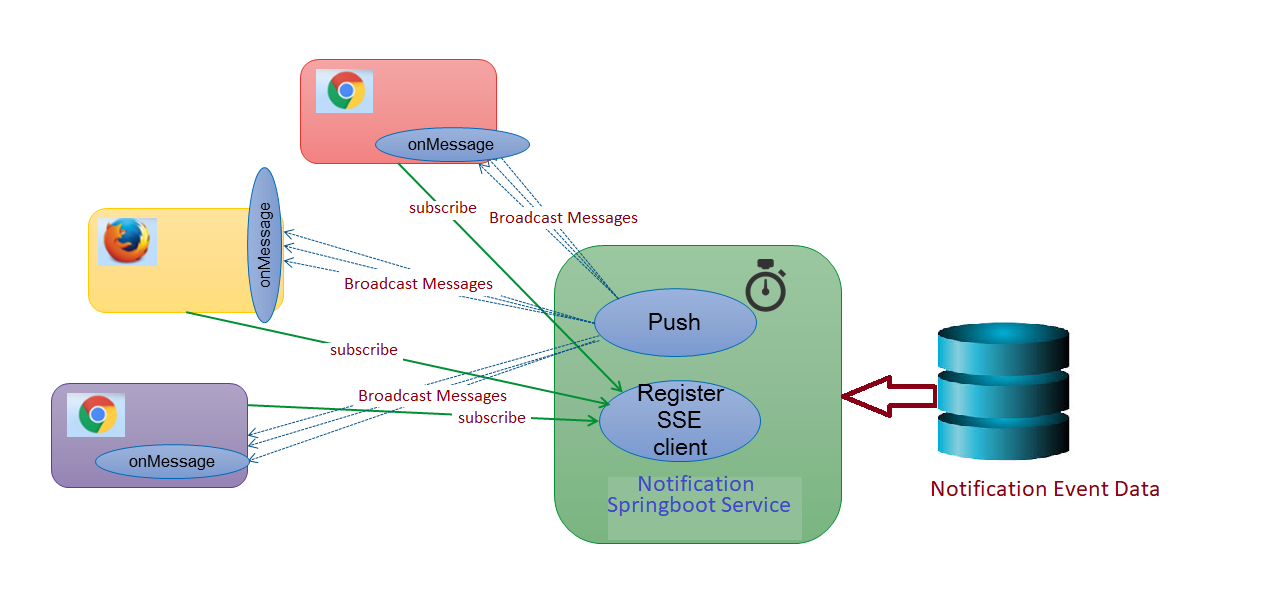
**Server-Sent Events (SSE) – Push Notification**

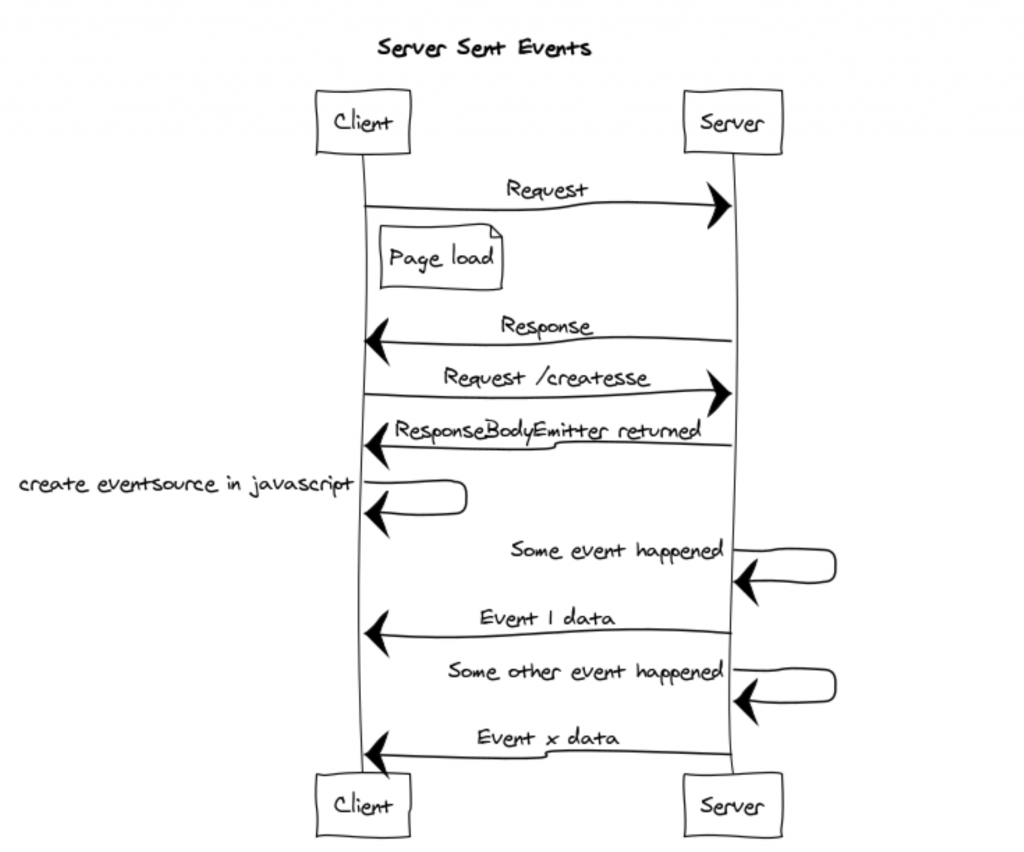
In-order to send data from server to client or vice versa, we have Polling, WebSockets & Server-Sent event mechanisms.

Among these WebSocket opens bidirectional connections between server and client. So, both server and client can send messages. In situations, where the application needs only one way communication, i.e., sending data from server to the client and for this Spring provides a simpler solution using Server Sent Events (SSE). SSE is a technology that allows you to stream data from server to the browser (Push Notifications) within one HTTP connection in one direction.

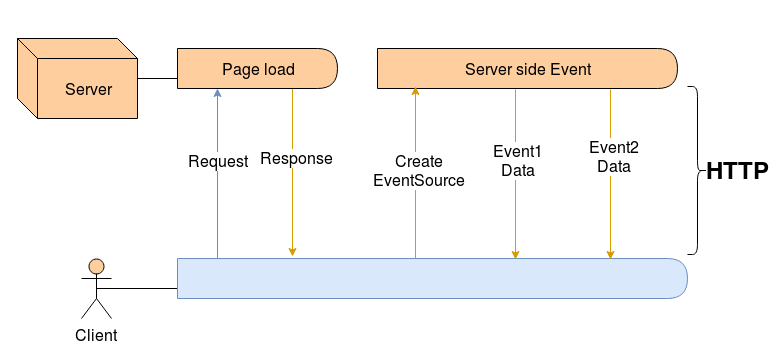
Example: pushing stock price changes in real-time or real time showing of cricket or football scores on display board etc.



High level Architecture of SSE with spring boot service



Client server view of SSE events flow



Eventsource creation of subscribed clients to receive notifications

**Implementation Approach\Solution using Springboot**

**SSE Emitter -** Spring has inbuilt support for SSE in the form of SseEmitter. Its a specialization of [ResponseBodyEmitter](https://howtodoinjava.com/spring-boot2/spring-async-controller-responsebodyemitter/) for sending Server-Sent Events.

Server-Sent-Events are messages from the server to the client. They have a Content-Type header of *text/event-stream*.

The events are simple and have only four fields.

* Id- The ID of the event
* Event-the type of event
* Data-The event data
* Retry- Reconnection time for the event stream

1. **Creating SSE events**

*OutboundSseEvent.Builder* - allows us to create events

A scheduler bean will be created to repetitively look for any new notification events available in database to create events thro’ the notification service.

*OutboundSseEvent.Builder* has support for all the method to pass event parameters

1. **Publish \ Broadcasting Events**

Sending an event is then as simple as invoking *SseEventSink.send()*

SSE Broadcaster API – allows to send notifications to multiple subscribers simultaneously

It can be done in three simple steps,

* create a SseBroadcaster object from an injected Sse context

SseBroadcaster sseBroadcaster = sse.newBroadcaster();

* Then, clients subscribed should be able to receive Sse Events. This is generally done in an SSE resource method where a SseEventSink context instance is injected
* And finally, we can trigger the event publishing by invoking the broadcast() method

1. **Consuming SSE events**

The client has to define and configure [EventSource](https://developer.mozilla.org/en-US/docs/Web/API/EventSource) & it is supported in all browser otherwise ajax request can be made to receive notifications on client devices. On client side, an application interacts with the [EventSource](https://developer.mozilla.org/en/docs/Web/API/EventSource) object & it is responsible for sending the SSE request to the server and calling listeners to the application registered on this object. The client must send an id that should be unique among all the clients.

**Advantages**

* The main benefits we get from this approach are:
* Data efficient
* Simpler implementation
* It is automatically multiplexed over HTTP/2
* Limits the number of connections for data on the client to one