Spring MVC2 10 [로그인 - 쿠키 사용]

스프링 MVC 2편 - 백엔드 웹 개발 활용 기술

김영한

2022.10.22

로그인 구현 요구사항

홈화면 - 로그인 전 :회원가입 로그인

홈화면 - 로그인 후 :상품관리 로그아웃

보안 요구 사항

로그인 한 사용자만 상품에 접근, 관리 가능

로그인하지않은 사용자가 접근시 로그인 화면으로 redirect

도메인?

화면, UI, 기술 인프라 등등의 영역을 제외한 시스템이 구현해야 하는 핵심 비지니스 영역

도메인과 Web부분을 나누고 domain(핵심 영역) 과 web 의존이 domain을 의존하는 것은 가능하지만 반대로 domain이 web단을 의존하면 안된다.

회원관리에 사용될 MEMBER

```
@Data
public class Member {
   private Long id;
   @NotEmpty
   private String loginId; //로그인 ID
   @NotEmpty
   private String name; //사용자 이름
   @NotEmpty
   private String password;
}
```

기본적인 로그인을 구현 한 후에 로그인 됐을시에 홈화면을 변경하기

로그인 처리하기 - 쿠키 사용

홈을 요청 할때마다 memberId를 쿠키에 보내주는 방식으로 구현해보기.

-HttpServletResponse로 Cookie 담기

```
@PostMapping("/login")
public String login(@Valid @ModelAttribute LoginForm form, BindingResult
bindingResult, HttpServletResponse response){
   if(bindingResult.hasErrors()){
       return "login/loginForm";
   }

   Member loginMember = loginService.login(form.getLoginId(), form.getPassword());

   if(loginMember == null){
       bindingResult.reject("loginFail", "아이디 또는 비밀번호가 맞지 않습니다.");
       return "login/loginForm";
   }

   //로그인 성공 처리
   log.info("login success");
   //쿠키에 시간 정보를 주지 않으면 세션 쿠키( 브라우저 종료시 모두 종료 )
   Cookie idCookie = new Cookie("memberId", String.valueOf(loginMember.getId()));
   response.addCookie(idCookie);
   return "redirect:/";
}
```

HomeController 변경

기존 Home 매핑

```
@GetMapping("/")
public String home() {
   return "home";
```

```
}
```

쿠키를 사용한 **Home** 매핑

```
@GetMapping("/")
public String homeLogin(@CookieValue(name = "memberId", required = false) Long
memberId, Model model){

   if(memberId == null){
       return "home";
   }
   //로그인
   Member loginMember = memberRepository.findById(memberId);
   if(loginMember==null){
       return "home";
   }

   model.addAttribute("member", loginMember);
   return "loginHome";
}
```

쿠키를 확인 후 repository에서 확인 후 로그인홈에 들어가게 된다.

`@CookieValue(name = "memberId", required = false) Long memberId`

애노테이션으로 쉽게 가져올 수 있다.

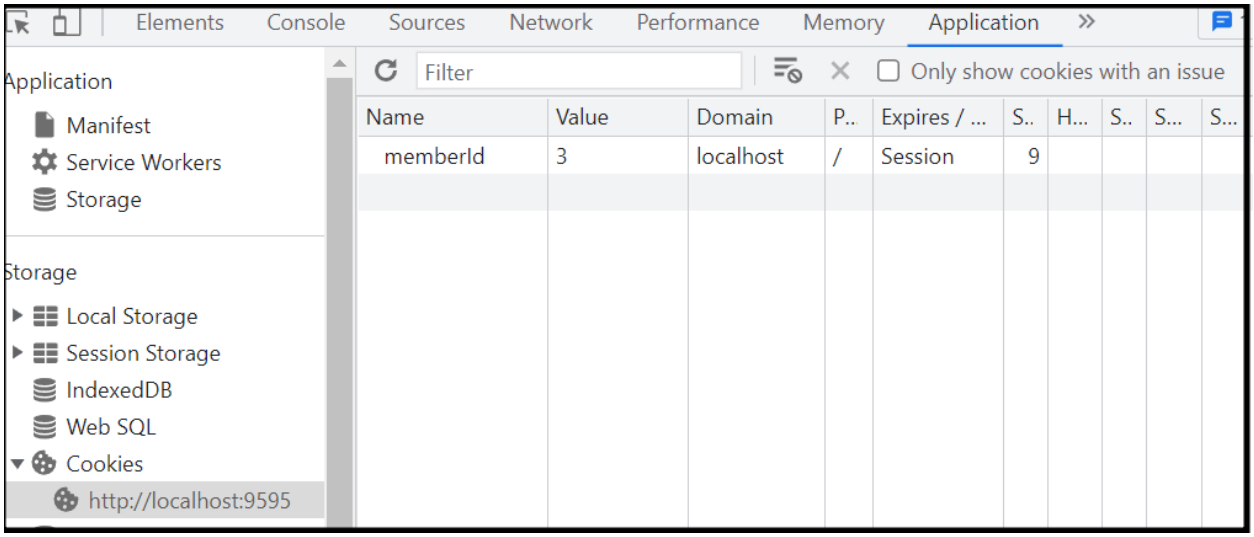
로그아웃 처리 - 쿠키 만료

```
@PostMapping("/logout")
public String logout(HttpServletResponse response){
   expireCookie(response, "memberId");
   return "redirect:/";
}

private void expireCookie(HttpServletResponse response, String cookieName) {
   Cookie cookie = new Cookie(cookieName, null);
   cookie.setMaxAge(0);
   response.addCookie(cookie);
}
```

이렇게 구현하면 심각한 보안문제들이 발생한다**!**

- 쿠키값은 임의로 변경이 가능하다



쿠키값을 바로 변경시킬 수 있다.

- 쿠키에 보관된 정보는 훔쳐 갈 수 있다.

  웹 브라우저에 보관이 되고 열람이 쉽기 때문에 그대로 가져갈 수 있다. HTTP 전송구간에서도 털릴 수 있다.

대안법

- 쿠키에 중요한 값을 노출하지 않고, 예측 불가능한 토큰을 노출하고 서버에서 토큰과 사용자 Id를 매핑해서 인식한다.(토큰은 서버에서 관리)
- 토큰은 예측 불가능한 값이어야 한다.
- 토큰의 만료시간을 짧게 유지한다. 해킹이 의심되면 토큰을 강제로 제거 시킨다.
