



프로그래밍/Web

# Spring과 Cookie & Session

국윤창 | 2018. 2. 5. 04:19

저번에 HTTP 프로토콜에 관해 간단히 알아봤는데, HTTP는 Connectionless, Stateless한 프로토콜이다.

- Connectionless: Client가 Request를 보내면 Server가 처리해서 Response를 한 뒤 접속을 끊는다.
- Stateless: Server가 Client에 관한 정보를 유지하지 않는다.

위의 두가지 특징때문에 다른 Request 간의 정보를 공유할 수 없다. 로그인 기능을 만드려고 하는데, 어느 Client가 로그인을 한 상태인지를 판단하려면 Cookie와 Session을 알아야한다.

## Cookie

Client에 데이터를 저장하는 수단이다. Client에 저장하므로 Server는 그 데이터를 받아야지만 사용할 수 있는데, 그 과정은 아래 1과같이 이루어진다.



[그림 1] Cookie 생성 및 Request

1. Client가 HTTP Request를 Server에 보낸다.

2. Server는 setcookie() 함수를 이용해 Cookie를 설정하고, HTTP 헤더와 함께 Cookie를 Client에게 전송한다.

3. Client는 HTTP Request를 할 때마다 Cookie를 설정한 뒤 Server에 HTTP 헤더와 함께 보낸다.



국윤창

노력중인 게으름뱅이



CATEGORY



Cookie의 크기는 최대 4kb로 상당히 작다. 데이터 외에 Cookie의 속성(Attribute)로는 아래와 같은 것들이 있다.

- Name: Cookie의 이름
- Value: Cookie의 값
- Secure: secure HTTPS 연결에서만 Cookie가 전송될 것인지의 여부. TRUE면 secure connection이 존재할 때만 Cookie가 보내진다. Default는 FALSE다.
- Domain: Cookie의 domain name이다. 설정한 domain(예를 들면, demo.com)의 모든 subdomain에서 이 Cookie가 이용가능하다. 다만 www.demo.com처럼 설정하면 안된다. 왜냐하면 이렇게 설정하면 www.subdomain에서만 접근 가능하기 때문이다.
- Path: 어떤 경로에서 Cookie를 사용가능하게 할지 지정하는 것. 예를 들어 "/"로 설정하면 domain 내의 모든 경로에서 Cookie를 사용할 수 있다. "/test/"로 설정하면 /test/ 와 그 아래 모든 sub directory에서 Cookie를 사용할 수 있다. Default는 Cookie가 만들어지는 경로이다.
- HTTPOnly: TRUE로 설정되면 HTTP 프로토콜을 통해서만 Cookie를 이용할 수 있다.(스크립팅 언어로는 접근이 안된다.) 이 설정은 XSS 공격으로부터 identity가 도둑맞지 않도록 도와준다. Default는 FALSE다.
- Expires: Cookie가 언제 파괴될 지 명시한다. 초 단위이며, 60\*60\*24\*30은 30일 뒤에 파괴된다는 것을 나타낸다. 이 값이 생략되거나 0으로 설정되면 Session이 끝날 때 파괴된다.(브라우저를 끌 때)

Server가 Cookie를 생성할 때 setcookie 함수를 사용하는데, 이 때 Name, Value, Expires는 꼭 명시해야 한다.

Cookie는 아래와 같은 다양한 목적을 위해 사용된다.

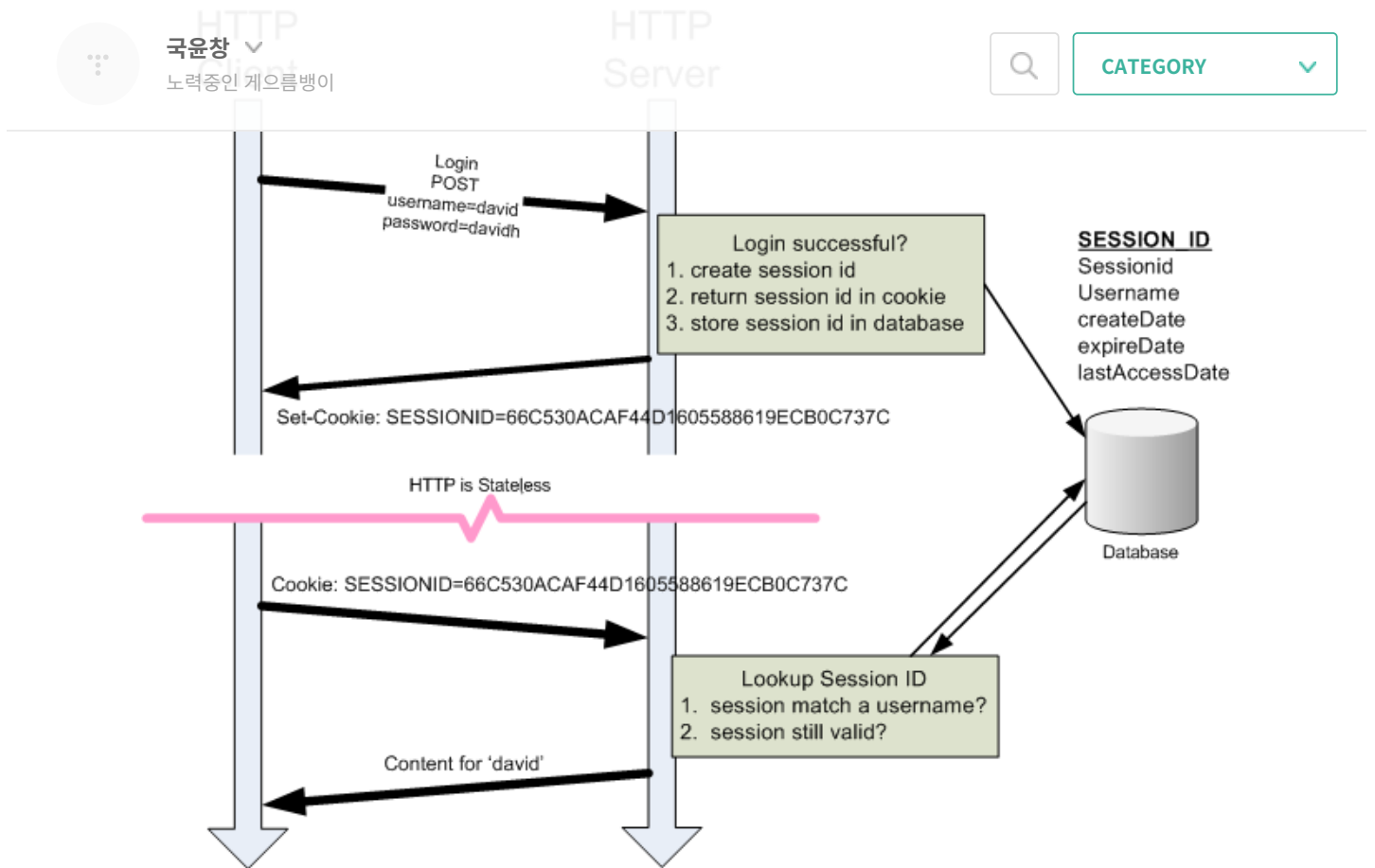
- 사용자의 신원 확인
- Application의 state를 추적하는 용도. (HTTP는 stateless이므로)
- Website에서 사용자가 얼마나 많은 시간을 있었는지 확인

Cookie의 종류에는 Session Cookie, Persistent Cookie, Third Party Cookie, Secure Cookie, HTTP Only Cookie, Zombies Cookie가 있는데 아래 참조 링크의 Cookie, Session이 자세히 나와있는 링크를 참고하자.

**Cookie는 사용자 컴퓨터에 파일로 남겨져서 보안상에 문제가 생길 수 있다.** 보안에 민감한 부분은 반드시 Server에 저장해야 한다. 그럼 Session에 대해서 알아보자.

## Session

Server에 데이터를 저장하는 수단이다. Session은 Cookie처럼 사용자를 식별하고 정보를 저장하는데 사용되지만, Server에 저장되므로 Cookie와 달리 정보가 노출되지 않는다. Session은 아래와같이 생성되고 이용된다. (Login을 예제로 가져옴)



[그림 2] Session 생성 및 사용

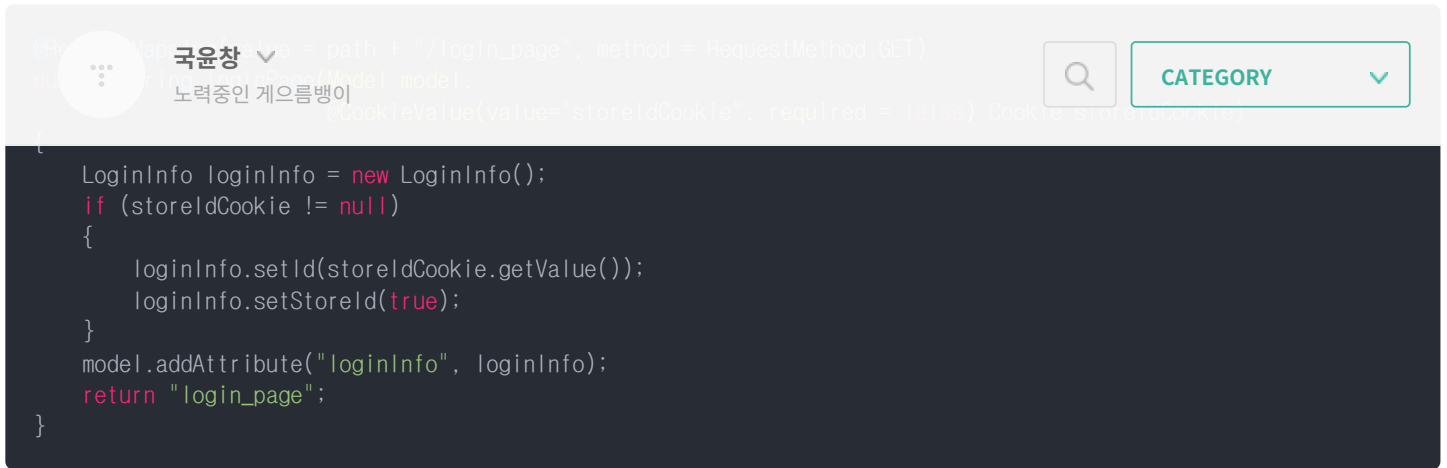
위 그림을 설명하기 전에 Session ID를 알아야 한다. **Session ID는 어떤 사용자의 Session인지를 식별하는 고유한 ID이다.** Request를 받았을 때 Session을 생성할 필요가 있으면(HTTP Request Header에 Session ID가 없으면), Session과 Session ID를 생성하고 `setcookie()` 함수로 Session ID를 담아서 Client에게 보낸다. 이 Session과 Session ID는 Server의 저장소(Database, Memory, File 등..)에 저장되며, Client는 그 후 Request를 보낼 때마다 Session ID를 Cookie에 담아 보내게 된다. Server는 Request를 받을 때마다 Session을 Session 저장소에서 Session ID로 찾는다. 아래 3단계와 같다.

1. Client가 Request를 보낸다.
2. Session ID가 Server 저장소에 존재하지 않으면 Session과 Session ID를 생성 및 저장하고 `setcookie()` 함수를 이용해 Cookie에 Session ID를 담아 Response로 돌려준다.
3. Client가 Request를 할 때 Session ID를 Cookie에 담아서 보내고, Server는 저장소에서 Session ID로 Session을 찾아서 필요한 정보를 Response에 담아 보낸다.

Session ID는 사용자를 구분할 수단임을 언급했는데, 이것을 해커가 가로채서 정보를 빼올수도 있다. 따라서 Session ID만으로는 보안이 다소 취약하다. 따라서 Session ID의 생성, 폐기 시간을 너무 길지 않게 정하고, Session ID값을 일정 시간마다 재생성, 일정 회수 이상의 인증 실패 시 서비스를 통제하는 등 다양한 보안처리를 해야한다. 자세한 내용은 아래 참조링크를 참고.

## Spring에서 Cookie 사용

Spring에서 Cookie를 사용하는 방법을 알아보자. 간단히 로그인 화면의 ID저장 기능을 추가 해봤다.



로그인 페이지로 이동하는 RequestMapping method이다. Method 내의 매개변수를 보면 매개변수 선언 앞에 `@CookieValue`라는 annotation이 붙어있는 것을 알 수 있다. **@CookieValue는 어떤 매개변수가 HTTP Cookie로 매핑될 것인지 명시하는 annotation이다.** 괄호 안에 속성들을 지정할 수 있는데, value는 Cookie의 이름을 나타내고, required는 이 Cookie가 반드시 존재해야 하는지 여부를 나타낸다.

위의 코드에서 Cookie가 null이 아니면 LoginInfo의 멤버변수를 Cookie에 있는 내용으로 바꾸는 것을 볼 수 있다. 아래 코드처럼 view로 넘겨서 사용할 수 있다.

```
<h1>Login</h1>
<p>Please enter ID & password</p>
<!--/*@thymesVar id="errStr" type="java.lang.String"*/-->
<div th:if="${errStr != null}" th:text="${errStr}"></div>
<!--/*@thymesVar id="loginInfo" type="com.board.dtos.LoginInfo"*/-->
<form action="#" th:action="@{/user/login}" th:object="${loginInfo}" method="post">
    ID: <br/>
    <input type="text" th:field="*{id}" th:value="*{id}"/><br/>
    Password: <br/>
    <input type="password" th:field="*{password}"/><br/>
    <input type="checkbox" th:field="*{storeId}" th:checked="*{storeId}"/>ID 저장
    <br/><br/>
    <input type="submit" value="Login"/>
</form>
```

위 코드에서 6번째줄부터 살펴보면 LoginInfo 객체를 Controller로부터 넘겨받아서, ID 입력 textbox와 ID 저장 checkbox에 멤버변수를 대입하는 것을 볼 수 있다. 만약 Cookie가 존재해서 Controller가 LoginInfo 객체에 값을 넣어서 view로 전달했다면, 초기값이 설정된 상태로 사용자에게 페이지가 보여질 것이다.

위 코드는 Cookie의 사용방법에 대한 코드이며 Cookie를 생성하는 코드는 아래에 나와있다.

```
@RequestMapping(value = path + "/login", method = RequestMethod.POST)
public String login(Model model, final HttpSession session,
                    HttpServletResponse response,
                    @ModelAttribute LoginInfo loginInfo)
{
    System.out.println(loginInfo.getId() + ", " +
                        loginInfo.getPassword() + ", " +
                        loginInfo.isStoreId());

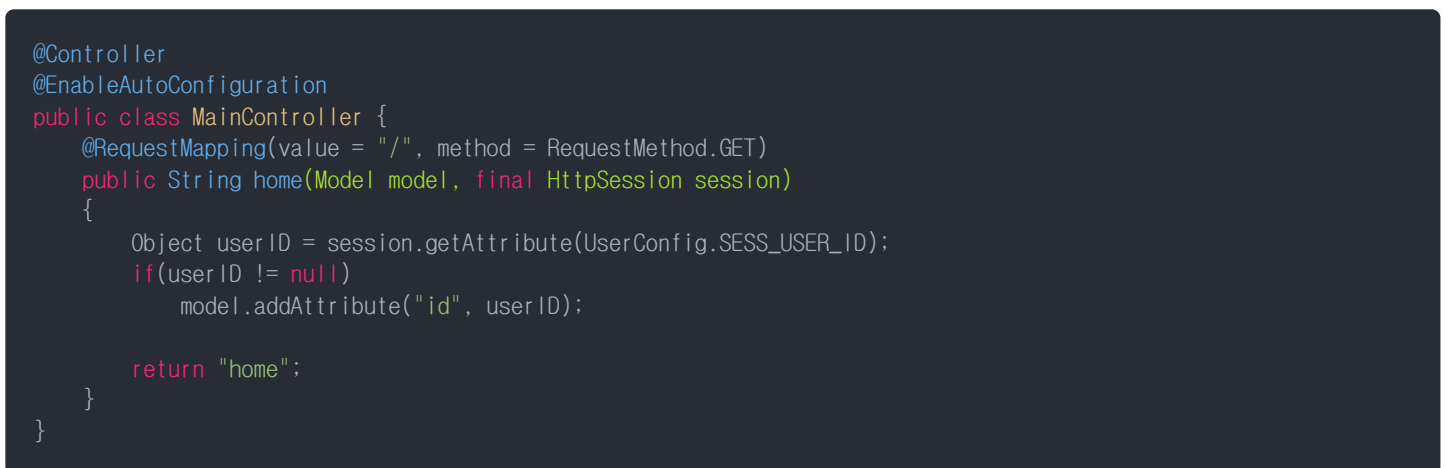
    // 로그인 정보 확인
    UserDTO userDTO = userService.loginVerification(loginInfo.getId(), loginInfo.getPassword());
    if(userDTO == null) {
        model.addAttribute("errStr", "로그인 실패: 아이디 및 패스워드를 확인 해주세요.");
        return "redirect:/login_page";
    }
}
```



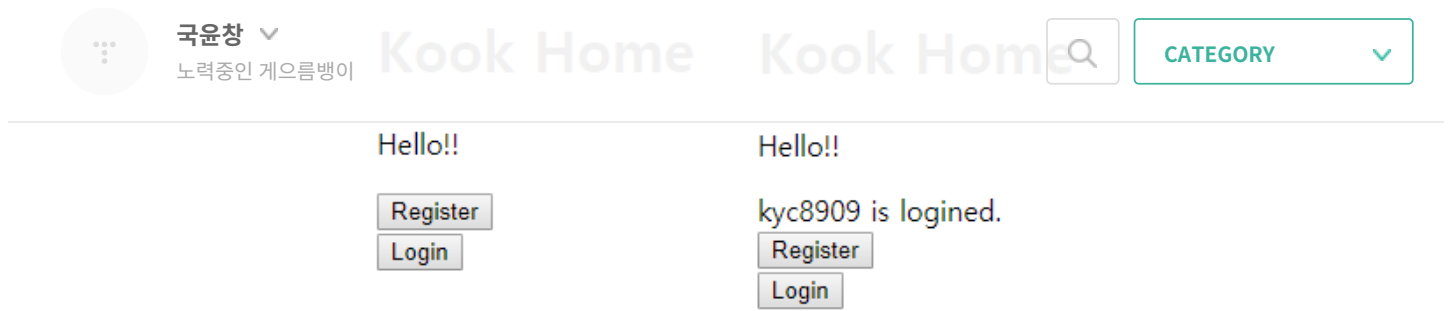
위의 login 메소드는 로그인을 시도할 때 접속되는 url에 매핑된 메소드이다. 쿠키 생성 및 설정 부분을 보면 view로부터 넘겨 받은 loginInfo 객체의 isStoreId() 함수가 true이면 Cookie를 생성하고 HttpServletResponse 객체에 담아서 클라이언트로 전송한다. 이 때 Cookie에 사용자의 ID를 담고, 어떤 url에서 Cookie를 사용할지와 Cookie의 유효기간을 설정한다. HttpServletResponse 객체는 Dispatcher Servlet으로부터 받은 것이다. Dispatcher Servlet은 다음에 Spring MVC패턴을 자세히 알아볼 때 살펴보자.

## Spring에서 Session 사용

사용자가 Login 후 Login이 됐는지 정보를 유지하기 위해 Session을 사용해보자. 위 코드에서 login 메소드의 매개변수 중에 HttpSession 객체가 있다. 이것을 가지고 Session attribute와 값을 설정하고, Cookie처럼 만료시간을 설정할 수 있다. **만료 시간이 지날 때까지 Request가 없으면 Session을 파기한다.** Session 생성이 완료되면, 아래 코드처럼 Session에 있는 값을 사용할 수 있다.



Session 객체에서 attribute 값을 얻어서 null이 아니면 view로 넘기는 함수이다. 로그인을 하면 아래 그림 3처럼 표시할 수 있다.




[그림 3] 로그인 전/후

다음번엔 게시판 만들기과 Dispatcher Servlet에 대해서 알아봐야겠다.

## 참조

### HTTP Cookie란

[https://ko.wikipedia.org/wiki/HTTP\\_%EC%BF%A0%ED%82%A4](https://ko.wikipedia.org/wiki/HTTP_%EC%BF%A0%ED%82%A4)

	<p>HTTP 쿠키 - 위키백과, 우리 모두의 백과사전</p> <p>ko.wikipedia.org</p>
---	--

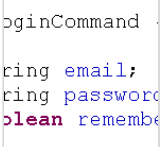
### HTTP Cookie 생성 및 Session 생성

[https://cinabrosite.wordpress.com/2017/01/24/cookie\\_session/](https://cinabrosite.wordpress.com/2017/01/24/cookie_session/)


	<p>쿠키와 세션에 대해 알아보자</p> <p>cinabrosite.wordpress.com</p>
--	---

### Spring Cookie, Session 예제

[http://clearpal7.blogspot.kr/2016/07/blog-post\\_75.html](http://clearpal7.blogspot.kr/2016/07/blog-post_75.html)

	<p>(스프링) 컨트롤러에서 쿠키 사용하기</p> <p>clearpal7.blogspot.com</p>
--	---

<http://ledgku.tistory.com/72>



국윤창

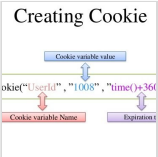
노력중인 게으름뱅이

[HTTP] 쿠키와 세션

CATEGORY

Cookie, Session 설명 자세히(영어)

<http://www.hackingarticles.in/beginner-guide-understand-cookies-session-management/>




Beginner Guide to Understand Cookies and Session Manage...

www.hackingarticles.in

Session Hijacking

<http://lanian.tistory.com/entry/HTTP-Session-Hijacking>



HTTP Session Hijacking

www.lanian.co.kr

공감

구독하기

'프로그래밍 > Web' 카테고리의 다른 글

- Web Server와 Web Application Server 차이 (0)
- 게시판 만들기 (0)
- Spring과 Cookie & Session (0)
- Mybatis & MySQL (0)
- DAO, DTO, Service (0)
- Annotation과 Bean (0)

댓글 0

여러분의 소중한 댓글을 입력해주세요

이름

비밀번호

☐ 비밀글

입력

공지사항

최근에 올라온 글

자바스크립트 클로저  
자바스크립트의 실행 컨텍..  
프로토타입 체이닝  
자바스크립트 this

최근에 달린 댓글

안녕하세요! rtp://127.0.0.1..  
안녕하세요.수신쪽 보다가 궁..  
안녕하세요 Jodu님 ㅎㅎ 맞습..  
좋은 자료 감사합니다. Spring..

Total  
71,067

Today 87  
Yesterday 232

링크

TAG

chunkMySQLJavaScript@Compon...spring batchBeanmybatis@AutowiredClosure@Qualifierunity클로저thymeleaf ...@Beanthymeleaf ...Barycentri...TaskletspringCheck poi...

more

2019/12						
일	월	화	수	목	금	토
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

글 보관함

2019/11 (6)  
2019/10 (1)  
2019/09 (1)  
2019/06 (1)