  void send(DOMString data);
  void send(Blob data);
  void send(ArrayBuffer data);
  void send(ArrayBufferView data);
};
```

Java WebSocket Implementations

Java-WebSocket	Kaazing WebSocket Gateway
Grizzly	WebSocket SDK
Apache Tomcat 7	Webbit
GlassFish	Atmosphere
Autobahn	websockets4j
WeberKnecht	GNU WebSocket4J
Jetty	Netty
JBoss	TorqueBox
Caucho Resin	SwaggerSocket
jWebSocket	jWamp

Browser Support



= Supported = Not supported = Partially supported = Support unknown

Web Sockets - **Working Draft**

Bidirectional communication technology for web apps

Resources: WebSockets information | Details on newer protocol | Wikipedia

Global user stats*:

Support:	48.38%
Partial support:	10.65%
Total:	59.03%

	IE	Firefox	Chrome	Safari	Opera	iOS Safari	Opera Mini	Android Browser	Blackberry Browser	Opera Mobile	Chrome for Android	Firefox for Android
18 versions back			4.0									
17 versions back			5.0									
16 versions back			6.0									
15 versions back		2.0	7.0									
14 versions back		3.0	8.0									
13 versions back		3.5	9.0									
12 versions back		3.6	10.0									
11 versions back		4.0	11.0									
10 versions back		5.0	12.0									
9 versions back		6.0 Moz	13.0		9.0							
8 versions back		7.0 Moz	14.0		9.5-9.6							
7 versions back		8.0 Moz	15.0		10.0-10.1							
6 versions back		9.0 Moz	16.0		10.5							
5 versions back		10.0 Moz	17.0	3.1	10.6			2.1				
4 versions back	5.5	11.0	18.0	3.2	11.0	3.2		2.2		10.0		
3 versions back	6.0	12.0	19.0	4.0	11.1	4.0-4.1		2.3		11.0		
2 versions back	7.0	13.0	20.0	5.0	11.5	4.2-4.3		3.0		11.1		
Current	9.0	15.0	22.0	6.0	12.0	6.0	5.0-7.0	4.1	7.0	12.0	18.0	15.0
Near future	10.0	16.0	23.0		12.1				10.0			
Farther future		17.0	24.0		12.5							

http://caniuse.com/websockets

JSR 356 Specification

- Standard API for creating WebSocket Applications
- Transparent Expert Group
  - jcp.org/en/jsr/detail?id=356
  - java.net/projects/websocket-spec
- Now: Early Draft Review
- December: Public Draft Review
- Will be in Java EE 7
  - Under discussion: Client API in Java SE

JSR 356: Reference Implementation

- Tyrus: java.net/projects/tyrus
- Originated as WebSocket SDK
  - java.net/projects/websocket-sdk
- Pluggable Protocol Provider
  - Default is Grizzly/GlassFish
  - Portable to WebLogic
- Integrated in GlassFish 4 Builds

JSR 356 Expert Group

Jean-Francois Arcand	Individual
Scott Ferguson	Caucho Technology, Inc
Joe Walnes	DRW Holdings, LLC
Minehiko IIDA	Fujitsu Limited
Wenbo Zhu	Google Inc.
Bill Wigger	IBM
Justin Lee	Individual
Danny Coward	Oracle
Rémy Maucherat	RedHat
Moon Namkoong	TmaxSoft, Inc.
Mark Thomas	VMware
Wei Chen	Voxeo Corporation
Greg Wilkins	Individual

Java API for WebSocket Features

- Create WebSocket Client/Endpoints
  - Annotation-driven (`@WebSocketEndpoint`)
  - Interface-driven (`Endpoint`)
- SPI for extensions and data frames
- Integration with Java EE Web container

Touring the APIs

Note: The APIs might change before final release !

Hello World and Basics
POJO

Hello World

```
import javax.net.websocket.annotations.*;

@WebSocketEndpoint("/hello")
public class HelloBean {

    @WebSocketMessage
    public String sayHello(String name) {
        return “Hello “ + name;
    }
}
```

WebSocket Annotations

Annotation	Level	Purpose
`@WebSocketEndpoint`	class	Turns a POJO into a WebSocket Endpoint
`@WebSocketOpen`	method	Intercepts WebSocket Open events
`@WebSocketClose`	method	Intercepts WebSocket Close events
`@WebSocketMessage`	method	Intercepts WebSocket Message events
`@WebSocketPathParam`	method parameter	Flags a matched path segment of a URI-template
`@WebSocketError`	method	Intercepts errors during a conversation

@WebSocketEndpoint attributes

`value`	Relative URI or URI template e.g. “/hello” or “/chat/{subscriber-level}”
`decoders`	list of message decoder classnames
`encoders`	list of message encoder classnames
`subprotocols`	list of the names of the supported subprotocols

Custom Payloads

```
@WebSocketEndpoint(
    value="/hello",
    encoders={MyMessage.class},
    decoders={MyMessage.class}
)
public class MyEndpoint {
    . . .
}
```

Custom Payloads – Text

```
public class MyMessage implements Decoder.Text<MyMessage>,
Encoder.Text<MyMessage> {
  private JsonObject jsonObject;

  public MyMessage decode(String s) {
    jsonObject = new JsonReader(new StringReader(s)).readObject();
    return this;
  }
  public boolean willDecode(String string) {
    return true;  // Only if can process the payload
  }

  public String encode(MyMessage myMessage) {
    return myMessage.jsonObject.toString();
  }
}
```

Custom Payloads – Binary

```
public class MyMessage implements Decoder.Binary<MyMessage>,
Encoder.Binary<MyMessage> {

  public MyMessage decode(byte[] bytes) {
    . . .
    return this;
  }
  public boolean willDecode(byte[] bytes) {
    . . .
    return true;  // Only if can process the payload
  }

  public byte[] encode(MyMessage myMessage) {
    . . .
  }
}
```

Chat Sample

```
@WebSocketEndpoint("/chat")
public class ChatBean {
    Set<Session> peers = Collections.synchronizedSet(…);

    @WebSocketOpen
    public void onOpen(Session peer) {
        peers.add(peer);
    }

    @WebSocketClose
    public void onClose(Session peer) {
        peers.remove(peer);
    }

    . . .
```

Chat Sample

```
    . . .

    @WebSocketMessage
    public void message(String message, Session client) {
        for (Session peer : peers) {
            peer.getRemote().sendObject(message);
        }
    }
}
```

URI Template Matching

- Level 1 only

```
@WebSocketEndpoint("/orders/{order-id}")
public class MyEndpoint {
  @WebSocketMessage
  public void processOrder(
    @WebSocketPathParam("order-id")String orderId) {
    . . .
  }
}
```

Which methods can be @WebSocketMessage ?

- A parameter type that can be decoded in incoming message
  - `String`, `byte[]`, `ByteBuffer` or any type for which there is a decoder
- An optional `Session` parameter
- 0..n `String` parameters annotated with `@WebSocketPathParameter`
- A return type that can be encoded in outgoing message
  - `String`, `byte[]`, `ByteBuffer` or any type for which there is a encoder

WebSocket Subprotocols

- Facilitates application layer protocols
- Registered in a Subprotocol Name Registry
  - Identifier, Common name, Definition
  - www.iana.org/assignments/websocket/websocket.xml#subprotocol-name
- 4 officially registered
  - Message Broker (2 versions)
  - SOAP
  - WebSocket Application Messaging Protocol (WAMP)
    - RPC, PubSub

WebSocket Subprotocols

TBD

WebSocket Extensions

- Add capabilities to the base protocol
  - Endpoints negotiate during opening handshake
  - “Formal” or Private (prefixed with “x-”)
- Multiplexing
  - http://tools.ietf.org/html/draft-tamplin-hybi-google-mux
- Compression: Only non-control frames/messages
  - Per-frame: http://tools.ietf.org/html/draft-tyoshino-hybi-websocket-perframe-deflate
  - Per-message: http://tools.ietf.org/html/draft-ietf-hybi-permessage-compression

WebSocket Extensions

Out of scope, prompted by EG

- `Extension.createIncomingFrameHandler`, `createOutgoingFrameHandler`
- Added to `EndpointConfiguration`

Packaging – Java EE Style

- Client side
  - Classes + resources packaged as a JAR
- Web Container
  - Classes + resources packaged in a WAR file

Hello World and Basics
Non-POJO

Hello World Server

```
import javax.net.websocket.*;

public class HelloServer extends Endpoint {
  @Override
  public void onOpen(Session session) {
    session.addMessageHandler(new MessageHandler.Text() {
      public void onMessage(String name) {
        try {
          session.getRemote().sendString("Hello " + name);
        } catch (IOException ex) {
        }
      }
    });
  }
}
```

Server Configuration - Bootstrap

```
URI serverURI = new URI("/hello");
ServerContainer serverContainer =
    ContainerProvider.getServerContainer();
Endpoint helloServer = new HelloServer();
ServerEndpointConfiguration serverConfig =
    new DefaultServerConfiguration(serverURI);
serverContainer.publishServer(helloServer, serverConfig);
```

Recommended in `ServletContextListener` *

Hello World Client

```
import javax.net.websocket.*;

public class HelloClient extends Endpoint {
  @Override
  public void onOpen(Session session) {
    try {
      session.getRemote().sendString("Hello you !");
    } catch (IOException ioe) {
      // . . .
    }
  }
}
```

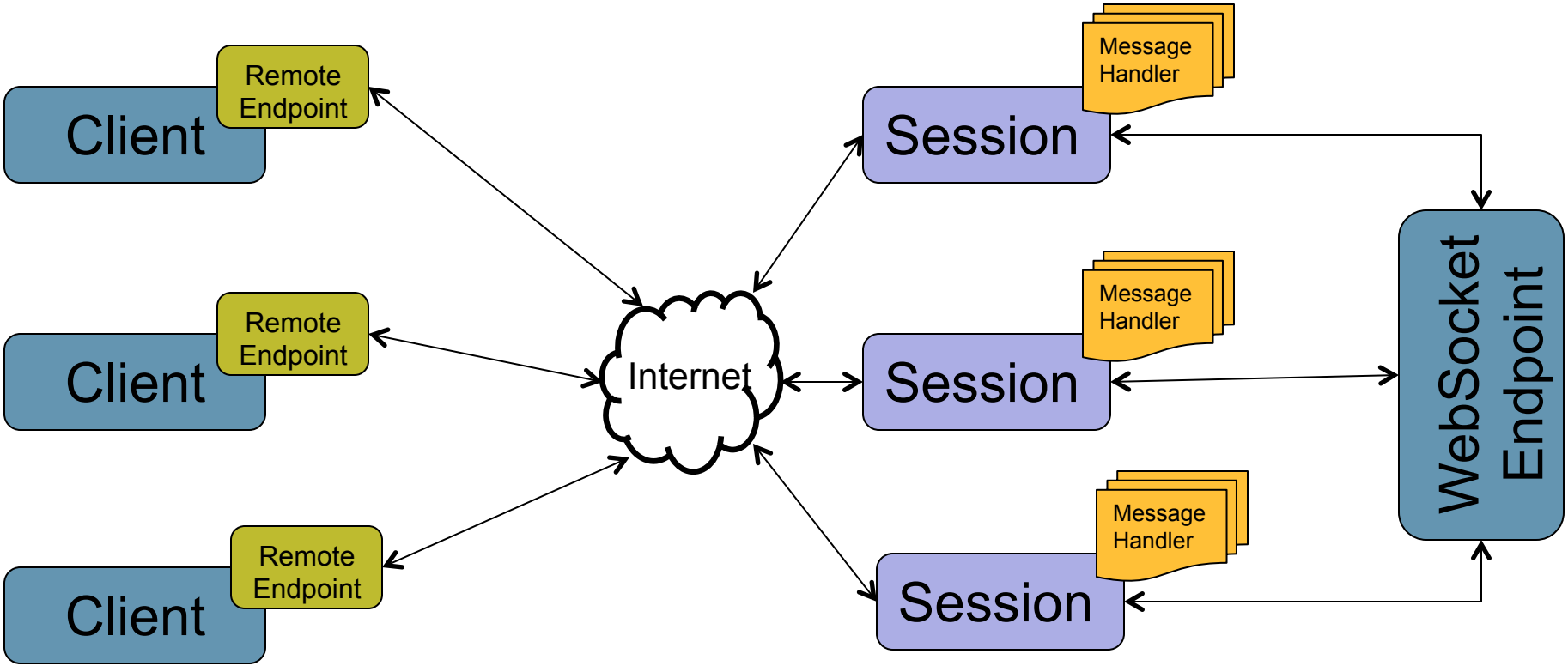
Server and Client Configuration

- Server
  - URI matching algorithm
  - Subprotocol and extension negotiation
  - Message encoders and decoders
  - Origin check
  - Handshake response
- Client
  - Requested subprotocols and extensions
  - Message encoders and decoders
  - Request URI

Main API Classes: javax.net.websocket.*

- **Endpoint**: Intercepts WebSocket lifecycle events
- **MessageHandler**: Handles all incoming messages for an Endpoint
- **RemoteEndpoint**: Represents the ‘other end’ of this conversation
- **Session**: Represents the active conversation

Object Model



Sending the Message

Whole string *	`RemoteEndpoint`	`sendString(String message)`
Binary data *	`RemoteEndpoint`	`sendString(ByteBuffer message)`
String fragments	`RemoteEndpoint`	`sendPartialString(String part, boolean last)`
Binary data fragments	`RemoteEndpoint`	`sendPartialData(ByteBuffer part, boolean last)`
Blocking stream of text	`RemoteEndpoint`	`Writer getSendWriter())`
Blocking stream of binary data	`RemoteEndpoint`	`OutputStream getSendStream()`
Custom object of type T *	`RemoteEndpoint<T>`	`sendObject(T customObject)`

* additional flavors: by completion, by future

Receiving the Message

Whole string	`MessageHandler.Text`	`onMessage(String message)`
Binary data	`MessageHandler.Binary`	`onMessage(ByteBuffer message)`
String fragments	`MessageHandler.AsyncText`	`onMessage(String part, boolean last)`
Binary data fragments	`MessageHandler.AsyncBinary`	`onMessage(ByteBuffer part, boolean last)`
Blocking stream of text	`MessageHandler.CharacterStream`	`onMessage(Reader r)`
Blocking stream of binary data	`MessageHandler.BinaryStream`	`onMessage(InputStream r)`
Custom object of type T	`MessageHandler.DecodedObject<T>`	`onMessage(T customObject)`

Relationship with Servlet 3.1

- Allows a portable way to upgrade HTTP request
- New API
  - `HttpServletRequest.upgrade(ProtocolHandler handler)`

Security

- Authenticates using Servlet security mechanism during opening handshake
  - Endpoint mapped by `ws://` is protected using security model defined using the corresponding `http://` URI
- Authorization defined using `<security-constraint>`
  - TBD: Add/reuse security annotations
- Transport Confidentiality using `wss://`
  - Access allowed over encrypted connection only

Advanced Features

- DataFrame SPI
- Integration with Java EE and Web Applications
  - Injectable components

API TODO

Lots …

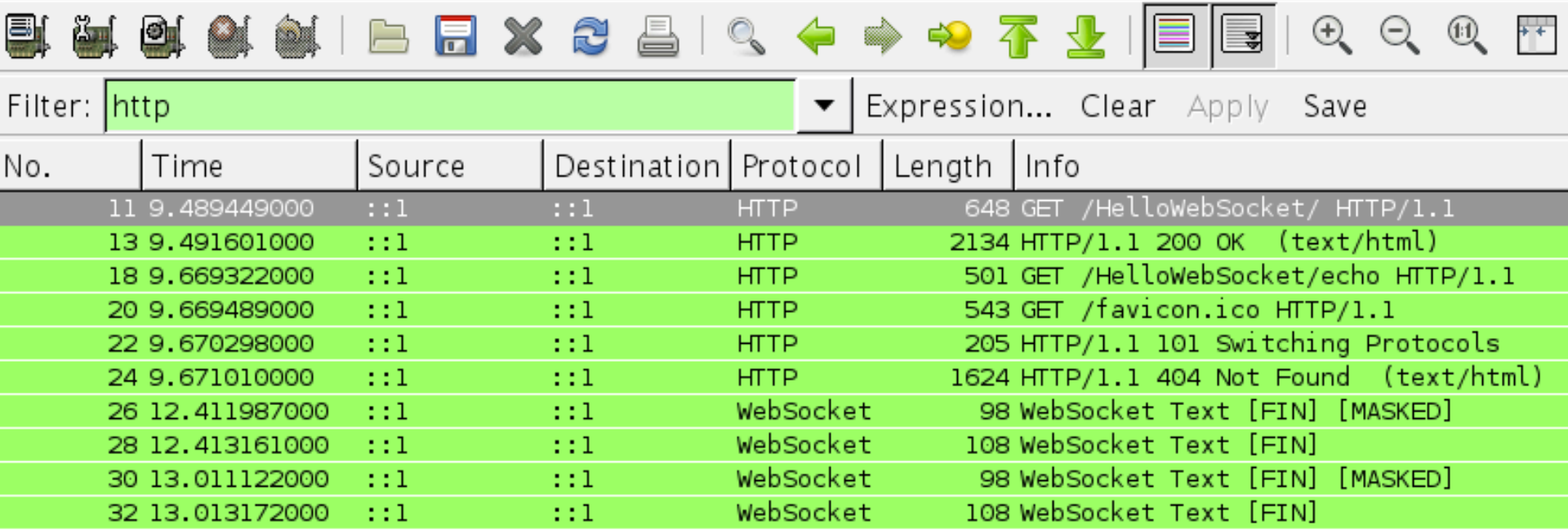
- Refactoring/renaming
  - Class naming, fluency
  - Collapse MessageHandlers
  - Re-org/rename annotations

Use of @WebSocketEndpoint on Endpoint instead of ServerConfiguration API

- More knobs and dials on POJO
- Exception handling
- Integration with Java EE

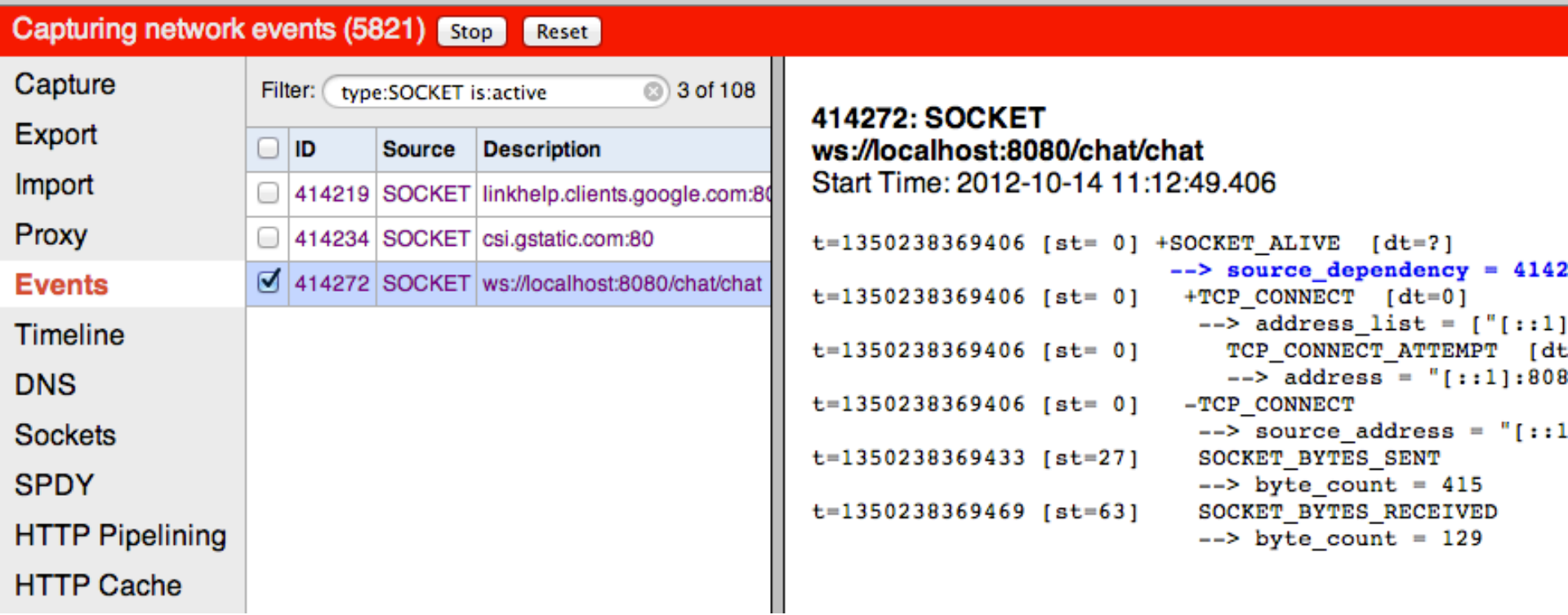
How to view WebSocket messages ?

Capture traffic on loopback



How to view WebSocket messages ?

chrome://net-internals -> Sockets -> View live sockets



WebSocket Future ???

- Protocol Updates
- HTTP 2.0
- SPDY

Resources

- Specification
  - JSR: jcp.org/en/jsr/detail?id=356
  - Mailing Lists, JIRA, Archive: java.net/projects/websocket-spec
  - Now: Early Draft Review
  - Will be in Java EE 7
- Reference Implementation
  - Tyrus: java.net/projects/tyrus
  - Now: Integrated in GlassFish 4 builds

Q & A

MAKE THE FUTURE JAVA

ORACLE


JavaOne™