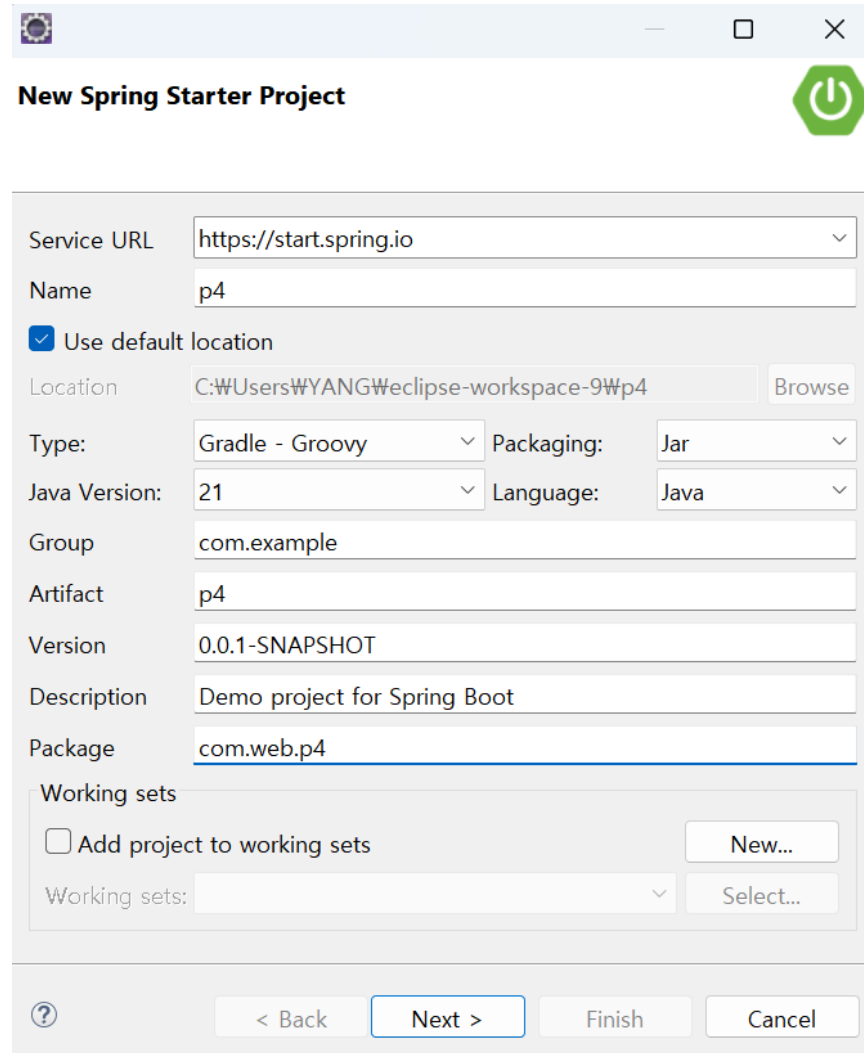


웹 프로그래밍

# 4장. 쿠키와 세션

동아대학교 컴퓨터공학과  
양 선

# p4 프로젝트 생성



The image shows a 'New Spring Starter Project' dialog box. It has a title bar with a gear icon and standard window controls. The title is 'New Spring Starter Project' with a green power button icon on the right. The dialog contains several input fields and checkboxes. The 'Service URL' is set to 'https://start.spring.io'. The 'Name' is 'p4'. The 'Use default location' checkbox is checked. The 'Location' is 'C:\Users\WYANG\workspace-9Wp4' with a 'Browse' button. The 'Type' is 'Gradle - Groovy', 'Packaging' is 'Jar', 'Java Version' is '21', and 'Language' is 'Java'. The 'Group' is 'com.example', 'Artifact' is 'p4', 'Version' is '0.0.1-SNAPSHOT', 'Description' is 'Demo project for Spring Boot', and 'Package' is 'com.web.p4'. There is a 'Working sets' section with an unchecked checkbox 'Add project to working sets' and a 'New...' button. Below it is a 'Working sets:' dropdown and a 'Select...' button. At the bottom are buttons for '?', '< Back', 'Next >', 'Finish', and 'Cancel'.

**New Spring Starter Project**

Service URL:

Name:

☒ Use default location

Location:

Type:  Packaging:

Java Version:  Language:

Group:

Artifact:

Version:

Description:

Package:

Working sets

☐ Add project to working sets

Working sets:

## 쿠키/세션의 필요성 (자료 보관하는 제3의 장소의 필요성)

- ❖ 3장에서 배웠던 데이터 전송은 오직 한 단계
  - 여러 단계 후까지 데이터를 넘기려면?
  - 중간 화면들은 데이터를 계속 넘겨주기 해야 함
- ❖ 쿠키/세션은 제3의 장소 역할 (즉, 장바구니 역할)
  - 서버와 접속 후에 발생된 고객의 정보를 저장
  - 그 정보를 클라이언트가 가지고 있으면 **쿠키**
    - ✓ 정보가 과자 부스러기처럼 클라이언트 컴퓨터에 남는다고 해서 쿠키라 불림
  - 서버가 가지고 있으면 **세션**

# 쿠키 VS 세션

- ❖ 쿠키를 사용하면 서버에 무리를 주지 않는다는 장점
- ❖ 반면 쿠키는 보안에 약하고 데이터 사이즈 작다는 단점
  - 공유 PC 사용 시 쿠키에 저장된 정보를 타인이 알아낼 수 있음!
  - 또한 저장할 수 있는 데이터 사이즈도 한계가 있음
    - ✓ 웹브라우저마다 조금씩 차이가 있긴 하지만, 예를 들어 최대 4KB 쿠키를 300개 이하로만 저장 가능하다 이런 식으로 제한을 둬
    - ✓ 이 경우 쿠키를 사용할 수 있는 최대 용량은  $4KB * 300개 = 1.2MB$
- ❖ 세션을 사용하여 이러한 단점 극복
  - 서버에서 관리하므로 **보안 강력하게 유지할 수 있음**
  - 또한 저장할 수 있는 데이터 사이즈도 매우 큼

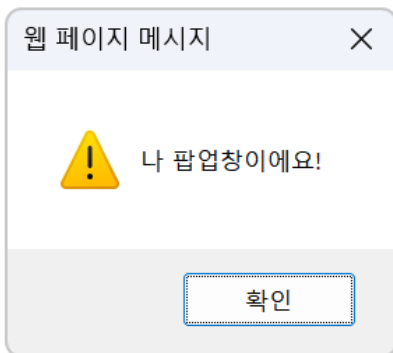
# 세션(session) 코딩 시 자주 헷갈리는 부분

코딩	설명
public String methodName(HttpSession se, ... ) { ... }	<ul style="list-style-type: none"><li>매개변수에 HttpSession 넣어 줌</li><li>세션 사용하겠다는 의미</li></ul>
se.setAttribute("mid", mid); se.setAttribute("won", won);	<ul style="list-style-type: none"><li>세션에 한 개 보관하기</li><li>mo.addAttribute("mid", mid); 랑 비슷</li></ul>
String mid = (String)se.getAttribute("mid"); Integer won = (Integer)se.getAttribute("won");	<ul style="list-style-type: none"><li>세션에 들어있는 항목 1개를 지역변수에 대입</li><li>리턴 타입이 Object이므로 <b>형변환 필요</b></li><li>단, 지역변수 거치지 않고 바로 모델에 넣을 때는 양쪽 다 Object이므로 형변환 필요 없음</li></ul> <pre>mo.addAttribute("mid", se.getAttribute("mid"))</pre>
se.removeAttribute("mid");	<ul style="list-style-type: none"><li>세션에서 한 개 제거</li></ul>
se.invalidate();	<ul style="list-style-type: none"><li>세션 무효화 (로그아웃)</li></ul>

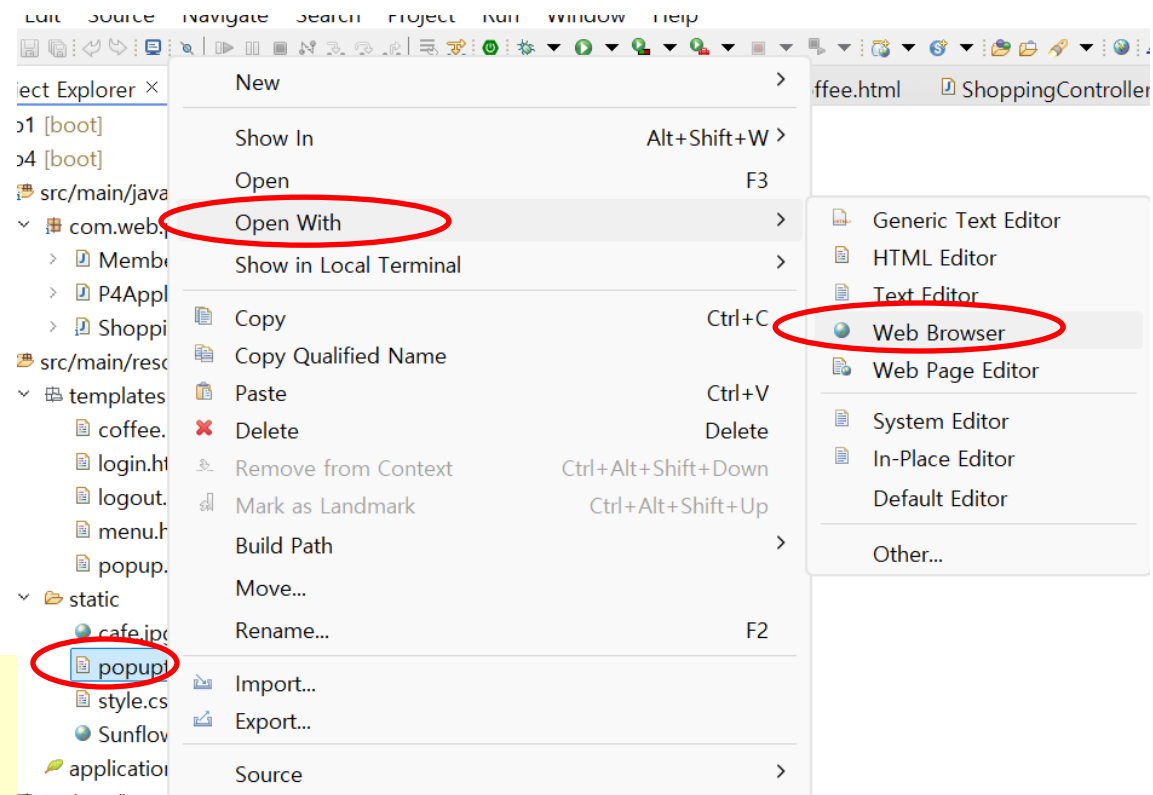
# 여기서 잠깐 팝업창 띄워보기!

- ✓ 세션 사용한 예제 구현하기 전에 잠깐 javascript 로 팝업창을 띄우는 걸 공부하겠습니다
  - templates 혹은 static폴더에 popuptest.html 코딩 후 (나중에 소스 삭제)
  - 우측 마우스 클릭 → Open With → Web Browser (서버구동 없이 실행. 1장 메모장 코딩이랑 동일)

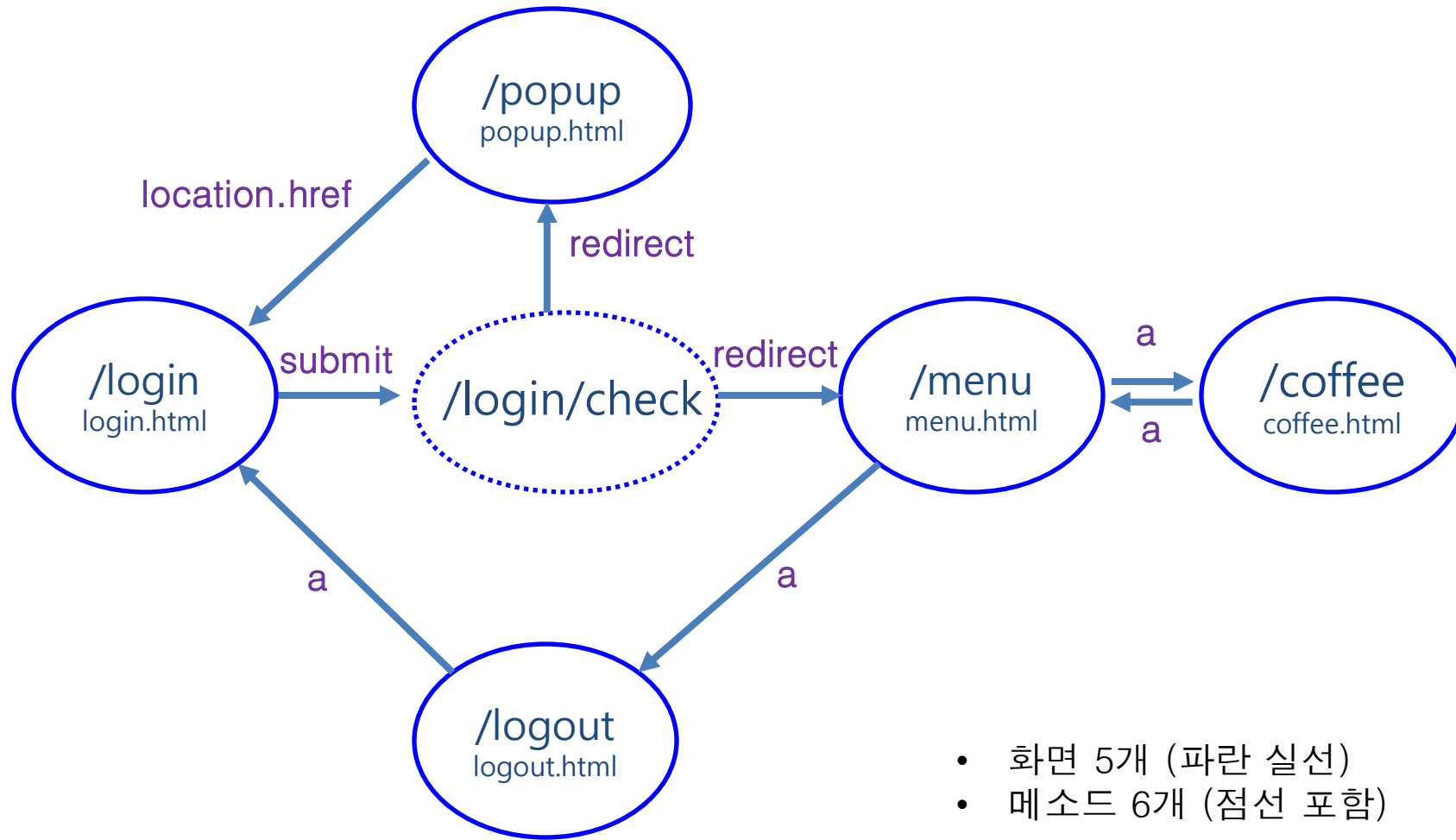
```
<!DOCTYPE html>
<html>
<head><meta charset="UTF-8">
<title>팝업창 테스트</title></head>
<body>
<script>
    alert("나 팝업창이에요!");
    location.href="/";
</script>
</body>
</html>
```



<script> ~ </script>는 자바스크립트입니다.  
alert은 확인버튼 1개만 있는 팝업창입니다.  
location.href 를 이용해서 다른 페이지로 이동합니다.



# 세션을 사용하는 간단한 로그인/로그아웃 예제 구현



# 회원이 아닌 경우

front-end	back-end
localhost:8080/login 접속	
	/login 메소드 실행. 로그인화면 보내줌
로그인화면 뜨면 아이디 입력 후 submit	
	/login/check 메소드 실행 아이디 비번 체크 후 만약 없는 회원이면 /popup 메소드로 redirect (이 때 팝업창에 뜰 메시지를 parameter로 보내줌)
	/popup 메소드 실행. 팝업창화면 보내줌
팝업창화면 뜨고 확인버튼 클릭	
	/login 메소드 실행. 로그인화면 보내줌
로그인화면 뜸	



# 회원인 경우

front-end	back-end
localhost:8080/login 접속	
	/login 메소드 실행 . 로그인화면 보내줌
로그인화면 뜨면 아이디 입력 후 submit	
	/login/check 메소드 실행. 아이디 비번 체크 후 회원 맞으면 <b>세션에 아이디 저장</b> 후 /menu 메소드로 redirect
	/menu 메소드 실행. 메뉴화면 보내줌
메뉴화면 뜨면 커피 클릭	
	/coffee 메소드 실행. 커피화면 보내줌
커피화면 뜨면 <u>메뉴로 돌아가기</u> 클릭	
	/menu 메소드 실행. 메뉴화면 보내줌
메뉴화면 뜨면 로그아웃 클릭	
	/logout 메소드 실행. 세션 무효화 시킨 후 로그아웃화면 보내줌
로그아웃 화면 뜨면 그림 클릭	
	/login 메소드 실행 . 로그인화면 보내줌
로그인화면 뜸	

# ShoppingController.java

```
package com.web.p4;

import java.util.ArrayList;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.RequestParam;

import jakarta.servlet.http.HttpSession;

@Controller
public class ShoppingController {

    /* 여기 메소드 추가 */

} // class
```

# style.css

CSS 파일 분리

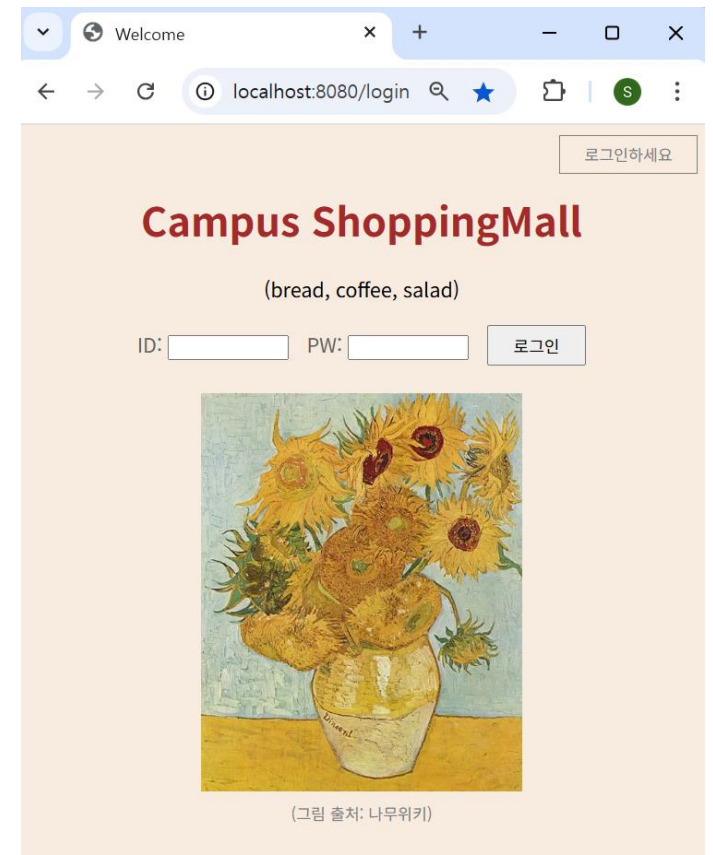
(1) 디자인 공유

(2) html소스가 길어지는 거 방지

```
#loginInfo { float: right; border:1px solid gray; /* 우측 작은 회색라인 네모 */
              width: 100px; height:20px; padding: 5px;
              color:gray; font-size:0.8em;}
label { color: RGB(100, 100, 100); } /* 블랙에 가까운 회색 */
input { width: 90px;
        margin:10px 10px 10px 0px; } /* top부터 시계방향 */
button { padding: 5px 20px 5px 20px; }
img { max-width: 320px; /* 그림 원본 비율 살림 */
      max-height: 320px; }
#s1 { color: blue; } /* id는 파란색으로 */
.s2 { color: gray; /* 그림 출처 등의 부가설명은 회색 작은 글씨 */
     font-size: 0.8em; } /* 한 화면에 부가설명 여러 개일 수도 있으니 s2는 클래스로 */
hr { border: 2px solid darkgray; } /* menu화면에 가로줄 */
li { line-height: 1.5em; }
li::marker { color: red; }
```

# 로그인 (login.html)

```
<!DOCTYPE html><html xmlns:th="http://www.thymeleaf.org">
<head><meta charset="UTF-8"> <title>Welcome</title>
<link rel="stylesheet" href="/style.css">
<style> body {background:#F8ECE0; text-align: center;}
    h1 {color:brown;}
</style></head><body>
<div id="loginInfo" th:text="${loginInfo}"></div><br>
<h1>Campus ShoppingMall</h1>
(bread, coffee, salad)<p>
<form method="get" action="/login/check">
    <label for="mid">ID:</label>
    <input type="text" name="mid" id="mid">
    <label for="pw">PW:</label>
    <input type="text" name="pw" id="pw">
    <button type="submit">로그인</button>
</form><p>
<br>
<span class="s2">(그림 출처: 나무위키)</span>
</body></html>
```



- body에 설정된 text-align은 table 안에 있는 글자는 가운데정렬 시킬 수 있지만 table 자체는 가운데정렬 안 됩니다. (나중에 다른 방법 사용)

- 화면에서 label 글자를 클릭하면 for 속성값에 있는 객체에 포커스가 갑니다.

```
@GetMapping("/login")
public String login(HttpSession se, Model mo) {
    String mid = (String)se.getAttribute("mid");
    mo.addAttribute("loginInfo", mid==null?"로그인하세요":mid+"님");
    return "login";
}
```

```
@GetMapping("/login/check")
public String loginCheck(HttpSession se, @RequestParam("mid") String mid,
                        @RequestParam("pw") String pw) {
    var arr = new ArrayList<Member>();
    arr.add(new Member("고흐", "g"));    arr.add(new Member("james", "j"));
    arr.add(new Member("dooli", "d"));  arr.add(new Member("iu", "i"));

    boolean ourMember=false;
    for(var a: arr) {
        if(a.mid.equals(mid) && a.pw.equals(pw))
            { ourMember = true;    break;    }
    }

    if( ourMember ) {
        se.setAttribute("mid", mid);
        return "redirect:/menu";
    }
    else {
        se.setAttribute("msg", mid + "는 미등록 아이디이거나 혹은 패스워드가 일치하지 않습니다.\n"
                        + "확인 후 로그인 부탁드립니다.");
        return "redirect:/popup";
    }
}

} // method
```

• 지금은 회원 4명 하드코딩 했지만, 나중에는 데이터베이스에서 회원정보 가지고 오게 됩니다.

• 지금까지는 return 다음에 뷰템플릿 파일 이름이 있었는데 redirect 콜론 다음에 주소가 나왔습니다.

• 예를 들어 return "redirect:/menu"; 한 줄의 의미는 컨트롤러 안에 있는 @GetMapping("/menu") 메소드로 주도권을 넘긴다는 의미입니다.

# Member.java

- 일단 회원정보에 아이디, 비밀번호만 있다고 가정

```
package com.web.p4;

public class Member {

    public String mid;          /* 필드 2개 */
    public String pw;

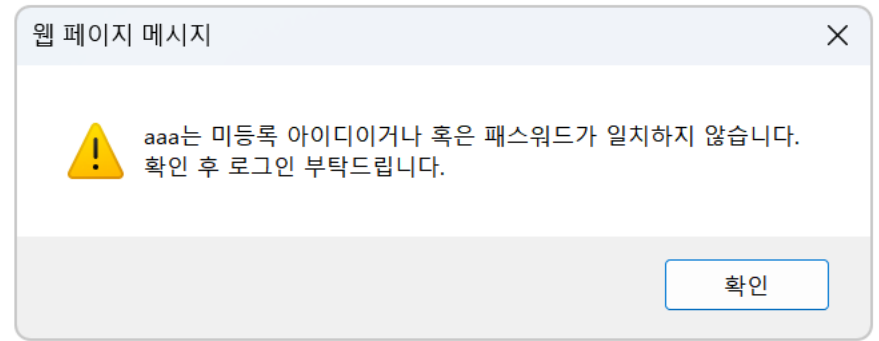
    public Member(String mid, String pw) { /* 생성자 */
        this.mid = mid;
        this.pw = pw;
    }

} // class
```

# 팝업 (popup.html)

```
1 <!DOCTYPE html>
2 <html xmlns:th="http://www.thymeleaf.org">
3 <head><meta charset="UTF-8">
4 <title>안내글</title></head>
5 <body>
6
7 <script th:inline="javascript">
8     alert([[${msg}]]);
9     location.href="/login";
10 </script>
11
12 </body>
13 </html>
```

```
@GetMapping("/popup")
public String popup(HttpSession se, Model mo) {
    mo.addAttribute("msg", se.getAttribute("msg"));
    se.removeAttribute("msg");
    return "popup";
}
```



• JavaScript는 태그를 사용하지 않기 때문에 속성에 해당하는 th:text 사용 못함. 이런 경우 [[ ~ ]] 사용하면 됩니다.

• 8번 라인에 **빨간 오류표시** 있어도 웹서버 구동에는 아무런 문제가 없습니다. 빨간 오류표시가 신경쓰이시면 좌측 코드의 7~10번 라인을 삭제하고 대신 아래 내용을 타이핑해 주세요.

```
<script th:inline="javascript">
    /**/
    let msg = /*[[${msg}]]*/ 'default';
    alert(msg);
    location.href="/login";
    /*]]&gt;*/
&lt;/script&gt;</pre></div><div data-bbox="449 938 545 964" data-label="Page-Footer"><p>4장 쿠키와 세션</p></div><div data-bbox="927 938 953 964" data-label="Page-Footer"><p>15</p></div>
```

# 메뉴 (menu.html)

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head><meta charset="UTF-8"> <title>MENU</title>
<link rel="stylesheet" href="/style.css">
<style>
    body {padding: 0px 40px 0px 40px;
          background:lightgray; }
</style>
</head><body>
<h2>MENU</h2><hr>
<span id="s1" th:text="${mid}">mid</span>님,
오늘도 즐겁게 쇼핑하세요! <br><hr>
<ul>
    <li>bread
    <li><a href="/coffee">coffee</a>
    <li>salad
    <li><a href="/logout">logout</a>
</ul>
</body></html>
```

## MENU

dooli님, 오늘도 즐겁게 쇼핑하세요!

- bread
- [coffee](#)
- salad
- [logout](#)

- <span>과 </span> 사이에 mid를 넣은 이유는?  
(어차피 타임리프 변수보다 우선순위 밀릴 텐데??)  
➔ 프론트엔드 레벨에서 실행할 때 중요

- <ul> 은 글머리기호 <li>은 번호매기기

```
@GetMapping("/menu")
public String menu(HttpSession se, Model mo) {
    mo.addAttribute("mid", se.getAttribute("mid"));
    return "menu";
}
```



# 커피 화면 (coffee.html)

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head><meta charset="UTF-8"><title>coffee</title>
<style>
  body { background-image: url(
https://upload.wikimedia.org/wikipedia/commons/thumb/
e/e4/Latte_and_dark_coffee.jpg/1200px-
Latte_and_dark_coffee.jpg  ); }
  h1, h3 { color:lightcyan; }
  #s1 { color:cyan; }
  a { background: lightyellow;}
</style></head>
<body>
<h1><span id="s1" th:text="${mid}">mid</span>님,
원하시는 커피를 선택하세요</h1>
<h3>Coming soon ...</h3>
<a href="/menu">메뉴로 돌아가기</a>
</body></html>
```



- background-image: url( ) 은 화면 배경 그림을 설정합니다. (그림 출처: 위키피디아)
- 쿠키/세션 설명을 위해 만든 미완성 화면입니다. 나중에 시간 여유가 되면 커피 뿐만 아니라, 빵, 샐러드도 판매하는 쇼핑몰 사이트를 구현해 봅시다!

```
@GetMapping("/coffee")
public String coffee(HttpSession se, Model mo) {
    mo.addAttribute("mid", se.getAttribute("mid"));
    return "coffee";
}
```

# 로그아웃(logout.html)

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head><meta charset="UTF-8"><title>logout</title>
<link rel="stylesheet" href="/style.css">
<style> body {background:#F3E2A9; text-align:center; }
</style></head>
<body>
<span id="s1" th:text="${mid}">mid</span> 님께서 로그아웃하셨습니다.<p>
다음에 또 만나요!!<p>
<span class="s2">(아래 그림을 클릭하시면 첫 화면으로 이동합니다.)</span>
<br>
<a href="/login">
  
</a><br>
<span class="s2">(그림 출처: 나무위키)</span>
</body></html>
```

```
@GetMapping("/logout")
public String logout(HttpSession se, Model mo) {
    mo.addAttribute("mid", se.getAttribute("mid"));
    se.invalidate();
    return "logout";
}
```

dooli 님께서 로그아웃하셨습니다.

다음에 또 만나요!!

(아래 그림을 클릭하시면 첫 화면으로 이동합니다.)



(그림 출처: 나무위키)

- <a>와 </a> 사이에 그림이 있으므로 그림을 클릭하면 다른 주소로 넘어갑니다.