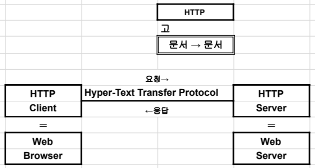

HTTP

고

문서 → 문서

요청→

Hyper-Text Transfer Protocol

←응답

HTTP Client = Web Browser

HTTP Server = Web Server

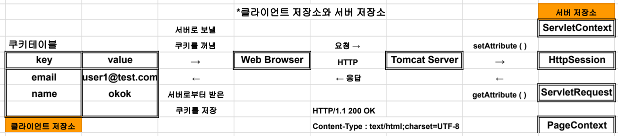
이 부분은 웹브라우저가 자동으로 처리

↓

서버로부터 받은 쿠키를 클라이언트에서 →
다시 서버로 보내는 방법

```
GET /app/auth/login HTTP/1.1
HOST : localhost:8888
cookie : email=user1@test.com; name=okok
```

*클라이언트 저장소와 서버 저장소


쿠키테이블

| key | value |
|---|---|
| email | user1@test.com |
| name | okok |

클라이언트 저장소

서버로 보낼 쿠키를 꺼냄 →

← 서버로부터 받은 쿠키를 저장

Web Browser

요청 →
HTTP
← 응답

Tomcat Server

setAttribute ( ) →

← getAttribute ( )

서버 저장소

| | | | |
|---|---|---|---|
| ServletContext | 사용자가 바뀌어도 공유라면 | *DAO / Service / Connection | |
| HttpSession | | *로그인 사용자 / 입력값 | 페이지가 바뀌더라도 유지한다면 사용 |
| ServletRequest | 하나의 요청을 처리할때, 여러개의 서블릿이 공유를 해서 작업을 할때 | *요청처리결과 /게시판 목록 | Include/forword하는 서블릿끼리 공유할 것이냐<br>*JSP 실행 X -> 서블릿이 실행됨 |
| PageContext | JSP 내에서 사용할 때 | *EL에서 사용할 객체 | |

```
HTTP/1.1 200 OK
Content-Type : text/html;charset=UTF-8
-
-
Set-Cookie: email; user1@test.com
Set-Cookie: name = okok
┘ 응답콘텐트
```

서버에서 클라이언트 저장소에
보관할 데이터를 보내는 방법
"cookie"라 부른다.

←★ 이 부분은 개발자의 코딩이 필요!

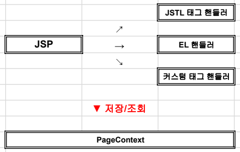
★ 쿠키는 클라이언트 쪽에 저장되기 때문에 쉽게 노출된다.

★ 보안데이터는 쿠키로 저장하면 안된다.

★ 쿠키데이터는 문자열이다!!!!!!★★

auth를 요청할때만 보냄

*JSP 와 PageContext


JSP

JSTL 태그 핸들러

EL 핸들러

커스텀 태그 핸들러

▼ 저장/조회

PageContext

JSP와 태그핸들러 사이에 객체를 공유할 때 사용하는 저장소

JSP는 화면에 뿌리는 일만 담당

JSP내에 특별한 태그를 넣는 것은 무의미 => 핸들러 사용

스프링부트는 JSP 를 쓰지 않도록 권유.

JSP는 굳이 디테일하게 배우지 않아도됨

EL문법 1-9번 예제 정도

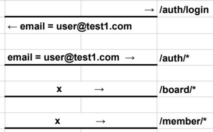
JSTL 예제에 있는거 써보는 정도

그 이상의 문법을 사용하면 퇴사하는게 좋음

*쿠키

Cookie 테이블

| name | value |
|---|---|
| email | user1@test.com |


→ /auth/login

← email = user@test1.com

email = user@test1.com → /auth/*

x → /board/*

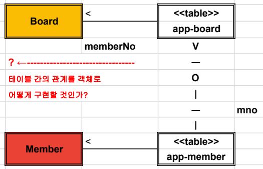
x → /member/*

← /auth/에 소속된 서블릿으로부터 받은 쿠키는
같은 /auth/에 소속된 서블릿을 요청할때만 서버로 보낸다.

경로에 소속된 서버로 부터 받은 쿠키는
그 경로의 서블릿으로 보내고 받는게 기본

로그인 사용자의 데이터 정보는 서버의 세션에서 꺼내서 써야함

*066. 로그인 정보 활용하기



? ←---------------------------------

테이블 간의 관계를 객체로
어떻게 구현할 것인가?

*게시글에 작성자 이름을 포함하기

방법1)

| Board |
|---|
| no |
| title |
| content |
| memberNo |
| password |
| viewCount |
| createdDate |
| memberName |

← 작성자 이름을 저장할 필드 추가

← 작성자 이름을 객체에 담아서 추가

도메인 객체를 포함하는 방식을 더 많이 씀.

방법2)

| Board |
|---|
| no |

| Member |
|---|
| no |

| | | |
|---|---|---|
| **title** | member | **name** |
| **content** | ◇→ | **email** |
| **memberNo** | | **password** |
| **password** | | **createdDate** |
| **viewCount** | | |
| **createdDate** | | |

객체가 다른 객체를 포함하는 식으로 표현

```
@WebFilter("*")
public class LoginCheckFilter implements Filter {

@Override
public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
throws IOException, ServletException {

HttpServletRequest httpRequest = (HttpServletRequest) request;
HttpServletResponse httpResponse = (HttpServletResponse) response;
String servletPath = httpRequest.getServletPath();

if (servletPath.endsWith("add") ||
servletPath.endsWith("update") ||
servletPath.endsWith("delete")) {

Member loginMember = (Member)httpRequest.getSession().getAttribute("loginMember");
if(loginMember == null) {
httpResponse.sendRedirect(httpRequest.getContextPath() + "/auth/form.jsp");
return;
}
}
```

```
//다음 필터를 실행한다.
//다음으로 실행할 필터가 없다면 원래 목적지인 서블릿이 실행될 것이다.
chain.doFilter(request, response);
```