Develop Real Time HTML5 Applications using WebSocket with Apache Camel & ActiveMQ

Camel One
16th of May - 2012

Charles Mouliard
Sr. Principal Solution Architect
Apache Committer
Speaker: Charles Moulliard

• Engineer in Agronomy & Master in Zoology
  – 18 years of experience in IT world development
  – Specialized in new technologies Web & Integration
  – Project manager in Bank, Financial, Telco world

• Solution Architect at FuseSource

• Committer on projects: Apache ServiceMix, Apache Karaf (PMC), Apache Camel, Fuse Fabric

- Twitter: http://twitter.com/cmoulliard
- LinkedIn: http://www.linkedin.com/in/charlesmoulliard
- My blog: http://cmoulliard.blogspot.com
- Slideshare: http://www.slideshare.net/cmoulliard
Agenda

- Introduction
  - History & HTML5
  - Websocket revealed
- ActiveMQ
  - Stomp and WebSocket
- Camel
  - Camel-websocket
- Demo
- Conclusion
Agenda

- **Introduction**
  - History & HTML5
  - Websocket revealed
- **ActiveMQ**
  - Stomp and WebSocket
- **Camel**
  - Camel-websocket
- **Demo**
- **Conclusion**
History

- Started with Java Applet and Shockwave player (199x)
- Followed by Macromedia Shockwave, Adobe Flash, Microsoft Silverlight
- Goal: Develop client-server application using HTTP/RMI protocol
Real Time HTML5 Apps - Introduction

- Evolution: Animation → Design Rich Internet Applications (Intranet)
- Limitations about Technology based on Web 1.0
  - No push → require lot of HTTP requests
  - RMI protocol → blocked by Firewall
  - Exchanges mainly use XML - WebServices
  - Compatibility between plugin(s) / browser
  - Code should be signed (to access local resources)
Real Time HTML5 Apps - Introduction

- **Web 2.0 “Revolution”**
  - HTTPs requests
  - Due to volume of data (video, audio, text) exchanged between “social networks”
  - User becomes “actor” – “collaborating”

- New solutions have been imagined to support that →
  - Ajax technology (2005), JSON instead of XML, RestFull
Real Time HTML5 Apps - Introduction

- JavaScript client uses asynchronous communication with server

→ Ajax Reverse, but not really PUSH

*Figure 1: How AJAX Technologies Handle a User Action*
First Strategy Developed
- Cometd/Bayeux protocol

Principe
- Use a **persistent** or **long-lasting** HTTP connection with HTTP server (streaming / polling)
- **Events** based to push data to the browser
- Connection is keep alive till timeout
Real Time HTML5 Apps - Introduction

- **Drawbacks:**
  - Browser (HTTP 1.x spec) only allow 2 connections with HTTP server → 1 connection will be kept alive for Cometd/Bayeux server
  - Any new event requests a new HTTP call
  - Bayeux communication protocol has not been standardized through instances IETF, W3C
  - Exchanges based on “Bayeux” messages (and not XML, JSON, ...) – more complex to handle
  - One Way communication (server → client)
Real Time HTML5 Apps - Introduction

- 2nd approach - WebSocket
  - ~= html + css + js
  - Offline storage
  - RealTime/Communication
  - File/Hardware Access
  - CSS 3
  - Graphics/Multimedia
  - Supported by Tablet/Pc/Mobile
Websocket revealed

- Full-duplex single socket connection
- HTTP request followed by WebSocket data packets exchange
- ws:// and wss:// protocol
- Developed part of HTML5 initiative
- Specification [rfc-6455](http://rfc-editor.org/rfc/rfc6455) (Dec-2011) managed by IETF Task Force
Real Time HTML5 Apps - Introduction

- What the browser sends

Request Headers

```plaintext
GET ws://localhost:9090/newsTopic HTTP/1.1
Origin: http://127.0.0.1:8080
Connection: Upgrade
Host: localhost:9090
Sec-WebSocket-Key: CTEq35HHaHuT+1Gs0MB3qQ==
Upgrade: websocket
Sec-WebSocket-Version: 13
```

Key which is base64 encoded and uses as handshake between client and server

Version of WebSocket used - allow client and server to check if they are compatible
And what it receives

Response Headers

HTTP/1.1 101 Switching Protocols
Connection: Upgrade
Upgrade: WebSocket
Sec-WebSocket-Accept: rygeGXZxH4n5ZWXC2emIAwi7jdA=

Response created by the server (client key + GUI) signed SHA-1 and encoded in base64
Next Data frames are exchanged back and forth between client and server through TCP/IP connection

- Bytes or UTF-8 Text packets are supported
- Text is mainly used within JavaScript
Real Time HTML5 Apps - Introduction

- **Benefits**
  - Use same ports as HTTP (80 and HTTPS (473))
  - Bandwidth reduction
  - No long polling process
  - No more unnecessary traffic
  - Standard based
  - Security managed though Web Server (SSL, Authentication)
Agenda

- Introduction
  - History & HTML5
  - Websocket revealed
- **ActiveMQ**
  - Stomp and WebSocket
- Camel
  - Camel-websocket
- Demo
- Conclusion
Real Time HTML5 Apps - ActiveMQ

- Provide a websocket implementation using ws:// transport connector
- Use Jetty WebSocket v7.5
- Is available since ActiveMQ 5.6
- Stomp is the wire format

More info:
http://activemq.apache.org/websockets.html
Real Time HTML5 Apps - ActiveMQ

1. HTTP Request (Upgrade)
2. HTTP Response
3. Stomp Subscribe
4. Message received from Topic
5. Send WebSocket Data

Client (Browser) -> WebSocket Connector -> Stomp Converter -> ActiveMQ (Backend)
Server side

1) Transport connector must be enabled for websocket

```xml
<transportConnectors>
  <transportConnector name="openwire" url="tcp://0.0.0.0:61616"/>
  <transportConnector name="websocket" url="ws://0.0.0.0:61614"/>
</transportConnectors>
```

Add `ws://` transport connector

2) Topics/Queues will be used as channels to send (or receive messages)

```java
// create the connection factory
ActiveMQConnectionFactory connectionFactory = new ActiveMQConnectionFactory("tcp://localhost:61616");
Connection connection = connectionFactory.createConnection("guest", "password");
//Connection connection = connectionFactory.createConnection();
connection.start();

// Create the session and topic
Session session = connection.createSession(false, Session.AUTO_ACKNOWLEDGE);
Topic quoteTopic = session.createTopic("stockQuoteTopic");
```

Nothing special...
Real Time HTML5 Apps - ActiveMQ

- Client side
  - Need js client
    - A) stomp.js : Author→ Jeff Mesnil
      - Code : https://github.com/jmesnil/stomp-websocket
      - Doc : http://www.jmesnil.net/stomp-websocket/doc/
    - B) stomple : Author→ Karl Krukow
      - Code : https://github.com/krukow/stomple
2) Create a WebSocket connection with ActiveMQ using Stomp.js client

A. Create a client to connect to the server

```javascript
$(document).ready(function() {
    var client, destinationQuotes;
    $('#connect_form').submit(function() {
        var url = $('#connect_url').val();
        var login = $('#connect_login').val();
        var passcode = $('#connect_passcode').val();
        destinationQuotes = $('#destinationQuotes').val();

        client = Stomp.client(url);
        client.connect(login, passcode, onconnect);

        client.debug = function(str) {
            $('#debug').append(str + '\n');
        };

        // this allows to display debug logs directly on the web page
        var onconnect = function(frame) {
            // the client is notified when it is connected to the server.
        };

        url : ws://localhost:61614/stomp:
        Destination : /topic/stockQuoteTopic
    });
});
```

JQuery helps us to display result in HTML page

C. Wait response ...

C. Establish connection using login, password

© 2012 FuseSource Corp. All rights reserved
• 3) Subscribe to the topic (= channel)

Subscribe to the topic and listen on messages

client.subscribe(destinationQuotes, function(message) {
    var quote = JSON.parse(message.body);
    $('.' + 'stock-' + quote.symbol).replaceWith('<tr class="stock" + quote.symbol + '">
    <td>' + quote.symbol + '</td>' +
    '<td>' + quote.open.toFixed(2) + '</td>' +
    '<td>' + quote.last.toFixed(2) + '</td>' +
    '<td>' + quote.change.toFixed(2) + '</td>' +
    '<td>' + quote.high.toFixed(2) + '</td>' +
    '<td>' + quote.low.toFixed(2) + '</td>' +
    '</tr>');
})

Parse JSON messages

• 4) When job is done → close connection

Close here
Benefits

- Acts as real time data feed provider (topic)
- Infrastructure can be secured
  - Web Server level (SSL, Authentication)
  - ActiveMQ “security plugin”
- By combining Scalability/High Availability features of ActiveMQ, feed are persisted, distributed
Agenda

- Introduction
  - History & HTML5
  - Websocket revealed
- ActiveMQ
  - Stomp and WebSocket
- Camel
  - Camel-websocket
- Demo
- Conclusion
Real Time HTML5 Apps - Camel

- Camel-websocket component - part of Camel 2.10 ([http://camel.apache.org/websocket.html](http://camel.apache.org/websocket.html))
- Use Jetty WebServer v7.5
- Can be used to produce or consume
- Allow to send packets to multiple clients ("sendToAll") or to individual
- Should be combined with EIP patterns to aggregate/split/enrich data and/or other components (RSS, Atom, Twitter, JSon, ...)

© 2012 FuseSource Corp. All rights reserved
Real Time HTML5 Apps - Camel

1. HTTP Request (Upgrade)
2. HTTP Response
3. Connect to channel
4. Message consumed From Topic
5. On Camel Exchange received
6. Send WebSocket Data

WebSocket Component
Data Format
JMS Component

Client (Browser)
WebSocket Component
TCP (WebSocket)
HTTP(S) Request/Response

Camel

ActiveMQ (Topic)
TCP (WebSocket)
HTTP(S)
Request/Response
Data
Format 
JMS 
Component 
ActiveMQ (Topic)
1) Create a Camel Route to produce websocket packets → Http Client

```java
public class WebSocketStockPricesRoute extends RouteBuilder {

    @Override
    public void configure() throws Exception {

        from("activemq:topic:stockQuoteTopic")
            .log(LoggingLevel.DEBUG, "〉 Stock price received : 
                    ${body}")
            .to("websocket:stockQuoteTopic?sendToAll=true");

    }
}
```

Data consumed from a topic are send to all the WebSocket clients connected
1) or Create a Camel Route to consume - produce websocket packets

```java
public class WebSocketChatRoute extends RouteBuilder {
    @Override
    public void configure() throws Exception {
        from("websocket:chat-room")
            .log(LoggingLevel.INFO, "Message received : ${body}")
            .to("websocket:chat-room?sendToAll=true");
    }
}
```

WebSockets packets are received by the websocket:chat-room endpoint.

Packets sent by client in the channel "chat-room" are resent to all the clients connected to the "chat-room".
2) Client side - Will use WebSocket js script in combination with jQuery, jQuery-ui, ... to communicate

```javascript
var host1 = $(' doctr1').val();
var host2 = $(' doctr2').val();

socket1 = new WebSocket(host1);
socket2 = new WebSocket(host2);

// Add a connect listener
socket1.onopen = function () {
    $('#msg').append('<p class="event">Socket News Status: ' + socket1.readyState + ' (open)</p>');
};
```

First, create a WebSocket connection

4 methods are available:
- `onopen`
- `onmessage`
- `onerror`
- `onclose`
2) Parse messages

Every websocket “message” is consumed here

Message is parsed using JSON (text → Object)

Next we display the result in the HTML page
2) Close connection

```javascript
$('#disconnect_form').submit(function () {
    socket1.close();
    socket2.close();

    $('#msg').append('<p class="event">Socket News Status: ' + socket1.readyState + ' (Closed)</p>');
    $('#msg').append('<p class="event">Socket Tweet Status: ' + socket2.readyState + ' (Closed)</p>');

    $('#disconnect').fadeOut({ duration: 'fast' });
    $('#connect').fadeIn();
    $('#send_form_input').addAttr('disabled');

    return false;
});
```
Benefits Provided by Camel framework

- Camel acts as a mediation router between “feed” providers and “feed” consumers
- Will facilitate “aggregation”, “enrichment”, “filtering” of the feeds receive before to distribute them
- Will provide the components required to interconnect “platforms”
- Camel + ActiveMQ + Fabric = Scalability, High Availability, Cloud of data Feed
Agenda

- Introduction
  - History & HTML5
  - Websocket revealed
- ActiveMQ
  - Stomp and WebSocket
- Camel
  - Camel-websocket
- Demo
- Conclusion
ActiveMQ – Stock Trader Demo 1
Real Time HTML5 Apps - Demo

Camel – Chat Room Demo 2

This is a CamelOne Demo of the camel-websocket component.
Camel – News Feed Demo 3

Need help: Send messages to @fusenews
Agenda

- Introduction
  - History & HTML5
  - Websocket revealed
- ActiveMQ
  - Stomp and WebSocket
- Camel
  - Camel-websocket
- Demo
- Conclusion
Real Time HTML5 Apps - Conclusion

- ActiveMQ and Camel are ready for “Real Time HTML5 Application Development”
- Provide best of the Fuse - Apache technology for integration (EIP and components)
- Will help you to drive your business (transport, tracking,...)
- Available for any device .... Tablet/iPhone/Android/Pc/Mac
Any Questions?

- Demo code: https://github.com/FuseByExample/websocket-activemq-camel
- Twitter: @cmoulliard
- More info: http://fusesource.com