



Alma Mater Studiorum Università di Bologna

Scuola di Ingegneria

Tecnologie Web T
A.A. 2022 – 2023

Esercitazione 07 - WebSocket

Home Page del corso:

<http://lia.disi.unibo.it/Courses/twt2223-info/>

Versione elettronica: L.07.WebSocket.pdf

Versione elettronica: L.07.WebSocket-2p.pdf

Limiti HTTP «tradizionale» e Web Socket

Prima soluzione proprietaria integrata Javascript in alcuni browser (Google Chrome) e poi supportata da specifici Web server. Poi successo e sforzo di standardizzazione...

- ☐ Protocollo Web Socket (basato su TCP/IP) – RFC 6455
- ☐ Integrazione di Web Socket in HTML5 (via Javascript) e in JEE (a partire da v7)
- ☐ Web Socket API per Java definite in JSR 356

Perché? Limiti del modello di interazione HTTP quando vogliamo usare HTTP per comunicazione 2-way:

- ☐ Polling
- ☐ Long polling
- ☐ Streaming/forever response
- ☐ Connessioni multiple

Web Socket: principali caratteristiche

❑ Bi-direzionali

- Client e server possono scambiarsi messaggi quando desiderano

❑ Full-duplex

- Nessun requisito di interazione solo come coppia request/response e di ordinamento messaggi

❑ Unica connessione long running

❑ Visto come «upgrade» di HTTP

- Nessuno sfruttamento di protocollo completamente nuovo, nessun bisogno di nuova «infrastruttura»

❑ Uso efficiente di banda e CPU

- Messaggi possono essere del tutto dedicati a dati applicativi

Lato Server, a partire da JEEv7

```
@ServerEndpoint("/actions")
public class WebSocketServer {

    @OnOpen
    public void open(Session session) { ... }

    @OnClose
    public void close(Session session) { ... }

    @OnError
    public void onError(Throwable error) { ... }

    @OnMessage
    public void handleMessage(String message, Session session) {
        // actual message processing
    }
}
```

Lato browser cliente, integrazione Javascript

```
var socket = new WebSocket("ws://server.org/  
    wsendpoint");  
socket.onmessage = onMessage;  
  
function onMessage(event) {  
    var data = JSON.parse(event.data);  
    if (data.action === "addMessage") {  
        ...  
        // actual message processing  
    }  
    if (data.action === "removeMessage") {  
        ...  
        // actual message processing  
    }  
}
```

Esercizio 1 - Calcolatrice

Si realizzi una semplice calcolatrice con le classiche operazioni aritmetiche (due caselle di input per operandi e 4 bottoni per le operazioni di addizione, sottrazione, moltiplicazione e divisione)

- ❑ Lato cliente si deve verificare che quanto inserito nei campi di input possano essere operandi validi (***controllo convertibilità in numero***)
- ❑ Sempre lato cliente, ***trasformazione dei dati in ingresso in formato JSON*** per trasporto verso servitore
- ❑ (opzionale) si verifichi che un utente non possa effettuare più di 100 richieste di servizio all'interno della sua sessione di interazione

Esercizio 1 - Calcolatrice

Si realizzi una semplice calcolatrice con le classiche operazioni aritmetiche (due caselle di input per operandi e 4 bottoni per le operazioni di addizione, sottrazione, moltiplicazione e divisione)

- ❑ Lato servitore si devono effettuare TUTTI i calcoli
- ❑ Lato servitore ***si mantiene risultato dell'ultima operazione*** eseguita con successo da ogni cliente
- ❑ Dati di ingresso al servitore e risultato dell'operazione svolta da restituire al cliente devono essere ***scambiati in formato JSON***
- ❑ Nessuna info di stato complessiva deve essere visibile a tutti gli utenti dell'applicazione

Esercizio 2 – Calcolatrice Collaborativa

Si trasformi la calcolatrice precedente in una «versione collaborativa» dove ogni utente può inserire operandi e premere pulsanti di operazioni in ogni momento

☐ **Uso rilevante di WebSocket**

☐ Se un cliente inserisce un operando, ogni volta che un carattere dell'operando è digitato **la casella di input deve aggiornarsi su tutti i clienti collegati**, come in una lavagna condivisa

☐ Il **risultato dell'ultima operazione** eseguita con successo deve essere visibile a tutti e condiviso fra tutti

☐ (opzionale) dopo 5 minuti dall'inizio dell'interazione il server deve decidere di terminare l'interazione e mandare al cliente un refresh della pagina con frase «Sessione terminata!»