

COMP 504: Graduate Object-Oriented Programming and Design

Lecture 19: Web Sockets

Mack Joyner (mjoyner@rice.edu)

<https://www.clear.rice.edu/comp504>



Announcements & Reminders

Hw #3 due **today** at 11:59pm

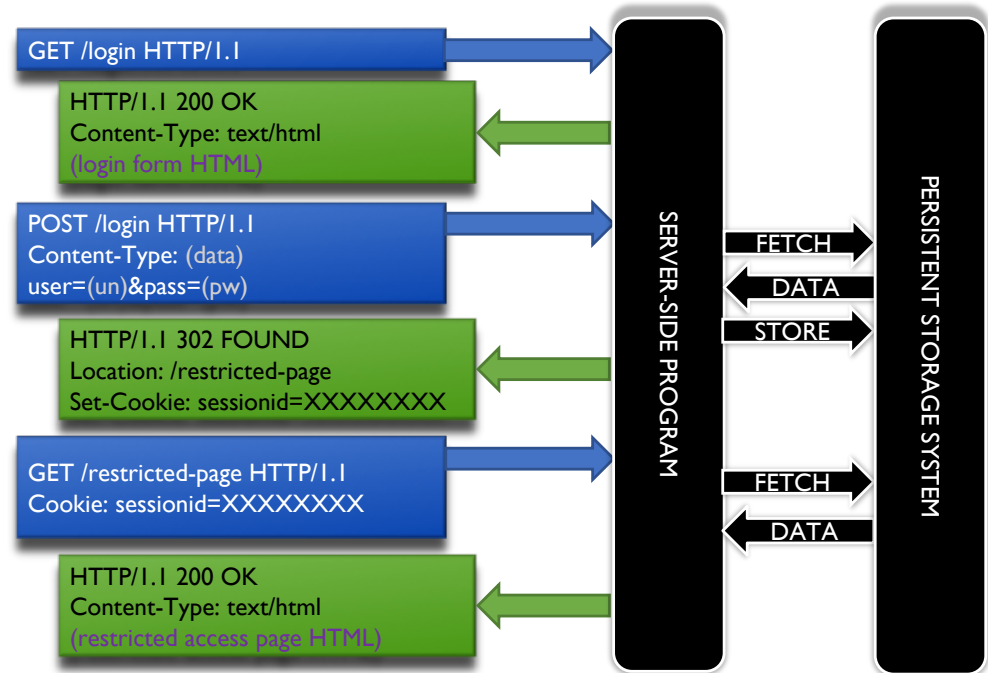
Quiz #3 (Command, Visitor, Strategy) due Wed. Oct. 14th at 11:59pm

Hw #4 available later today, due Wed. Oct. 21st at 11:59pm



Http Requests

- Initiated by the browser
- One-way communication
- Requires polling by client for server updates



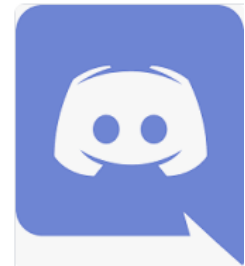
Chat Apps



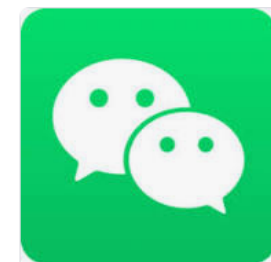
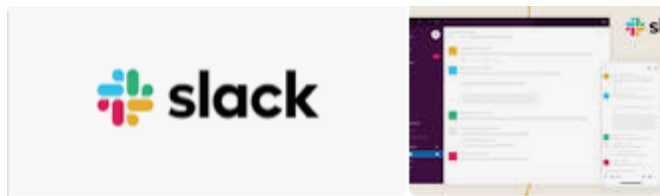
WhatsApp



Facebook Messenger

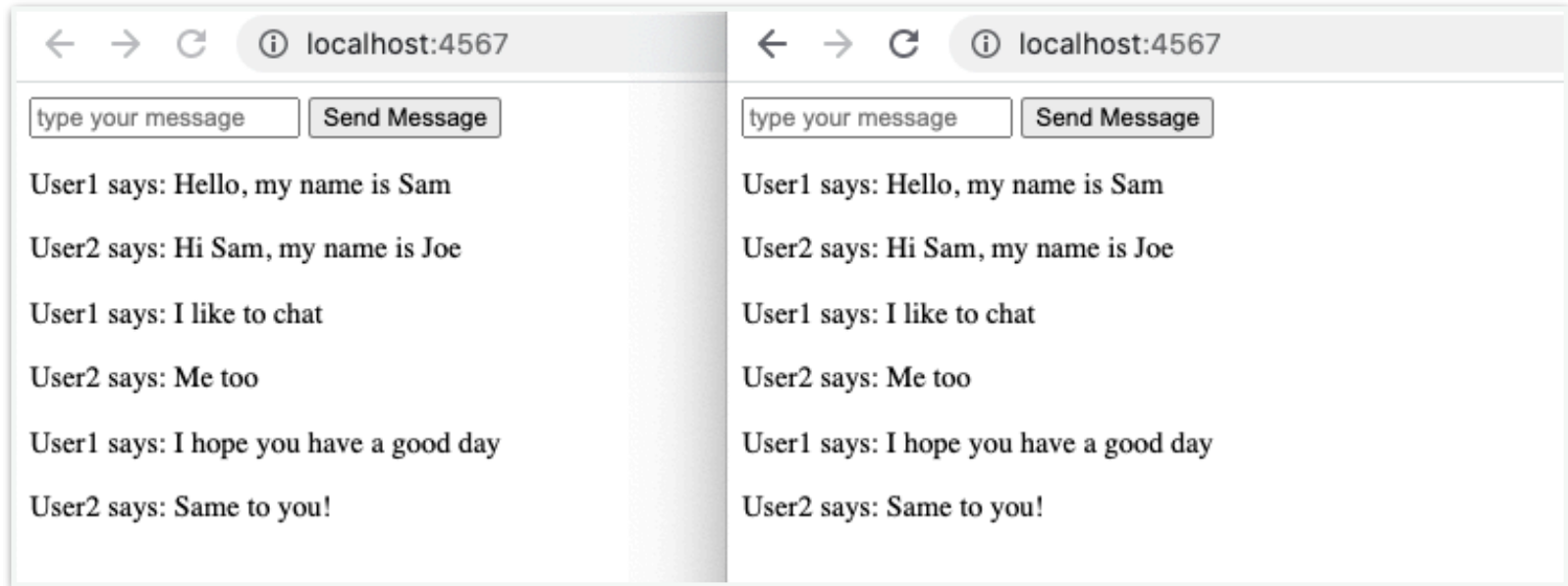


Discord



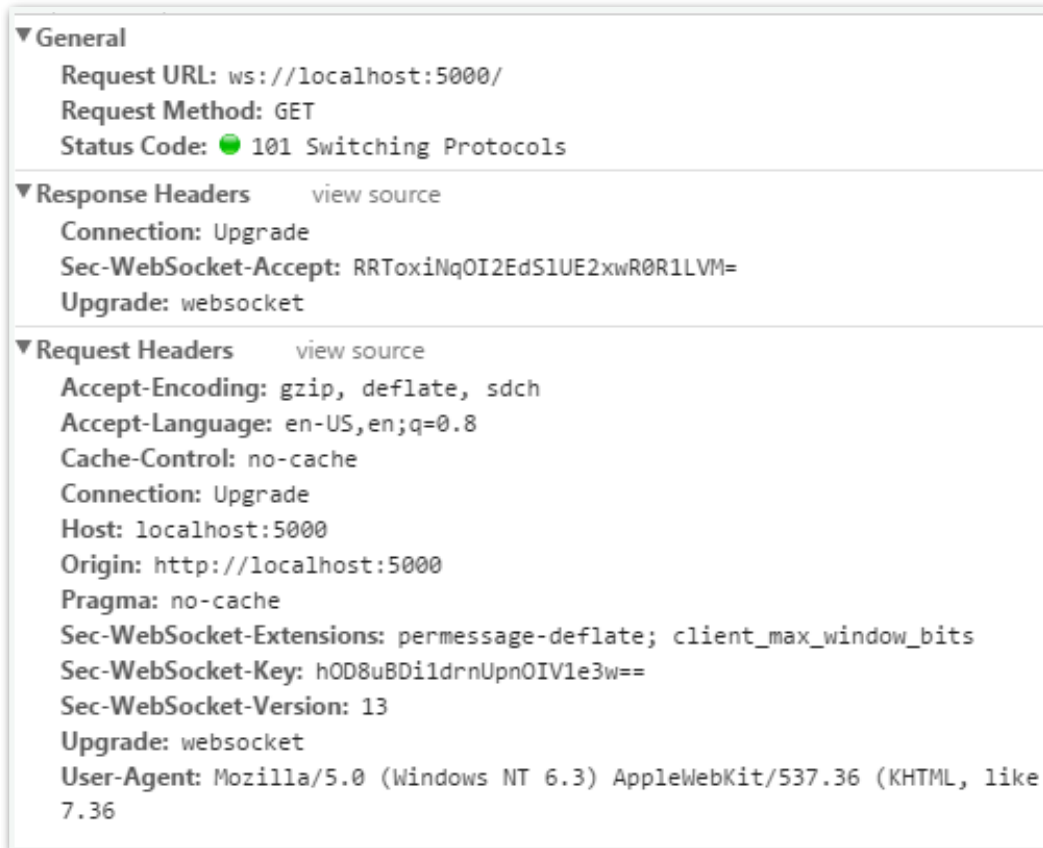
WeChat

Two-Way Chat Communication



Web Sockets

- Two-way pipes
- Ideal for chatty communication



The screenshot displays a browser's developer console with three expanded sections: General, Response Headers, and Request Headers. The General section shows a GET request to ws://localhost:5000/ with a 101 status code. The Response Headers section shows the upgrade to websocket. The Request Headers section lists various headers including Accept-Encoding, Accept-Language, Cache-Control, Connection, Host, Origin, Pragma, Sec-WebSocket-Extensions, Sec-WebSocket-Key, Sec-WebSocket-Version, Upgrade, and User-Agent.

```
▼ General
  Request URL: ws://localhost:5000/
  Request Method: GET
  Status Code: 101 Switching Protocols

▼ Response Headers view source
  Connection: Upgrade
  Sec-WebSocket-Accept: RRToxiNqOI2EdS1UE2xwR0R1LVM=
  Upgrade: websocket

▼ Request Headers view source
  Accept-Encoding: gzip, deflate, sdch
  Accept-Language: en-US,en;q=0.8
  Cache-Control: no-cache
  Connection: Upgrade
  Host: localhost:5000
  Origin: http://localhost:5000
  Pragma: no-cache
  Sec-WebSocket-Extensions: permessage-deflate; client_max_window_bits
  Sec-WebSocket-Key: hOD8uBDi1drnUpnOIV1e3w==
  Sec-WebSocket-Version: 13
  Upgrade: websocket
  User-Agent: Mozilla/5.0 (Windows NT 6.3) AppleWebKit/537.36 (KHTML, like 7.36
```



Creating a Web Socket on Server

- Annotations are used instead of implementing WebSocketListener
- onConnect: called when navigating to app
 - Once for each client instance
 - Unique session for each client
 - Establish user id for client
- onClose: called when closing the web socket
 - Terminating the server
 - Remove user id for client
- onMessage: called when sent a message from a client instance

```
@WebSocket
public class WebSocketController {

    /**...*/
    @OnWebSocketConnect
    public void onConnect(Session user) {...}

    /**...*/
    @OnWebSocketClose
    public void onClose(Session user, int statusCode, String reason) {...}

    /**...*/
    @OnWebSocketMessage
    public void onMessage(Session user, String message) {...}
}
```



Chat App Using Web Socket

- Main controller, starts the web socket for the server
- Specify the endpoint for the web socket
- Connect to a defined web socket on the server side
- Init to begin listening for messages
 - Immediately finishes without *init()*

```
/**
 * The chat app controller communicates with all the clients on the web socket.
 */
public class ChatAppController {
    static Map<Session,String> userNameMap = new ConcurrentHashMap<>();
    static int nextUserId = 1;

    /**
     * Chat App entry point.
     * @param args The command line arguments
     */
    public static void main(String[] args) {
        port(getHerokuAssignedPort());
        staticFiles.location( folder: "/public");

        webSocket( path: "/chatapp", WebSocketController.class);
        init();
    }
}
```



Send a Message to Client(s)

- Send message to client with user *session*
- Pass string representation of JSON object to client
- On client side, parse string as JSON
 - enables client to select object fields

```
JsonObject jo = new JsonObject();  
session.getRemote().sendString(String.valueOf(jo));
```



Create a Web Socket on Client

- Create a Web Socket on the client side (view.js)
- Use web socket schema (ws://)
- URL including endpoint (/chatapp) should match server endpoint to establish two-way communication



Create a Web Socket on Client

```
const websocket = new WebSocket("ws://" + location.hostname + ":" + location.port + "/chatapp");
```



Send a Message to Server

- Communicate with server by calling web socket *send*
 - pass message as an argument
- Web socket has other useful functions that mirror server web socket
 - *onConnect*
 - *onClose*
 - *onMessage*

```
/**  
 * Send a message to the server.  
 * @param msg The message to send to the server.  
 */  
function sendMessage(msg) {  
    if (msg !== "") {  
        websocket.send(msg);  
        $("#message").val("");  
    }  
}
```



Live Demo: Chat with Web Sockets



Worksheet #13: Web Sockets

Assume there's a Web Socket variable *WebSocket*.

1. Write code in the client that sends an alert message “*Web Socket connection closed*” when the connection closes.
2. What client method is called when the server sends a message?

