

# cse15l-lab-reports

---

## Part 1: ChatServer

---

### ChatServer.java Code

```
import java.io.IOException;
import java.net.URI;

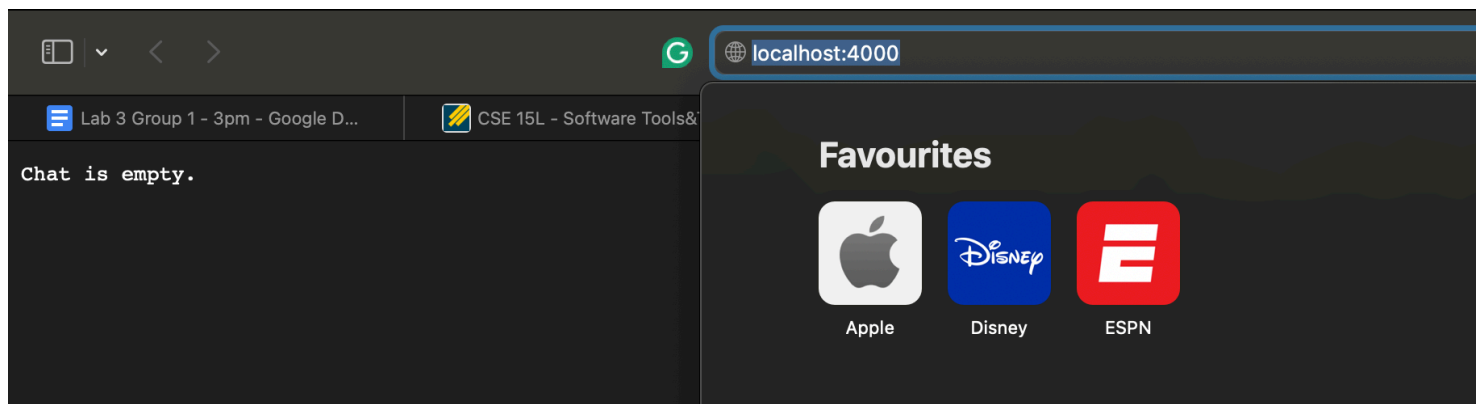
class Handler implements URLHandler {
    private StringBuilder chatLog = new StringBuilder();

    public String handleRequest(URI url) {
        if (url.getPath().equals("/")) {
            return chatLog.length() == 0 ? "Chat is empty." : chatLog.toString();
        } else if (url.getPath().equals("/add-message")) {
            String query = url.getQuery();
            String user = null;
            String message = null;
            for (String param : query.split("&")) {
                String[] entry = param.split("=");
                if (entry.length > 1) {
                    if (entry[0].equals("user")) {
                        user = entry[1];
                    } else if (entry[0].equals("s")) {
                        message = entry[1];
                    }
                }
            }
            if (user != null && message != null) {
                chatLog.append(user).append(": ").append(message).append("\n");
                return chatLog.toString();
            } else {
                return "Invalid request! Parameters 's' and 'user' are required.";
            }
        }
        return "404 Not Found!";
    }
}
```

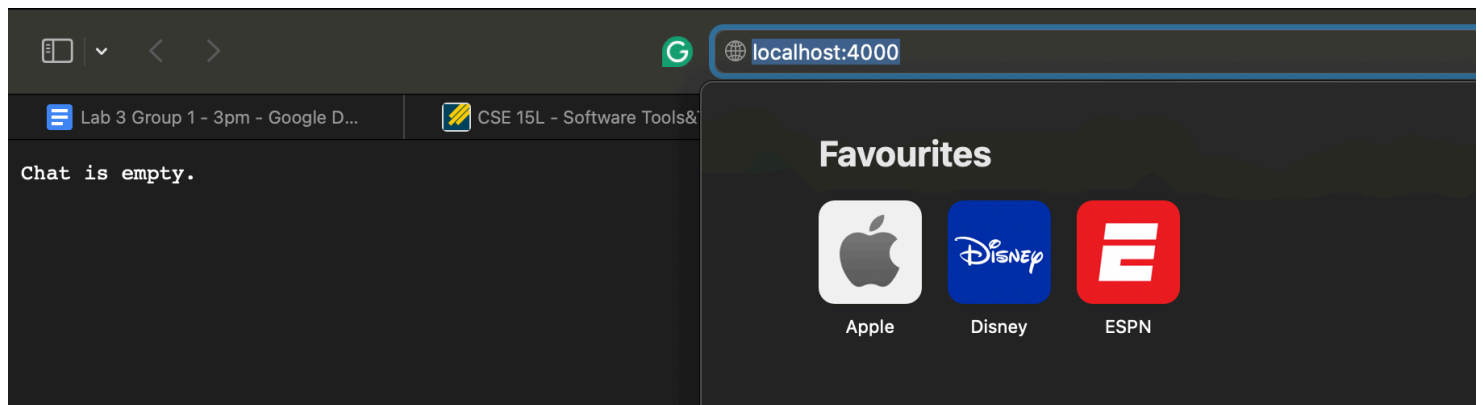
```
}  
}  
  
class ChatServer {  
    public static void main(String[] args) throws IOException {  
        if (args.length == 0) {  
            System.out.println("Missing port number! Try any number between 1024 t  
            return;  
        }  
        int port = Integer.parseInt(args[0]);  
        Server.start(port, new Handler());  
    }  
}
```

## Demo

### Before:



### After:



**Methods Called:** `handleRequest(Uri url)` - The URL from the request is parsed to determine

the path and query parameters.

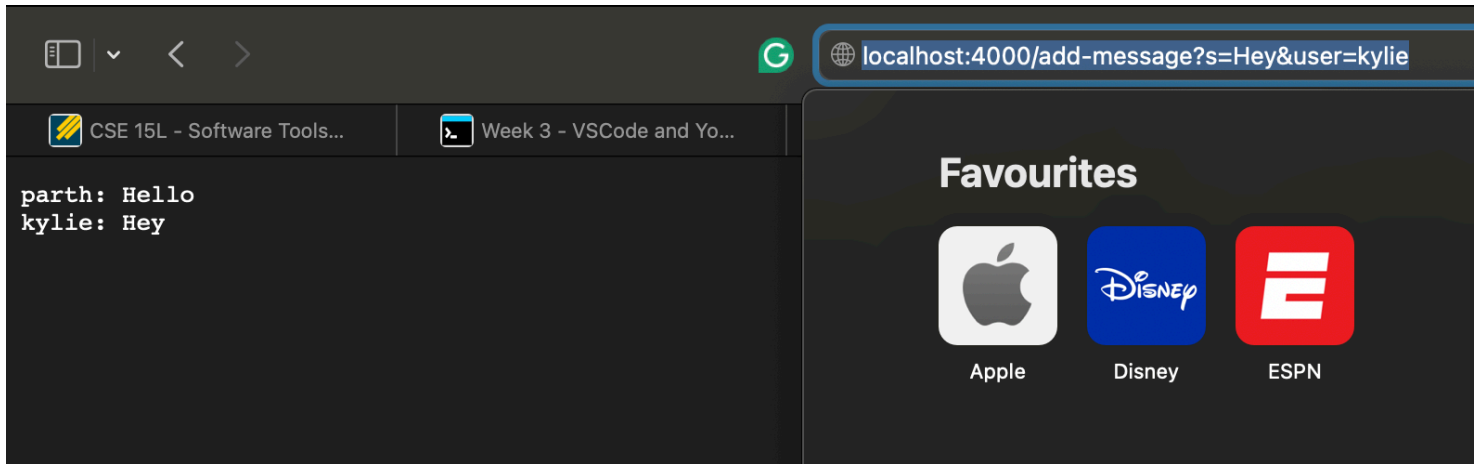
## Arguments and Values of Fields

- `url (URI)`: An instance of `URI` , representing the URL `http://localhost:4000/add-message?s=Hello&user=parth` .
- `chatLog (StringBuilder)`: This field in the `Handler` class is used to accumulate chat messages as they are received. Initially, it is an empty `StringBuilder` .

**How Field Values Change** Here is the flow and changes in the field values step-by-step, given the request `http://localhost:4000/add-message?s=Hello&user=parth` :

1. Checking the path:
  - The `getPath()` method on the `URI` instance returns `/add-message` , matching one of the conditions in `handleRequest` .
2. Parsing the Query:
  - The `getQuery()` method on the `URI` instance returns the string `"s=Hello&user=parth"` .
  - The query string is split into parameters: `["s=Hello", "user=parth"]` .
  - Each parameter is further split into key-value pairs: `["s", "Hello"]` and `["user", "parth"]` .
3. Extracting Parameters:
  - From the splits, `user` is assigned the value `"parth"` and `message` is assigned `"Hello"` .
4. Appending to `chatLog`:
  - Given both `user` and `message` are non-null, the message `"parth: Hello\n"` is appended to `chatLog` .
  - Before the request, `chatLog` was an empty `StringBuilder` . After the request, it contains the string `"parth: Hello\n"` .

## 2nd Screenshot



**Methods Called:** `handleRequest(Uri url)` - This method is called again when the new request arrives. The purpose remains to parse the URI and handle the request based on its path and parameters.

### Arguments and Values of Fields

- `url (Uri)`: Represents the URL `http://localhost:4000/add-message?s=Hey&user=kylie`.
- `chatLog (StringBuilder)`: Before this request, `chatLog` contained `"parth: Hello\n"`.

### How Field Values Change

#### 1. Checking the Path:

- The `getPath()` method on the URI returns `/add-message`, which matches the corresponding condition in `handleRequest`.

#### 2. Parsing the Query:

- The `getQuery()` returns `"s=Hey&user=kylie"`.
- The string is split into parameters: `["s=Hey", "user=kylie"]`.
- These are then split into key-value pairs: `["s", "Hey"]` and `["user", "kylie"]`.

#### 3. Extracting Parameters:

- `user` is assigned the value `"kylie"` and `message` is assigned `"Hey"`.

#### 4. Appending to chatLog:

- The `chatLog` already contains `"parth: Hello\n"`. After extracting the parameters, the new message `"kylie: Hey\n"` is appended.
- Therefore, `chatLog` changes from `"parth: Hello\n"` to `"parth: Hello\nkylie: Hey\n"`.

## Part 2: SSH and Remote Servers

Path to Private Key on local device

```
[parthshinde@Parths-MacBook-Air-3 ~ % ls ~/.ssh
config          id_ed25519.pub  known_hosts
id_ed25519      id_rsa          known_hosts.old
[parthshinde@Parths-MacBook-Air-3 ~ % ls ~/.ssh/id_ed25519
/Users/parthshinde/.ssh/id_ed25519]
```

Path to Public Key on ieng6 device

```
[[pshinde@ieng6-202]:~:11$ ls .ssh/authorized_keys
.ssh/authorized_keys
[pshinde@ieng6-202]:~:12$ █
```

Logging into ieng6 without a Password

```
Last login: Mon Apr 22 13:33:33 on ttys002
parthshinde@Parths-MacBook-Air-3 ~ % ssh pshinde@ieng6.ucsd.edu
Last login: Mon Apr 22 13:40:51 2024 from 100.83.41.80
quota: Cannot resolve mountpoint path /home/linux/staff/.snapshot/daily.2024-04-02_0010: Stale file handle
quota: Cannot resolve mountpoint path /home/linux/staff/.snapshot/hourly.2024-04-05_0801: Stale file handle
Hello pshinde, you are currently logged into ieng6-202.ucsd.edu

You are using 0% CPU on this system

Cluster Status

| Hostname  | Time     | #Users | Load             | Averages |
|-----------|----------|--------|------------------|----------|
| ieng6-201 | 13:45:01 | 14     | 1.17, 0.68, 0.61 |          |
| ieng6-202 | 13:45:01 | 12     | 0.28, 0.35, 0.29 |          |
| ieng6-203 | 13:45:01 | 9      | 1.85, 1.71, 1.50 |          |



To begin work for one of your courses [ cs15lsp24 ], type its name
at the command prompt. (For example, "cs15lsp24", without the quotes).

To see all available software packages, type "prep -l" at the command prompt,
or "prep -h" for more options.
[pshinde@ieng6-202]:~:14$ █
```

## Part 3: Reflection

In week 2 I learned how to log into a remote server using `ssh` and how to download things from an URL using `curl`. I also learned about starting and running servers. In week 3, I learned about private/public keys and how they could be use to login without requiring passwords.

