컴퓨터 네트워크 Assignment 2 - WIKI

- 이름 : 문원찬
- 학번 : 2019008813

---

1. Design & Implement

- 각 함수의 완전한 코드는 `Main2.java` 파일을 참고

사용한 패키지

```
// 입출력, 파일, 인코딩 관련 패키지
import java.io.*;
import java.nio.file.*;
import java.util.*;

// TCP 통신을 위한 socket 관련 패키지
import java.net.ServerSocket;
import java.net.Socket;

// JSON 파싱 관련 패키지
import com.google.gson.*;
```

main 함수

```
public static void main(String[] args) {
        int port = 8080;
        try {
                // 서버 소켓 생성
                ServerSocket serverSocket = new ServerSocket(port);
                System.out.println("Listening in port: " + port);
                while (true) {
                        // 클라이언트로부터 연결을 수락하고 클라이언트 소켓 받음
                        Socket clientSocket = serverSocket.accept();
                        handleRequest(clientSocket);
                }
        } catch (IOException e) {
                e.printStackTrace();
        }
}
```

로컬호스트 8080포트로 서버 소켓을 생성하고 `"Listening in port: 8080"` 로그를 남긴다.
클라이언트가 요청을 보낼 때까지 기다리다 요청이 오면 수락하고 통신용 소켓을 생성한다.

이때 통신용 소켓을 통해 받은 요청을 [handleRequest 함수]를 통해 응답하는 구조로 TCP 통신을 구현했다.

요청 핸들링 함수

```
private static void handleRequest(Socket clientSocket) {
        // 클라이언트로부터의 요청을 읽기
        // 요청 메서드와 요청 URI 추출
        // 쿠키 체크
        // GET 요청에 대한 응답
        // 지원하지 않는 HTTP 메서드에 대한 405 응답
}
```

- 입력 : 통신용 소켓(Socket)
  해당 함수는 클라이언트의 요청을 가장 먼저 처리하는 함수이다. http 헤더의 첫 줄을 읽어 요청 메서드와 요청 URI를 추출한다.
  다음으로 쿠키를 체크한다. 이는 따로 [cookieCheck 함수]를 만들어 관리한다.
  요청이 GET이라면 [handleGetRequest 함수]를 통해 요청을 관리한다. 아니라면 [sendRseponse 함수]를 통해 405(Method Not Allowed) 상태코드를 클라이언트에게 보낸다.
  자바의 finally 문법을 통해 통신용 소켓을 닫을 수 있게 처리했다.

GET 요청 핸들링 함수

```
private static void handleGetRequest(String uri, OutputStream out, String
cookieValue) throws IOException {
        // 각 경로마다 핸들링
        // 잘못된 URI인 경우 404 Not Found 응답
}
```

- 입력 : 각 경로(String), 통신용 소켓의 출력 스트림(OutputStream), 쿠키 값(String)
  루트 경로에선 `index.html` 파일을 [readFile 함수]를 통해 읽고 [sendResonse 함수]를 통해 응답한다.
  가구에 해당하는 경로들은 각 경로 마다 `"[가구] page requested"` 라는 로그를 남기고 `detail.html` 파일을 [readFile 함수]를 통해 읽고 응답한다. 이때 [jsonToHtml 함수]를 통해 `furniture.json` 파일의 내용으로 HTML을 수정하여 [sendResonse 함수]를 통해 응답한다.
  잘못된 경로인 경우 404(Not Found) 상태코드를 [sendResonse 함수]를 통해 응답한다.
  `[가구]` 는 `chair`, `table`, `closet` 을 의미한다.

응답 함수

```
private static void sendResponse(OutputStream out, String responseStatus,
byte[] responseBody, String cookieValue) throws IOException {
        // 쿠키 없으면 생성
        // 쿠키 있으면 log 남기기
        // body 있으면 첨부하고 응답 보내기
}
```

- 입력 : 통신용 소켓의 출력 스트림(OutputStream), 상태 코드를 담은 http 헤더(String), http 바디(byte[]), 쿠키 값(String)
  쿠키 값이 null이면 쿠키가 없는 것이므로 쿠키를 생성한다. `"New user requested page, cookie will be set."` 로그를 남긴다.
  쿠키 값은 본인의 학번인 `"2019008813"`으로 설정하고 쿠키가 1시간만 유지되도록 했다.
  `responseStatus`에 쿠키 관련 String을 붙여서 클라이언트가 쿠키를 받을 수 있도록 구현했다.
  쿠키 값이 있다면 `"Returning user, welcome [value]"` 로그를 남긴다.
  마지막으로 http 바디가 null이 아니면 이를 첨부해서 응답을 보낸다.

쿠키 체크 함수

```
private static String cookieCheck(BufferedReader in) throws IOException {
        // 클라이언트로부터의 HTTP 요청에서 쿠키 확인
        // 있으면 쿠키 값 리턴
        // 없으면 null 리턴
}
```

- 입력 : 통신용 소켓의 입력(BufferedReader)
- 출력 : 쿠키 값 또는 null(String)
  HTTP 요청 입력에 쿠키가 있는지 while문을 통해 한 줄씩 체크한다. 있다면 쿠키 값을 리턴한다.
  만약 while문을 빠져나와 쿠키가 없다고 판단하면 null을 리턴한다.

JSON 파싱 및 detail.html 수정 함수

```
private static String jsonToHtml(String htmlContent, int num) throws
IOException {
        // JSON 읽기
        // 가구에 맞게 HTML 수정
        // 이미지 파일을 바이트 배열로 읽기
        // 바이트 배열을 Base64 문자열로 인코딩하여 HTML에 첨부
        // 수정된 HTML 리턴
}
```

- 입력 : HTML(string), 번호(int)
- 출력 : 수정된 HTML(string)
  `funiture.json` 파일을 [readFile 함수]로 불러온다.
  `.com.google.gson 패키지`를 사용해 JSON을 파싱한다.
  입력받은 번호에 따라 `[가구]`에 관련된 정보를 JSON으로부터 불러온다. (0: chair 1: table 2: closet)
  입력받은 HTML에 불러온 정보를 이용해 `detail.html` 파일을 수정한다. 이때, 이미지는 path이므로 이미지를 byte 배열로 불러와 BASE64로 인코딩해서 HTML 데이터에 삽입한다.
  수정된 HTML을 리턴한다.

파일 읽기 함수

```
private static String readFile(String filePath) throws IOException {
        // 해당 경로의 파일을 읽어옴
        // 한 줄씩 읽어서 String으로 합침
        // 합친 String 리턴
}
```

- 입력 : 파일주소(string)
- 출력 : 파일(string)
  HTML, JSON 파일을 줄마다 읽기 위해 사용된다.
  파일의 주소를 받고 null인 줄이 나올 때까지 불러오고 병합한다.
  String으로 병합된 파일을 리턴한다.

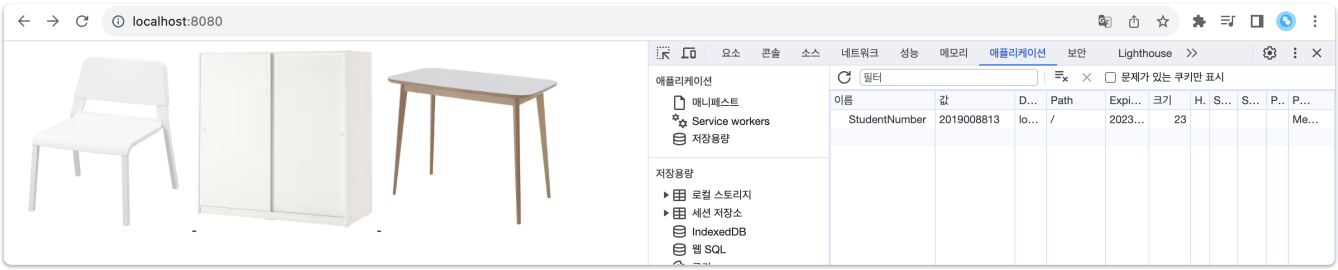
---

2. Result

- 저번 과제와 동일한 결과를 나타내게 된다.

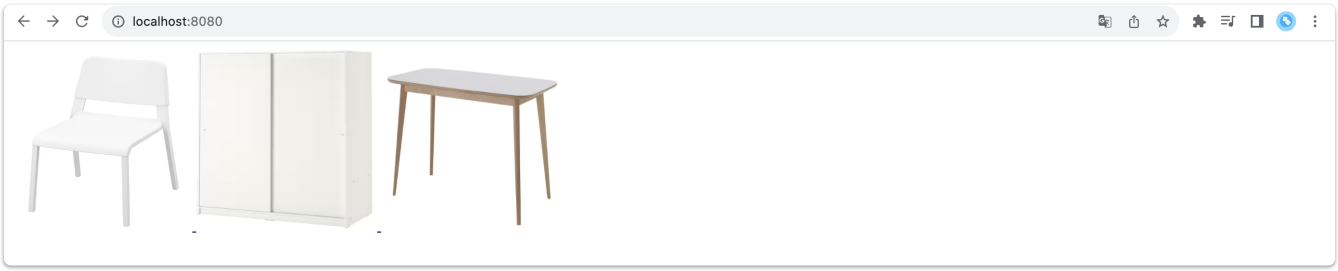
쿠키 체크

쿠키가 없는 상태에서 서버에 접속하면 3번째 줄과 같이 새로운 유저로 인식하고 쿠키를 생성한다.



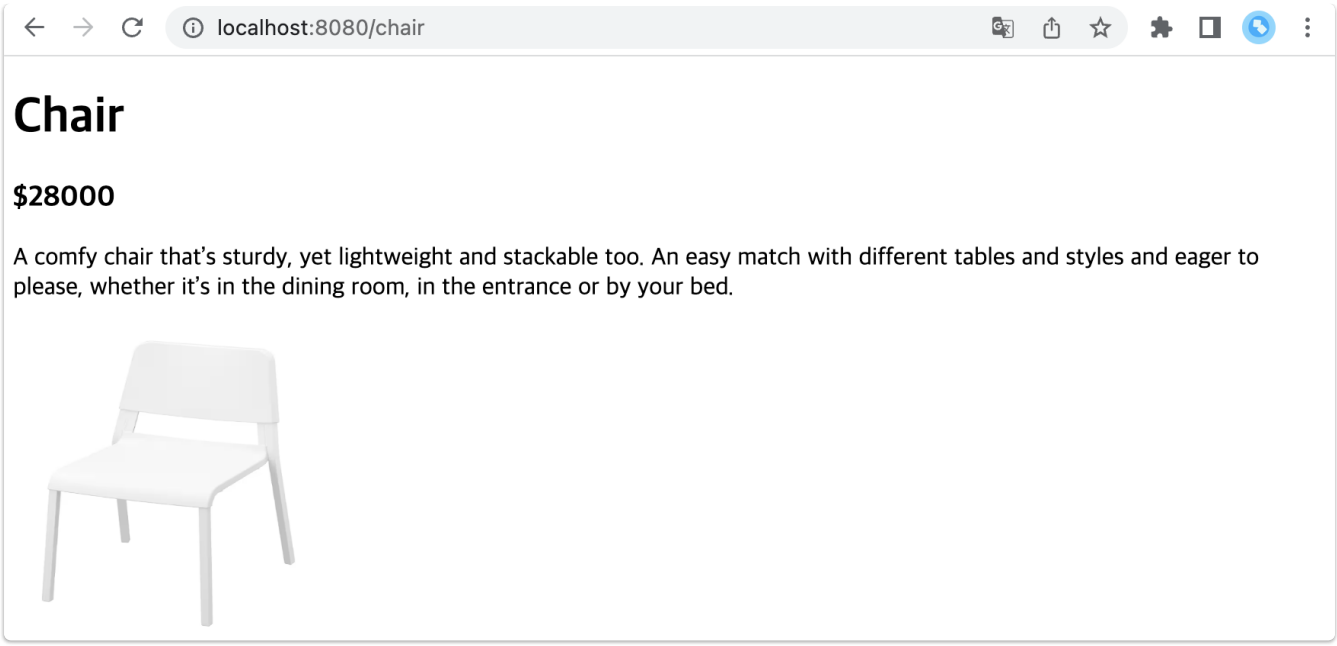
새로고침하여 다시 서버에 요청을 하면 요청 http 헤더에 쿠키가 존재하므로 기존의 유저로 인식한다.

루트 페이지



`index.html` 파일이 잘 전달된 모습이다.

세부사항 페이지 (chair)



`furniture.json` 파일을 파싱해 동적으로 데이터를 얻고 `detail.html` 파일이 제대로 수정됨을 확인할 수 있다.
다른 가구들 또한 동일하게 작동하므로 사진은 생략했다.

---

3. Trouble Shooting

index.html

주어진 index.html 파일에서 사진과 href 경로가 맞지 않음을 확인하였다. 따라서 각 사진에 맞게 index.html 파일의 href 경로를 수정해 주어서 해결했다.