SPRINGONE2GX
WASHINGTON, DC

# From 0 to Hero with Spring WebSocket

Sergi Almar
@sergialmar




# Agenda

- Realtime Web
  - Evolution of the web
  - Server Sent Events
- WebSocket
  - Intro
  - Spring WebSocket
  - Scaling
  - Security

# Agenda

- Realtime Web
  - **Evolution of the web**
  - Server Sent Events
- WebSocket
  - Intro
  - Spring WebSocket
  - Scaling
  - Security

Google Analytics

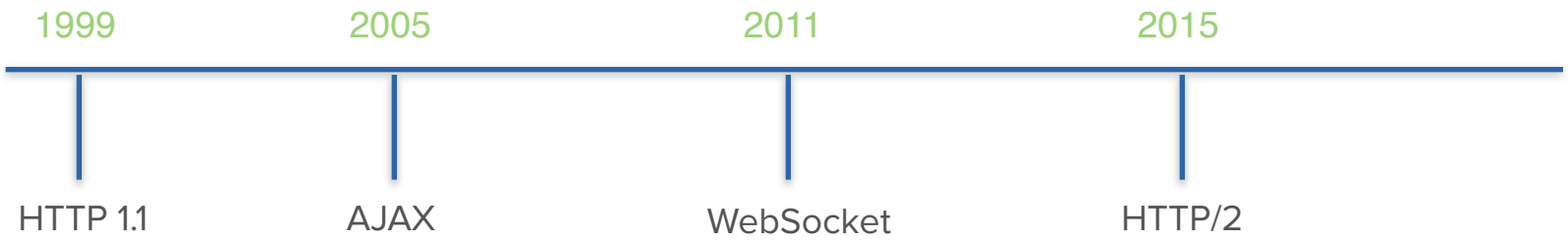
# HOW MANY WEB APPLICATIONS WITH REALTIME NOTIFICATIONS DO YOU USE?

FOURSQUARE

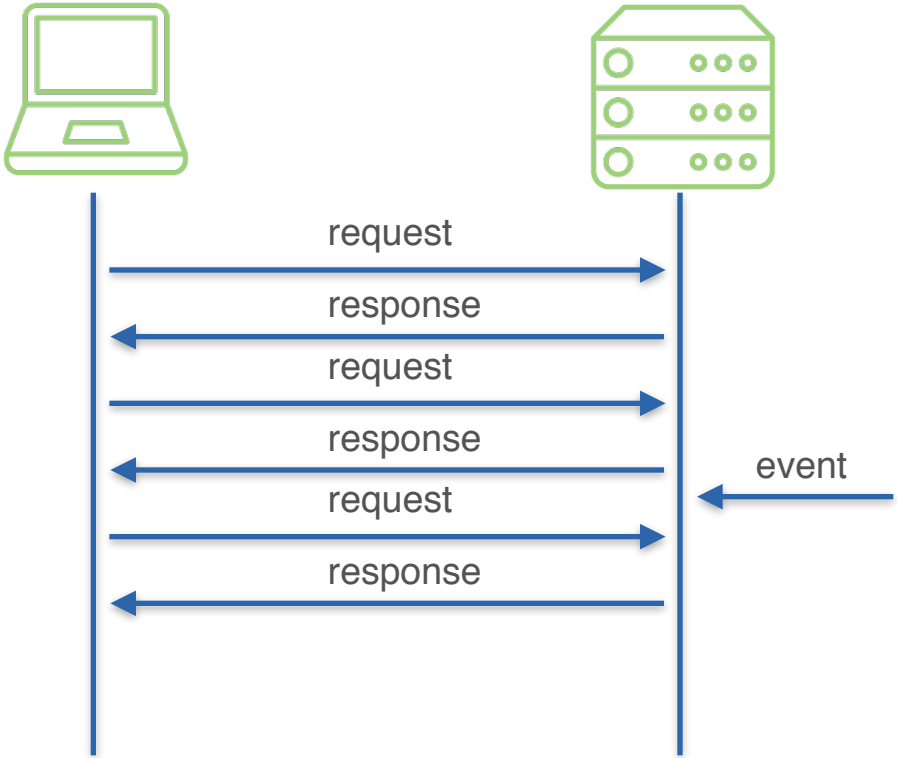
stackoverflow

Dropbox

# Evolution



# Polling

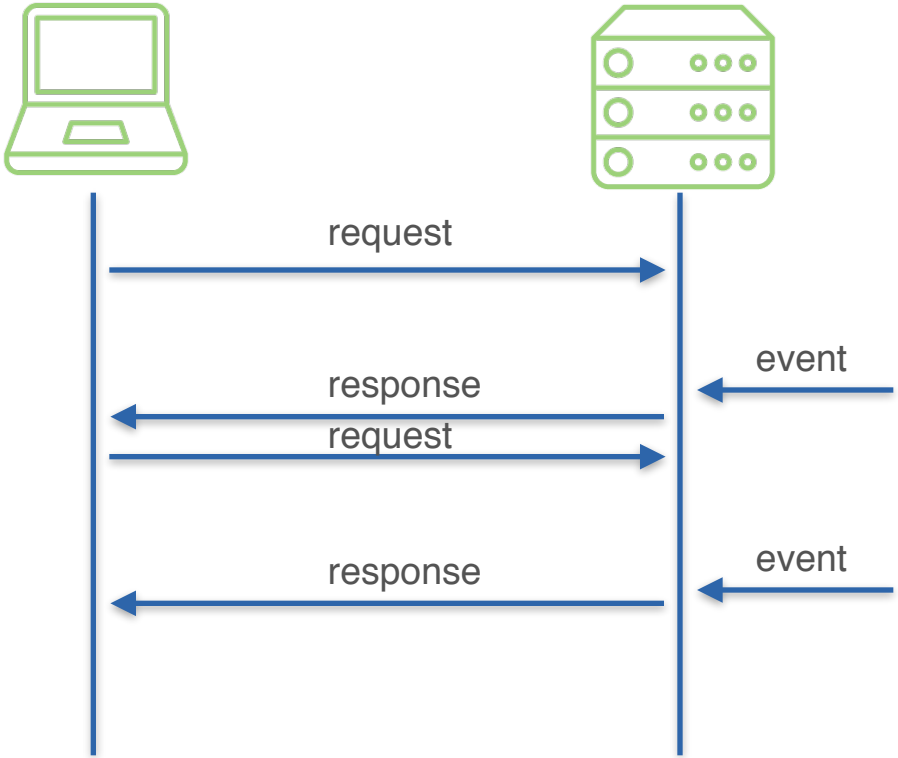


# Polling

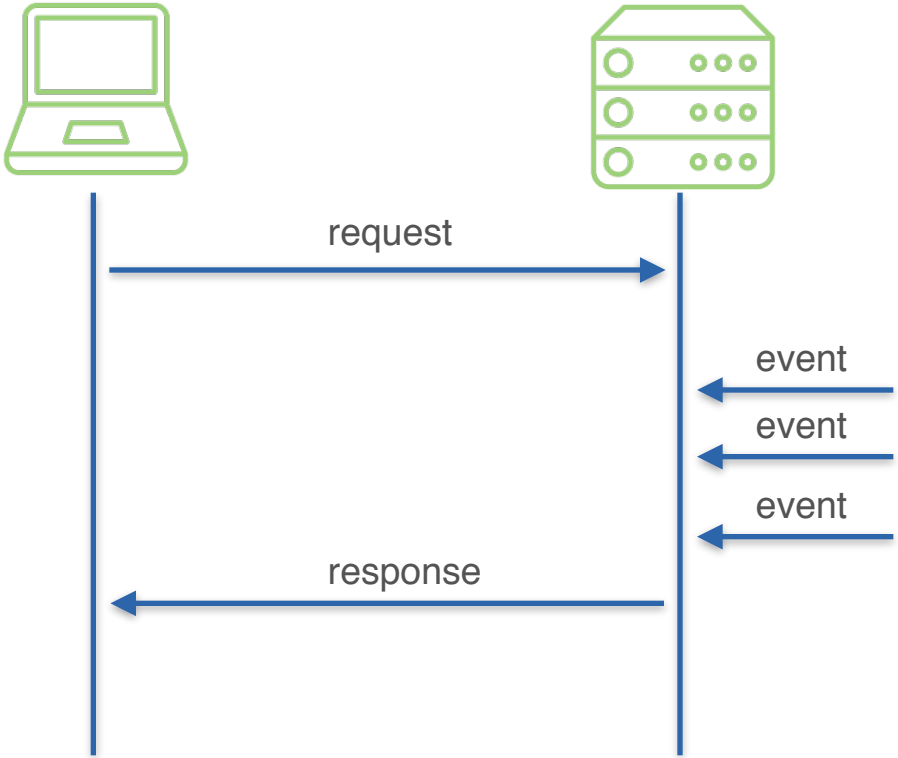
```
setInterval(function(){
        $.ajax({ url: "server", success: function(data){
                //Process
        }, dataType: “json"});
}, 30000);
```



# Long Polling



# HTTP Streaming

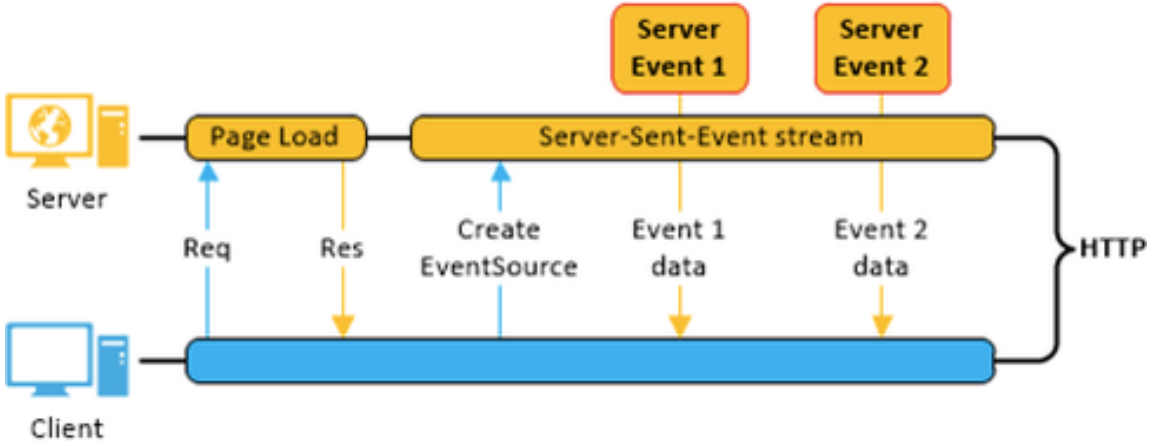


# Agenda

- Realtime Web
  - Evolution of the web
  - **Server Sent Events**
- WebSocket
  - Intro
  - Spring WebSocket
  - Scaling
  - Security

# Server Sent Events

- uni-directional
  - server push
- built on top of HTTP
  - a form of HTTP streaming
- long-loved HTTP connection
- EventSource API



# EventSource API

```
var source = new EventSource("/metrics");

source.addEventListener('memory', function(event) {
      console.log(event.data);
});

source.addEventListener('uptime', function(event) {
      console.log(event.data);
});
```

# Spring SSE

- return SseEmitter from method handler (since Spring 4.2)
- use **onCompletion**() to be notified when async request completes
  - also called when network errors occur

```
@RequestMapping("/metrics")
public SseEmitter subscribeMetrics() {
   SseEmitter emitter = new SseEmitter();

   // Save emitter for further usage
   return emitter;
}
```

# DEMO

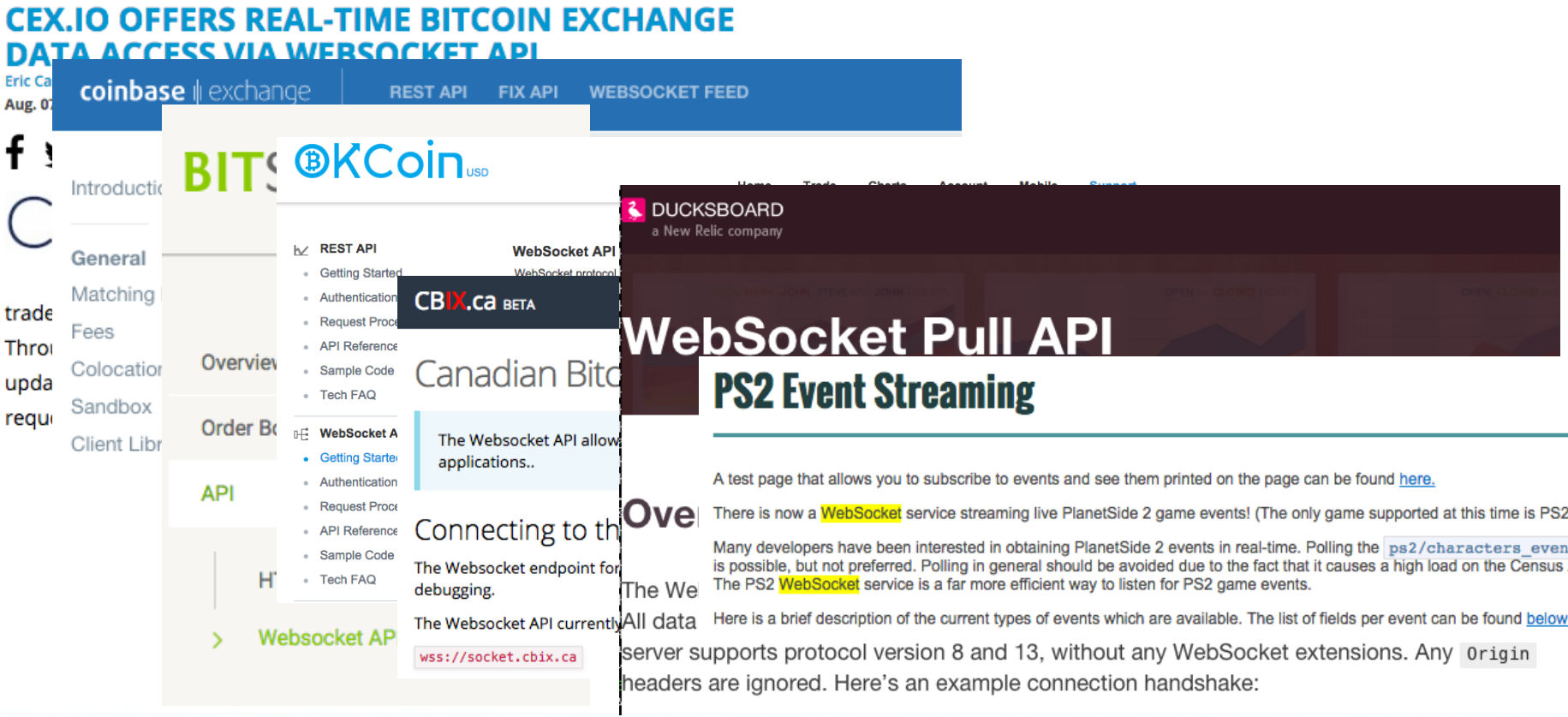
# Agenda

- Realtime Web
  - Evolution of the web
  - Server Sent Events
- WebSocket
  - **Intro**
  - Spring WebSocket
  - Scaling
  - Security

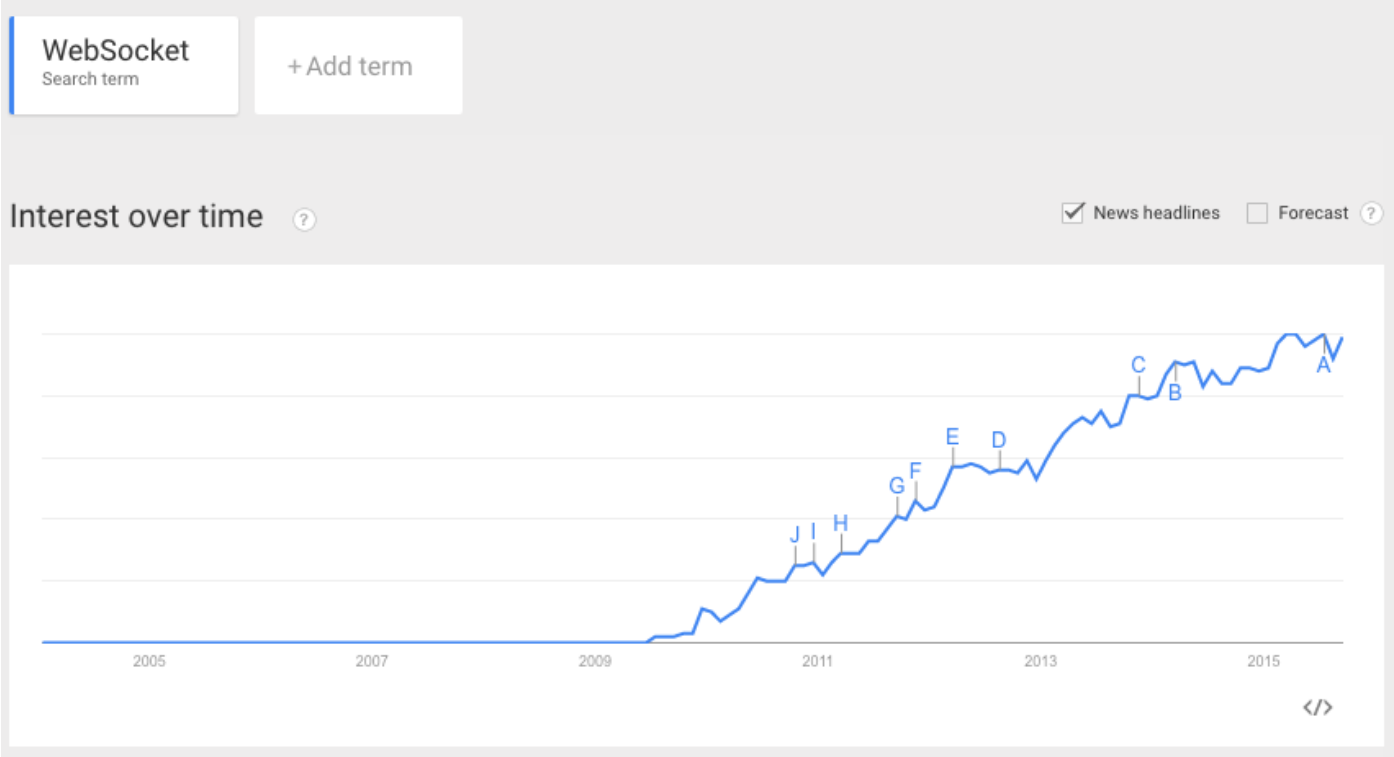
# WebSocket Protocol

- Real-time full duplex communication over TCP
  - Standardized by the IETF (RFC 6455)
- Uses port 80 / 443 (URL scheme ws:// wss://)
- Small overhead for text messages (frames)
  - 0x00 for frame start, 0xFF for frame end (vs HTTP 1Kb)
- Use cases: games, collaborative apps, financial tickets, social feeds, chat…

# Realtime WebSocket-based APIs



# WebSocket - Anyone interested?

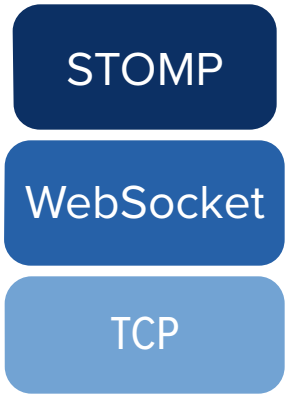


# WebSocket API


```
var ws = new WebSocket("ws://localhost/ws");

ws.onopen = function () {
    ws.send('Here I am!');
};

ws.onmessage = function (event) {
    console.log('message: ' + event.data);
};

ws.onclose = function (event) {
    console.log('closed:' + event.code);
};
```

# Subprotocols

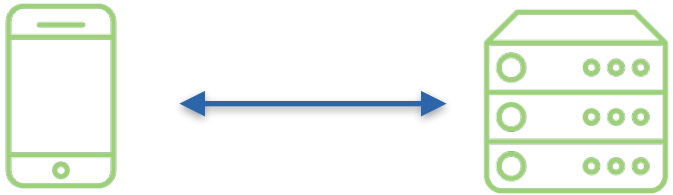
- WebSocket doesn't define any application protocol
  - As opposed to HTTP
- Too low level, applications need to interpret meaning of messages
- A subprotocol can be negotiated during handshake
  - STOMP, WAMP, MQTT, XMPP...
- Spring WebSocket supports STOMP



# WebSocket clients

- Not only browsers...

- Mobile clients

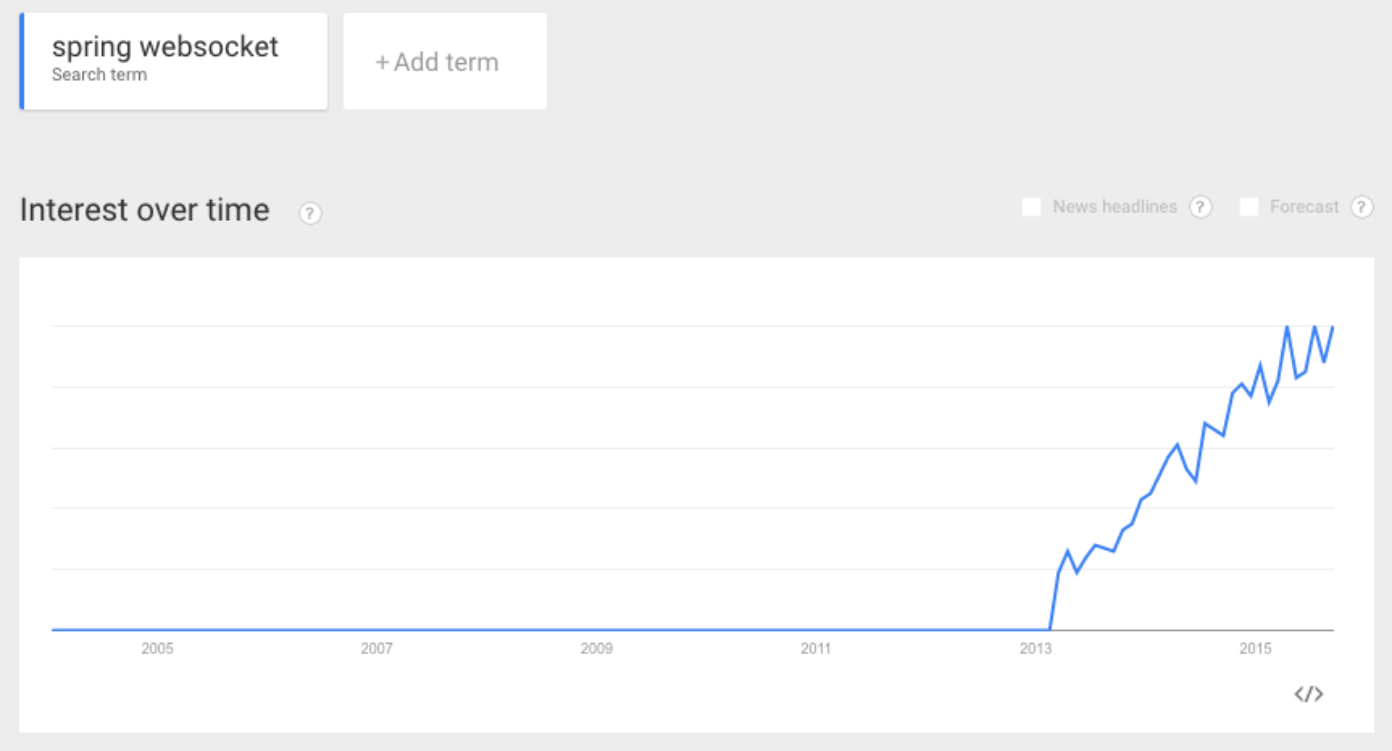


- Server to server communication
  - SockJS and STOMP clients available in Spring

# Agenda

- Realtime Web
  - Evolution of the web
  - Server Sent Events
- WebSocket
  - Intro
  - **Spring WebSocket**
  - Scaling
  - Security

# Spring WebSocket - Anyone interested?



# WebSocket Config

```
@Configuration
@EnableWebSocket
public class WebSocketConfig implements WebSocketConfigurer {

    @Override
    public void registerWebSocketHandlers(WebSocketHandlerRegistry registry) {
        registry.addHandler(echoHandler(), "/echo");
    }

    @Bean
    public EchoHandler echoHandler() {
        return new EchoHandler();
    }
}
```

# CODING TIME

# STOMP over WebSocket

- When using STOMP, we will have an event-driven, message architecture
  - Similar to JMS or AMQP
- Types of destinations
  - Application destinations
    - Messages routed to controller message handler methods
  - Broker destinations
    - Messages routed to the message broker
  - User destinations
    - Messages routed to a specific user

# Broker

- Two options:
  - SimpleBroker
    - built-in broker
    - everything in memory
  - BrokerRelay
    - Forwards messages to a STOMP broker (RabbitMQ, ActiveMQ...)
    - Better for scaling

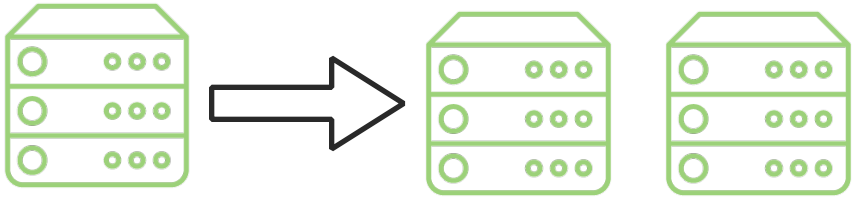
# CODING TIME

# Agenda

- Realtime Web
  - Evolution of the web
  - Server Sent Events
- WebSocket
  - Intro
  - Spring WebSocket
  - **Scaling**
  - Security

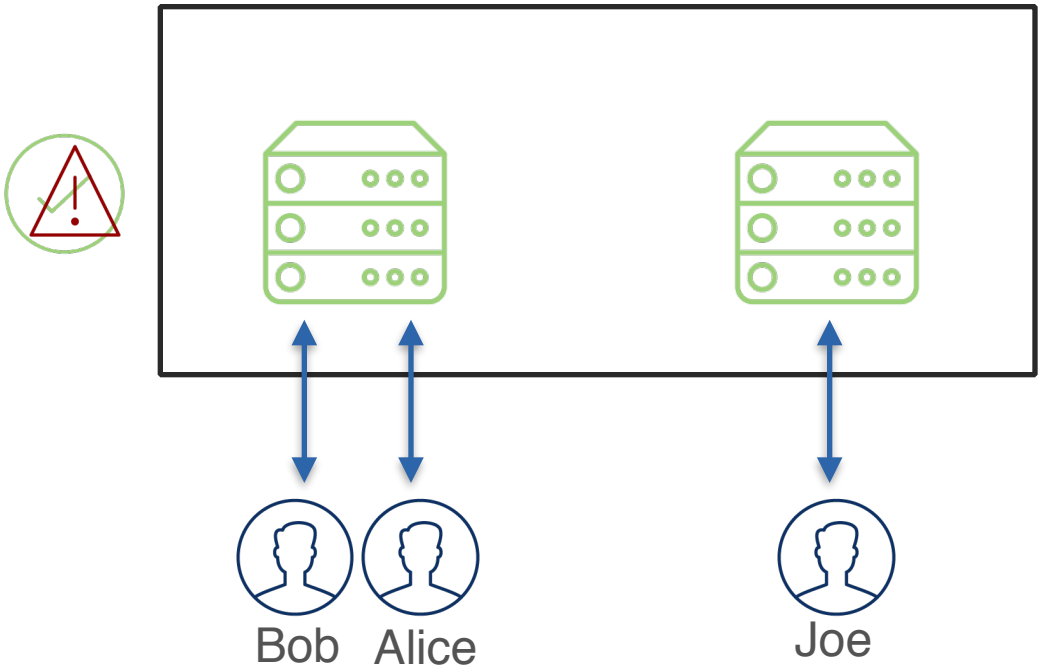
# Time to Scale

```
cf scale spring-questions -i 2
```



# Messaging users

**Bob** sends a message to **Alice**
**Alice** sends a message to **Joe**.
Instance1 does not resolve the user destination (*/user/alice/queue/messages*). As they are on the same instance, the user destination can be resolved



# Resolving user destinations

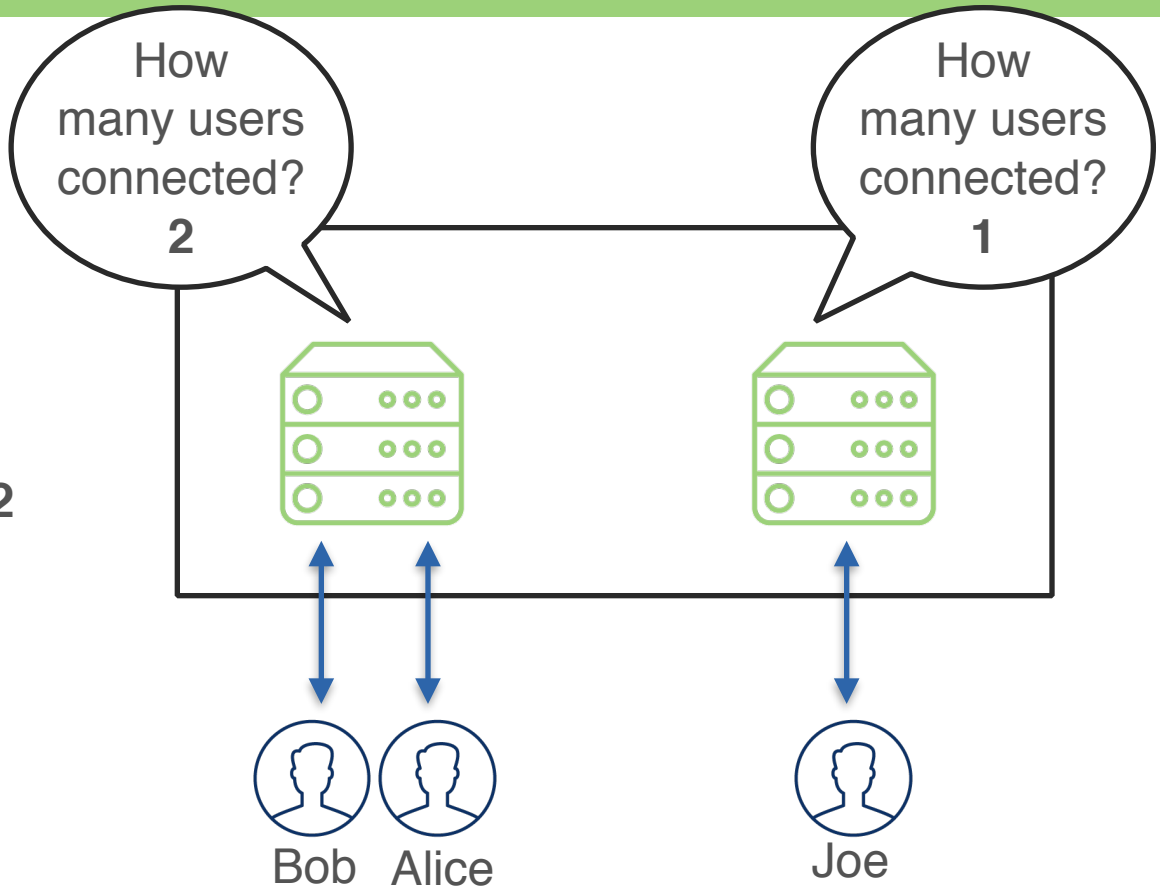
- When a user destination cannot be resolved, a message will be broadcasted
  - This allows other instances to resolve the destination

```
@Override
public void configureMessageBroker(MessageBrokerRegistry registry) {
      registry.setApplicationDestinationPrefixes("/app")
                  .enableStompBrokerRelay("/queue", "/topic")
                     .setUserDestinationBroadcast("/topic/unresolved-user-dest");
}
```

# Connected users

*Bob* and *Alice* are connected to **instance1**

*Joe* is connected to **instance2**



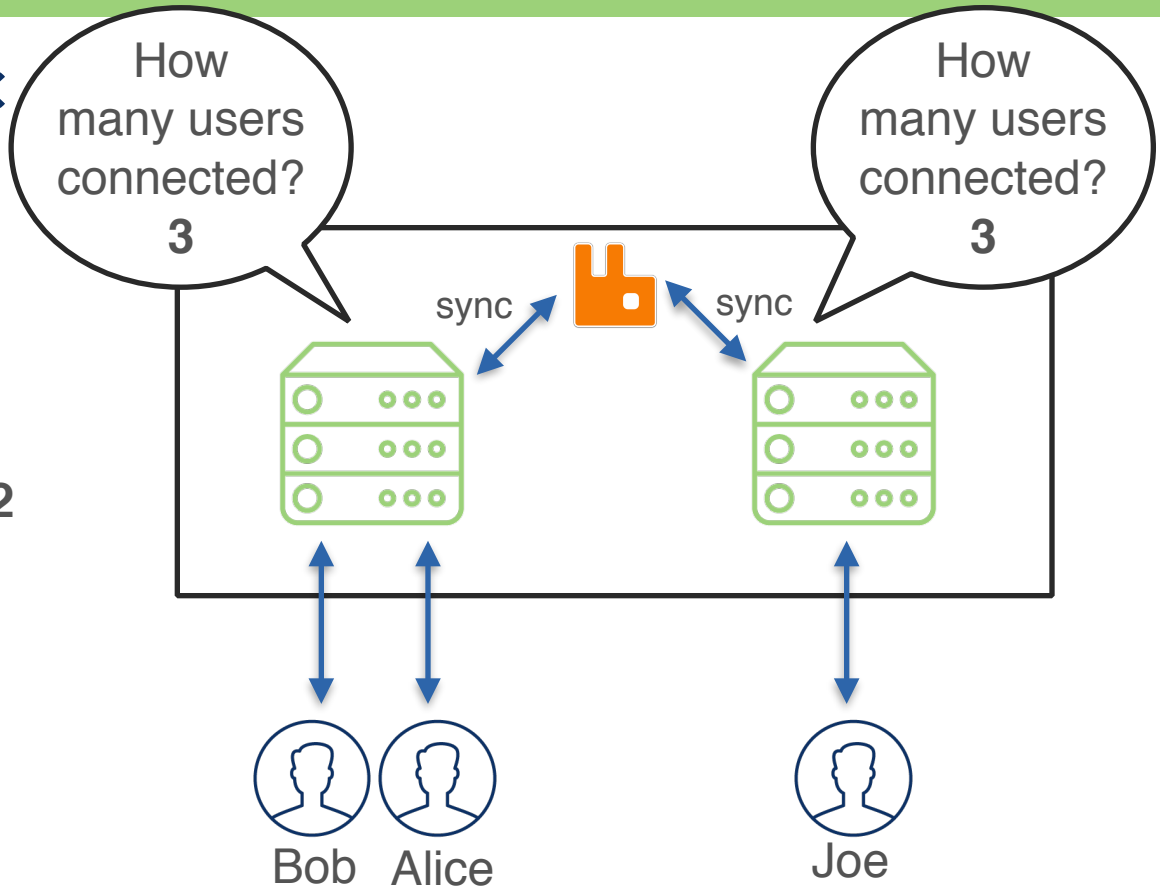
# SimpUserRegistry

- A registry for the currently connected users and subscriptions
- Two implementations:
  - **DefaultSimpUserRegistry** (default strategy)
    - stores everything in memory
  - **MultiServerUserRegistry**:
    - shares user registries across multiple servers

# UserRegistry Sync

*Bob* and *Alice* are connected to **instance1**

*Joe* is connected to **instance2**



# Resolving user destinations

- When a user destination cannot be resolved, a message will be broadcasted
  - This allows other instances to resolve the destination

```
@Override
public void configureMessageBroker(MessageBrokerRegistry registry) {
      registry.setApplicationDestinationPrefixes("/app")
                  .enableStompBrokerRelay("/queue", "/topic")
                     .setUserDestinationBroadcast("/topic/unresolved-user-dest");
}
```

# Agenda

- Realtime Web
  - Evolution of the web
  - Server Sent Events
- WebSocket
  - Intro
  - Spring WebSocket
  - Scaling
  - **Security**

# WebSocket Security

- Handshake request is a simple HTTP request
  - Protect it as a normal URL
- Use WebSocket Secure connection (wss://)
- Origin

# Message Security

```
@Configuration
public class WebSocketSecurityConfig extends
AbstractSecurityWebSocketMessageBrokerConfigurer {

        @Override
        protected void configureInbound(MessageSecurityMetadataSourceRegistry messages) {
                messages
                        // restrict subscription with role ADMIN
                        .simpSubscribeDestMatchers("/topic/admin.notifications").hasRole("ADMIN")
                        // users cannot send to these broker destinations, only the application can
                        .simpMessageDestMatchers("/topic/orders", "/topic/order.conf").denyAll()
                        .anyMessage().authenticated();
        }
}
```

SPRINGONE2GX
WASHINGTON, DC

Learn more
http://www.infoq.com/presentations/spring-4-websockets

 @sergialmar

 salmar

Q & A

Thanks!

See you at Spring I/O 2016
Barcelona, May 19-20

spring I/O
www.springio.net