Assignment1 design document

수업코드 12175

2018079116 정진성

Contents

Summary

Assignment1은 가구 카탈로그를 제공하는 작은 웹 서버를 만드는 것입니다. HTTP 프로토콜을 사용하여 클라이언트와 서버가 상호작용하고 쿠키를 사용하여 Statefull을 구현합니다.

Specification

- Java 17 version
- com.sun.net package
- java.net.HttpCookie package

Design and Implementation

HTTP 프로토콜을 사용하는 서버를 만들기 위해 com.sun.net.httpserver 패키지의 HttpServer 클래스를 사용하였습니다.

```
// HttpServer 클래스 이용해서 8080 포트에 인스턴스 생성
HttpServer httpServer = HttpServer.create(new InetSocketAddress(8080), 0);
System.out.println("Listening on port: " +
httpServer.getAddress().getPort());
```

HttpServer.create 메소드를 사용하여 InetSocketAddress클래스의 인스턴스를 생성해 인자로 넣어주어 8080포트에 서버를 생성하였습니다. getPort() 메소드를 사용하여 현재 어느 포트에 서버가 생성되었는지 로그로 알 수 있게 만들었습니다.

```
// 요청 처리할 RequestHandler 설정
httpServer.createContext("/", new RequestHandler());
```

createContext 메소드를 사용하여 "/"의 경로로 요청 시 RequestHandler()를 생성해 처리하도록 하였습니다. 즉 모든 경로의 요청은 RequestHandler 클래스의 인스턴스가 처리하도록 만들었습니다.

```
// 요청 처리할 스레드풀 설정 (null = 기본)
httpServer.setExecutor(null);
httpServer.start();
```

HTTP 서버가 클라이언트의 요청의 처리하는 데 사용할 스레드 풀을 setExecutor에 null을 인자로 주어 설정하였습니다. Null을 사용하면 기본 스레드 풀을 사용하여 동적으로 여러 요청을 처리할 수 있게 해줍니다. httpServer.start()로 서버를 시작하여 http:/localhost:8080 에서 클라이언트 요청을 수신하였습니다.

```java
public class RequestHandler implements HttpHandler {
    @Override
    public void handle(HttpExchange httpExchange) throws IOException {
        …
    }
}
```

클라이언트의 요청을 처리하는 클래스인 RequestHandler 클래스는 HttpHandler 인터페이스의 handle 메소드를 구현하여 만들었습니다. Handler를 구현하여 사용자가 정의한 방식으로 요청을 처리할 수 있도록 하였습니다.

Handler는 크게 4가지로 구현하였습니다.

```java
        // index page
        if (path.equals("/")) {
            path += "index.html";
            System.out.println("Index page requested");

            // 쿠키 처리
            manageCookie(httpExchange);

            // 전송 path, type 지정
            detailedPath = basePath + "html/" +path;
            type = "text/html";
        }
```

Index 페이지로 들어가는 경로에서는 html파일의 경로로 path를 수정해주고 쿠키 관련 처리를 위해 만든 manageCookie 메소드를 실행합니다. 또한 응답 헤더의 타입을 설정해주기 위해 type을 text/html로 지정합니다.

```
        // detail page
        else if (path.equals("/table") || path.equals("/chair") ||
path.equals("/closet")) {
            System.out.println(path.substring(1) + " page requested");

            // 쿠키 처리
            manageCookie(httpExchange);

            // 전송 path, type 지정
            detailedPath = basePath + "html/detail.html";
            type = "text/html";
        }
```

detail page는 가구 이미지를 클릭하는 요청을 처리하도록 하였습니다. Index 페이지와 마찬가지로 path, type을 지정해주고 쿠키를 처리합니다.

```
        // JSON request
        else if (path.equals("/furniture.json")) {
            System.out.println("furniture.json requested");

            // 전송 path, type 지정
            detailedPath = basePath + "data" + path;
            type = "application/json";
        }
```

Detail 페이지를 가구 이름에 맞게 동적으로 구현하기 위해 json을 요청하는 처리도 구현하였습니다. Furniture.json 파일의 경로를 지정해주고 type을 application/json으로 저장하였습니다. 쿠키는 페이지 요청일 때만 처리하기 위해 넣지 않았습니다.

```
        // image request
        else if (path.equals("/table.png") || path.equals("/chair.png") ||
path.equals("/closet.png")) {
            System.out.println(path.substring(1) + " requested");

            // 전송 path, type 지정
            detailedPath = basePath + "data" + path;
            type = "image/png";
        }
```

Detail page에 사용될 가구의 이미지 요청을 처리하는 부분입니다. 마찬가지로 경로를 설정하고 type을 image/png로 지정합니다. 쿠키는 페이지 요청일 때만 처리하기 위해 넣지 않았습니다.

```
        // 응답 전송
        sendResponse(detailedPath,httpExchange, type);
```

처리를 마치면 sendResponse 메소드를 통해 응답을 전송합니다.

```
        // 클라이언트에게 응답 보내는 메소드
    public static void sendResponse(String path, HttpExchange httpExchange,
String type) throws IOException {
        // 요청한 path 의 파일을 읽어 byte 배열로 저장
        File file = new File(path);

        // 파일이 존재하지 않을 경우 404
        if (!file.exists()) {

            System.out.println("unexpected resource requested");
            String response = "404 Not Found";
            byte[] responseBytes = response.getBytes("UTF-8");

            // 응답해더에 type 지정
            Headers headers = httpExchange.getResponseHeaders();
            headers.set("Content-Type", "text/plain");

            // 상태 코드 404, 파일 길이 헤더에 설정해서 send
            httpExchange.sendResponseHeaders(404, responseBytes.length);

            // 클라이언트에게 전송
            OutputStream outputStream = httpExchange.getResponseBody();
            outputStream.write(responseBytes);
            outputStream.close();
        }
    }
```

sendResponse는 파일 경로와 HttpExchange 클래스, type을 인자로 받아 로직을 처리합니다. File이 존재하지 않을 경우에는 404 Not Found가 페이지에 출력되도록 구현하였습니다.

```java
// 클라이언트에게 응답 보내는 메소드
    public static void sendResponse(String path, HttpExchange httpExchange,
String type) throws IOException {
        // 요청한 path 의 파일을 읽어 byte 배열로 저장
        File file = new File(path);

        // 파일이 존재하지 않을 경우 404
        ...
        else
        {
            FileInputStream fileInputStream = new FileInputStream(file);
            byte[] fileBytes = new byte[(int) file.length()];
            fileInputStream.read(fileBytes);
            fileInputStream.close();

            // 응답헤더에 type 지정
            Headers headers = httpExchange.getResponseHeaders();
            headers.set("Content-Type", type);

            // 상태 코드 200, 파일 길이 헤더에 설정해서 send
            httpExchange.sendResponseHeaders(200, fileBytes.length);

            // 클라이언트에게 전송
            OutputStream outputStream = httpExchange.getResponseBody();
            outputStream.write(fileBytes);
            outputStream.close();
        }
    }
```

파일이 존재할 경우에는 File을 바이트 배열로 저장하고 응답 헤더에 인자로 받은 타입을 Content-Type에 지정합니다. 상태코드로 200, file의 바이트 수를 헤더에 담고, 바디에는 파일의 내용을 outputStream으로 담아 전달합니다.

```java
import com.sun.net.httpserver.Headers;
import com.sun.net.httpserver.HttpExchange;

import java.net.HttpCookie;

public class CookieHandler {
    // 쿠키 존재 확인
    public static boolean isExist(HttpExchange httpExchange, String key) {
        String cookie = httpExchange.getRequestHeaders().getFirst(key);
        return !(cookie == null || !cookie.contains("StudentNumber"));
    }

    // 쿠키 반환
    public static String getCookie(HttpExchange httpExchange, String key) {
        return httpExchange.getRequestHeaders().getFirst(key);
    }

    // 쿠키 생성
    public static void setCookie(HttpExchange httpExchange, String key,
String value) {
        // 쿠키 name, value 설정
        HttpCookie httpCookie = new HttpCookie(key,value);

        // 쿠키 경로 설정("/"하면 모든 경로에서 쿠키접근 가능)
        httpCookie.setPath("/");

        // 응답헤더 가져와서 쿠키 추가
        Headers responseHeader = httpExchange.getResponseHeaders();
        responseHeader.add("Set-Cookie", httpCookie.toString());
    }
}
```

쿠키 처리는 CookieHandler 클래스를 구현하여 만들었습니다. httpExchange.getRequestHeader를 이용해 헤더를 받아 key를 키로 가지는 인스턴스가 있는지 확인하는 메소드를 구현하였고, 같은 방식으로 쿠키를 반환하는 메소드, 또한 HttpCookie 패키지를 사용해 쿠키를 생성하는 메소드를 구현하였습니다. 쿠키의 경로를 setPath("/)로 설정하여 모든 경로에서 쿠키가 접근 가능하도록 하였고 응답을 전송할 때 헤더에 쿠키를 담아 전송해야 하기에 add를 사용해 Set-Cookie에 쿠키 값을 넣는 방식으로 구현하였습니다.

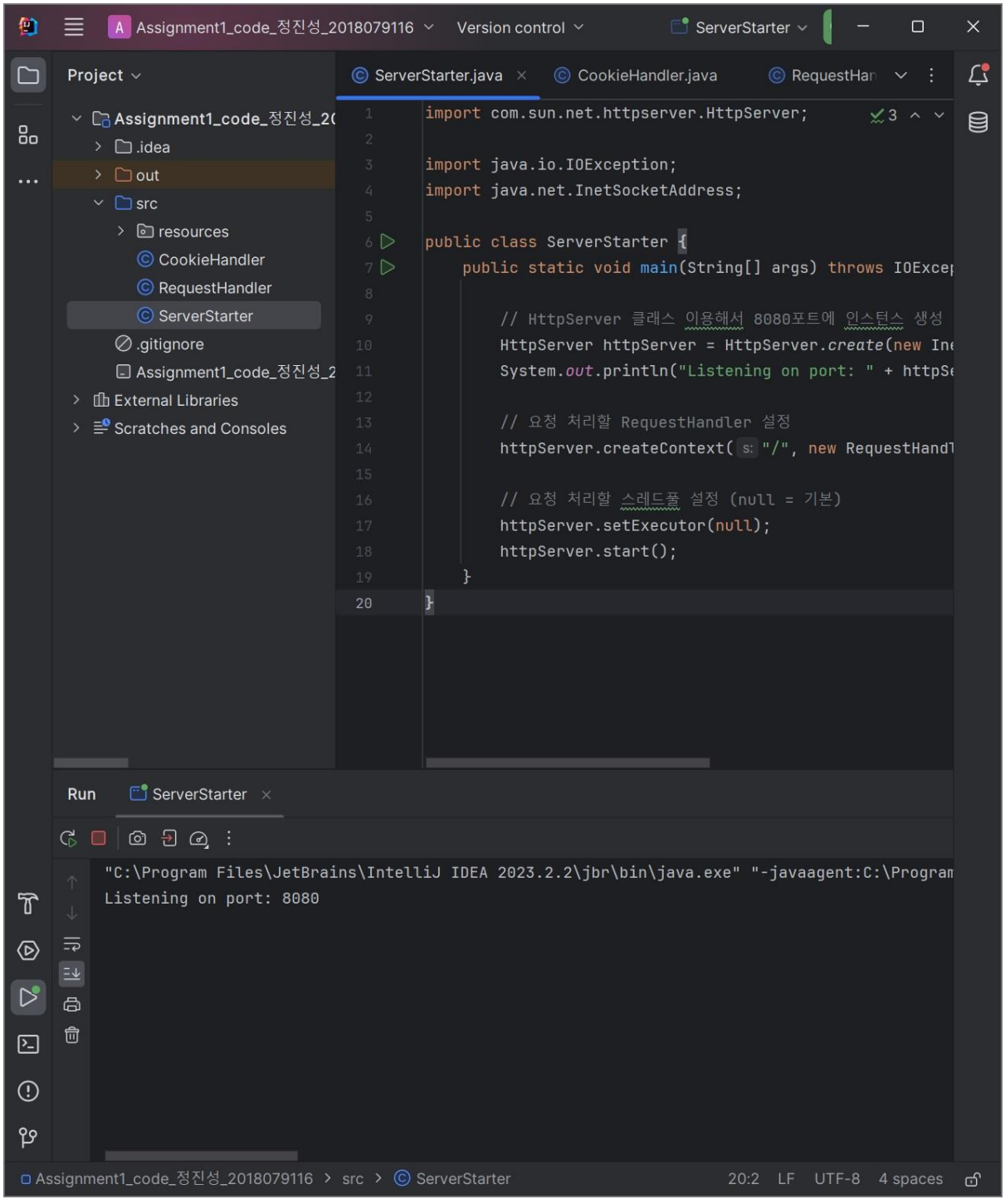
```
// 쿠키 관련 처리 메소드
    public static void manageCookie(HttpExchange httpExchange) {
        // 쿠키 이미 존재
        if (CookieHandler.isExist(httpExchange, "Cookie")) {
            // 쿠키 값 가져오기
            String cookie = CookieHandler.getCookie(httpExchange, "Cookie");
            System.out.println("Returning user, welcome " +
cookie.substring(15,cookie.length()-1));
        }
        // 쿠기 없음
        else {
            System.out.println("New user requested page, cookie will be
set.");
            // 쿠키 생성
            CookieHandler.setCookie(httpExchange, "StudentNumber",
"2018079116");
        }
    }
```

mamgeCookie 메소드는 RequestHandler 클래스에서 쿠키를 처리하도록 만든 static 메소드입니다. 쿠키가 현재 브라우저에 존재하는지 검사하고 그에 따라 쿠키가 존재한다면 쿠키의 value를 출력하고, StudentNumber의 쿠키가 없다면 쿠키를 생성합니다.

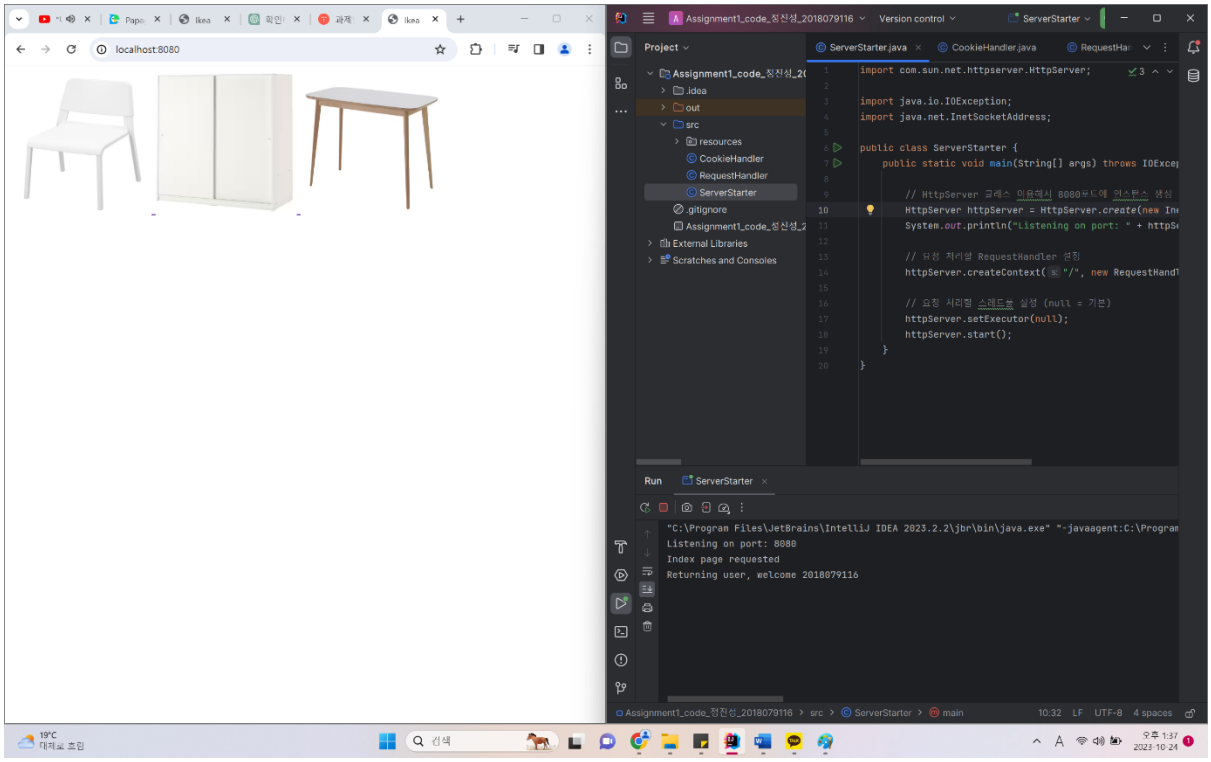
```
<script>
    var resource = window.location.href.split('/');
    resource = resource[resource.length - 1];

    fetch("./furniture.json")
        .then(function (response) {
            return response.json();
        })
        .then(function (data) {
            var json = data["Furniture"];
            for ( var i = 0; i < 3 ;i++) {
                if (json[i]["Name"].toLowerCase() === resource) {
                    var obj = json[i];
                    document.getElementById("title").innerHTML = obj["Name"];
                    document.getElementById("price").innerHTML =
obj["Price"];
                    document.getElementById("description").innerHTML =
obj["Description"];
                    document.getElementById("img").src =
obj["ImageLocation"];
                }
            }
        })
        .catch(function (error) {
            console.error("데이터를 로드하는 동안 오류가 발생했습니다.", error);
        });
</script>
```

detail.html에 javascript 코드를 추가하여 동적으로 html 페이지가 변경되도록 구현하였습니다. 여기서 furiniture.json을 요청한 후 현재 경로에서 어떤 상세 페이지인지를 알아내어 그에 맞는 title, price, description, image src를 변경하도록 구현하였습니다.

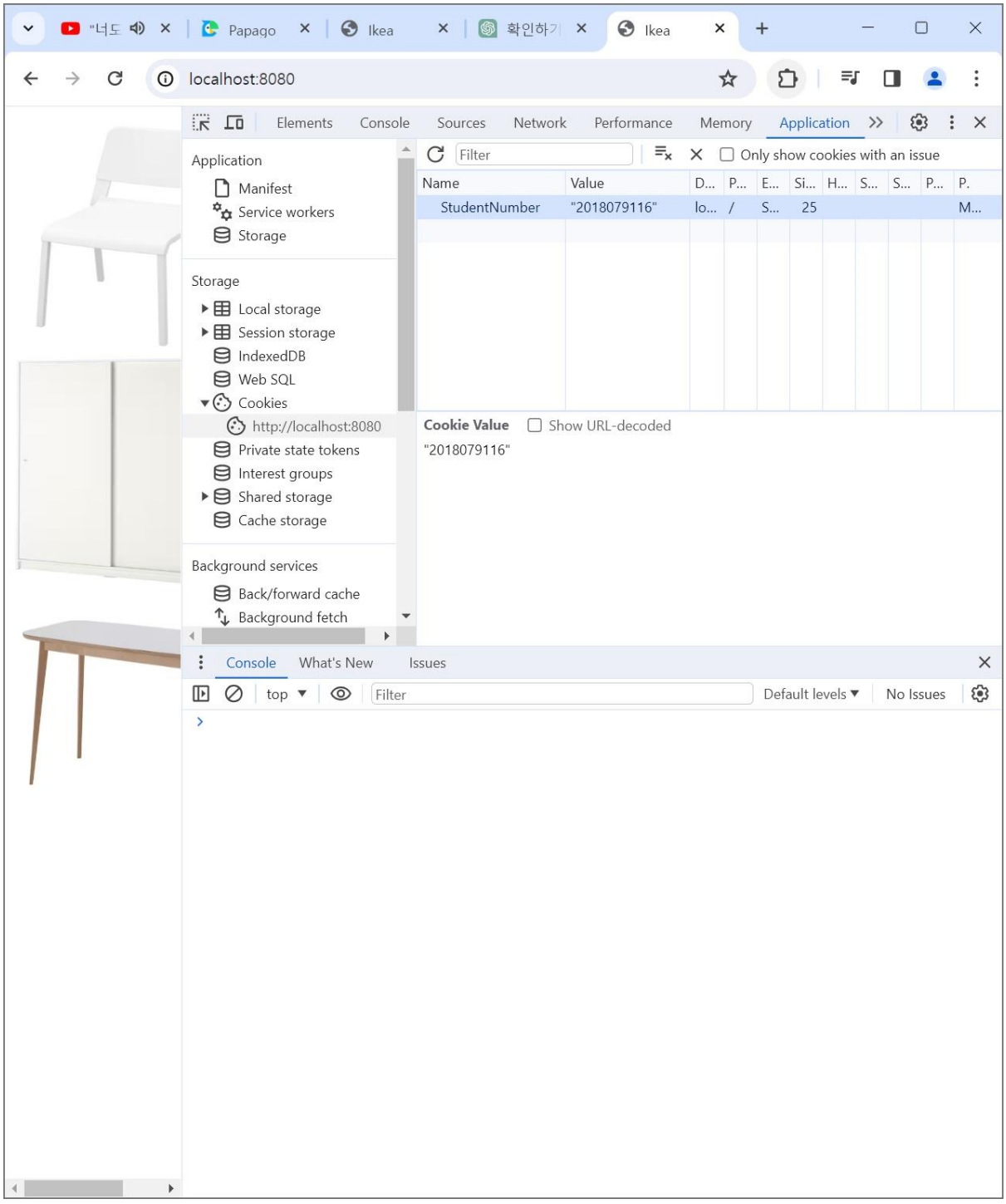
Testing



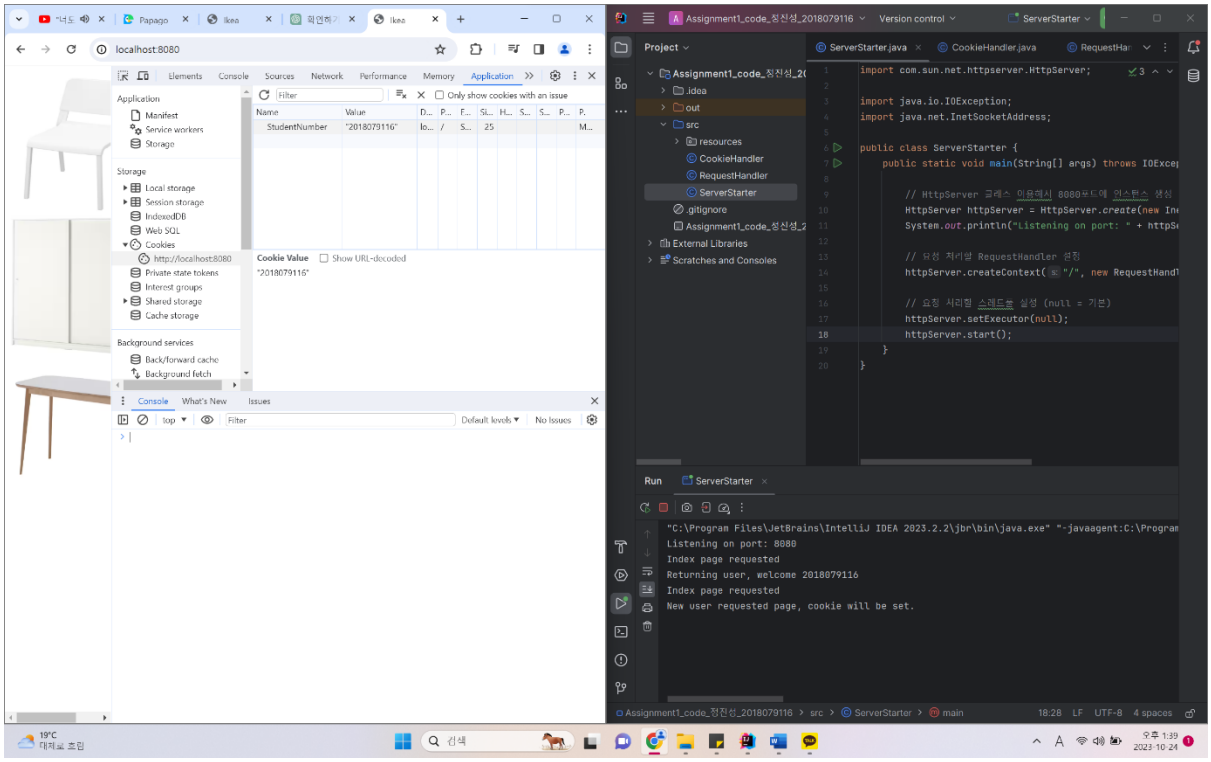
서버를 실행하기 위해 인텔리제이를 이용해 실행하였습니다. ServerStarter를 실행하면 8080 포트에 서버를 생성하고, 그에 따른 로그를 출력합니다.



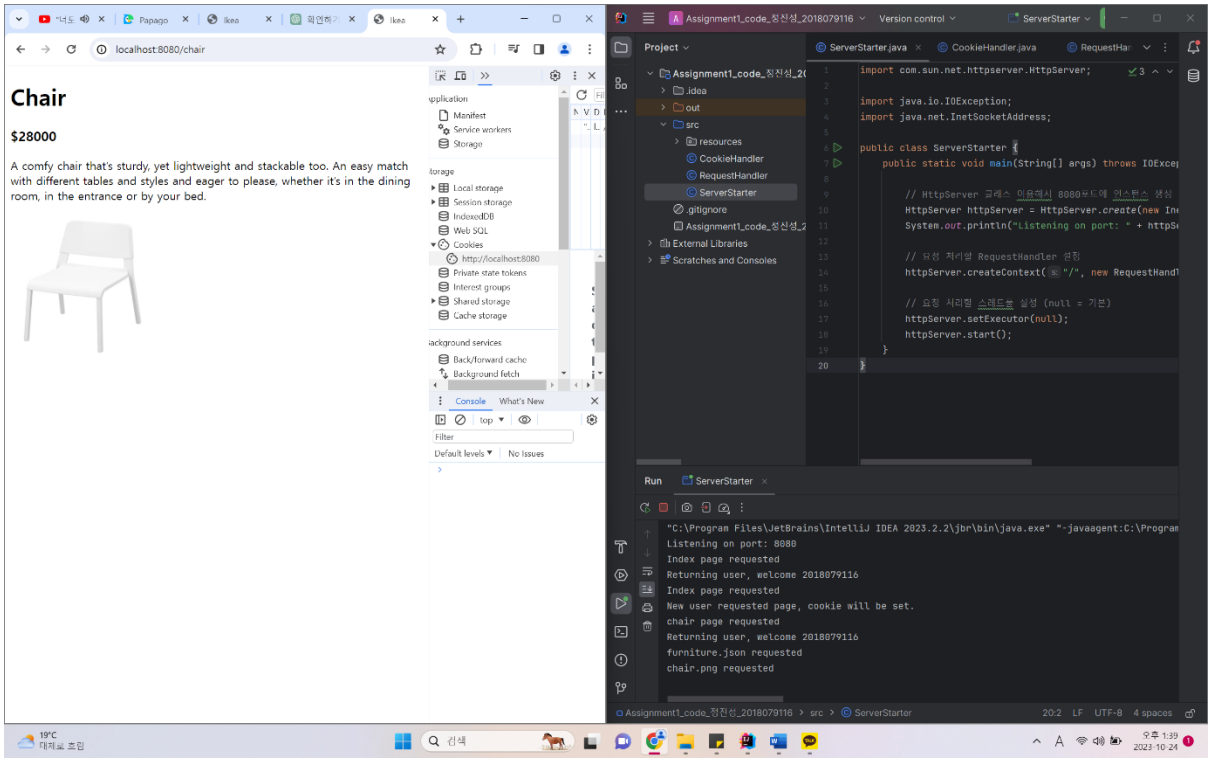
Localhost:8080으로 접속하면 index page를 요청했다는 로그를 출력합니다. 이후 쿠키를 확인하고 이미 studnetnumber쿠키가 지정되어 있어 Returnning user, welcome 로그를 출력합니다.

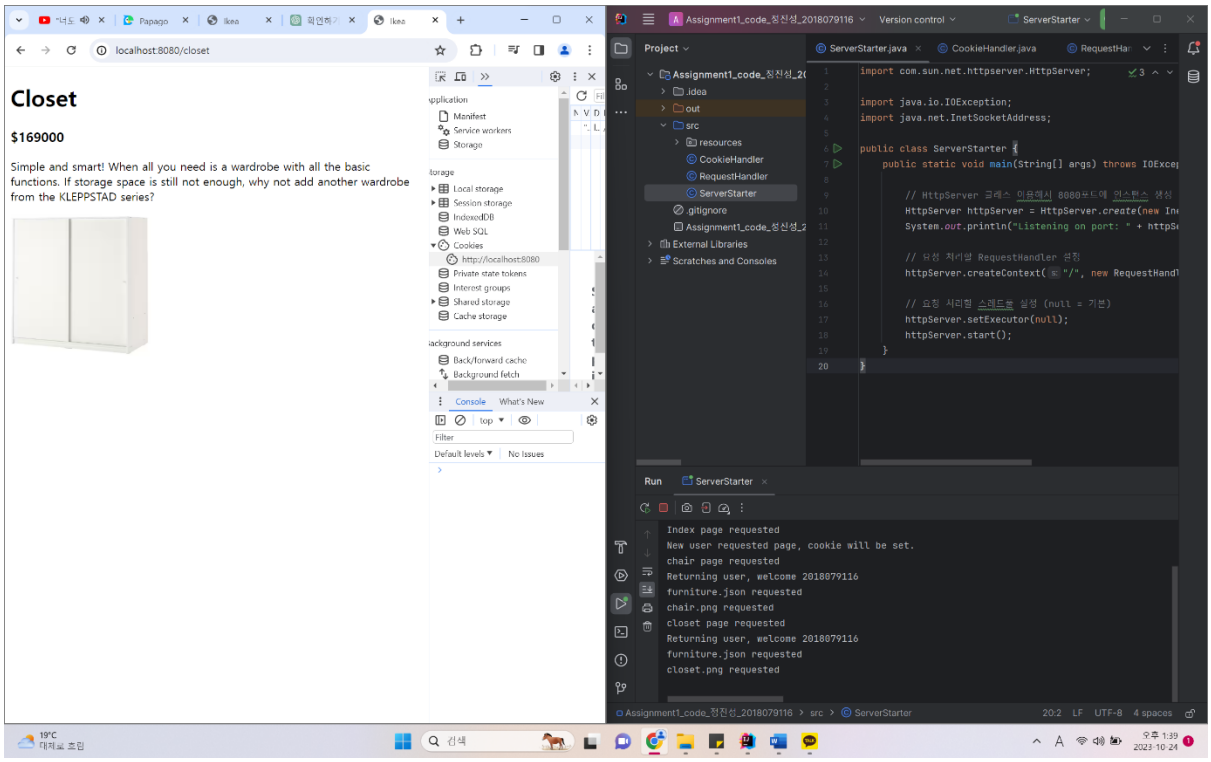


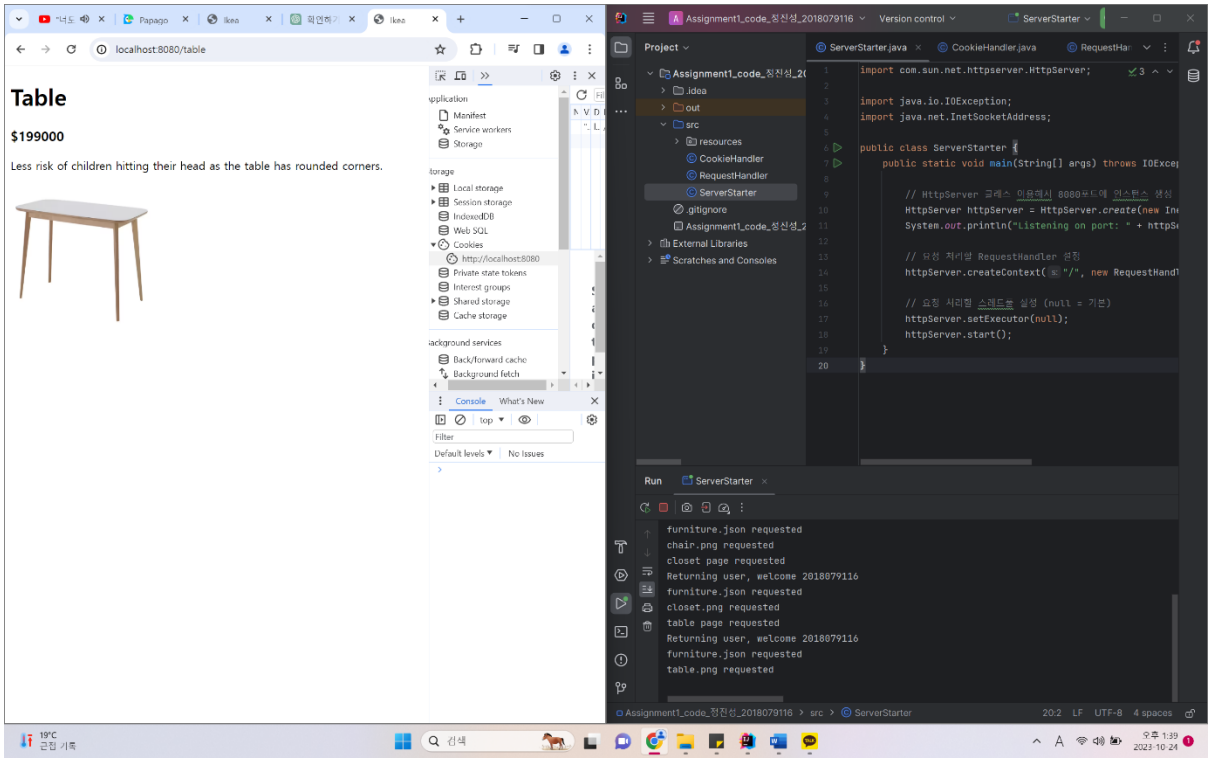
쿠키를 확인해보면 StudentNumber, 2018079116으로 저장되어있음을 확인할 수 있습니다.



쿠키 제거 후 다시 접속해 보면 New user 메시지가 출력되고 새롭게 쿠키가 생성되었음을 확인할 수 있습니다.


localhost:8080/chair
Chair
$28000
A comfy chair that's sturdy, yet lightweight and stackable too. An easy match with different tables and styles and eager to please, whether it's in the dining room, in the entrance or by your bed.
Assignment1_code_정진성_2018079116
Version control
ServerStarter
ServerStarter.java
CookieHandler.java
import com.sun.net.httpserver.HttpServer;
import java.io.IOException;
import java.net.InetSocketAddress;
public class ServerStarter {
HttpServer httpServer = HttpServer.create(new In
System.out.println("Listening on port: " + httpS
httpServer.createContext(s: "/", new RequestHandl
httpServer.setExecutor(null);
httpServer.start();
Run
ServerStarter
"C:\Program Files\JetBrains\IntelliJ IDEA 2023.2.2\jbr\bin\java.exe" "-javaagent:C:\Program
Listening on port: 8080
Index page requested
Returning user, welcome 2018079116
Index page requested
New user requested page, cookie will be set.
chair page requested
Returning user, welcome 2018079116
furniture.json requested
chair.png requested


localhost:8080/closet
Closet
$169000
Simple and smart! When all you need is a wardrobe with all the basic functions. If storage space is still not enough, why not add another wardrobe from the KLEPPSTAD series?
Assignment1_code_정진성_2018079116
Version control
ServerStarter
ServerStarter.java
CookieHandler.java
import com.sun.net.httpserver.HttpServer;
import java.io.IOException;
import java.net.InetSocketAddress;
public class ServerStarter {
HttpServer httpServer = HttpServer.create(new In
System.out.println("Listening on port: " + httpS
httpServer.createContext(s: "/", new RequestHandl
httpServer.setExecutor(null);
httpServer.start();
Run
ServerStarter
Index page requested
New user requested page, cookie will be set.
chair page requested
Returning user, welcome 2018079116
furniture.json requested
chair.png requested
closet page requested
Returning user, welcome 2018079116
furniture.json requested
closet.png requested



각각의 상세 페이지는 furniture.json, detail page, image를 요청하고 있으므로 그에 따른 메시지 로그들이 출력되었습니다.