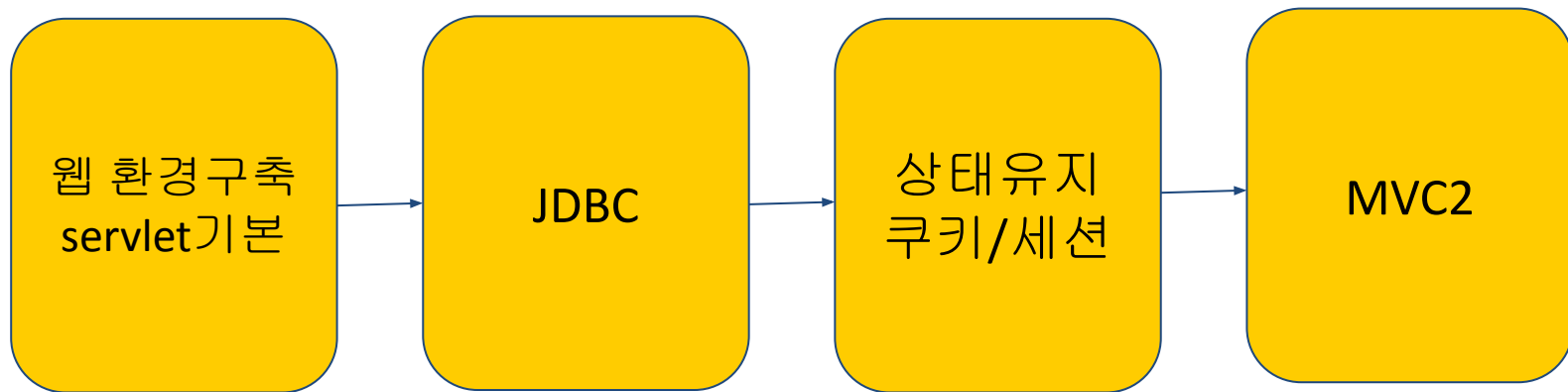
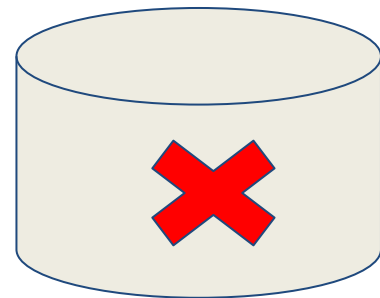
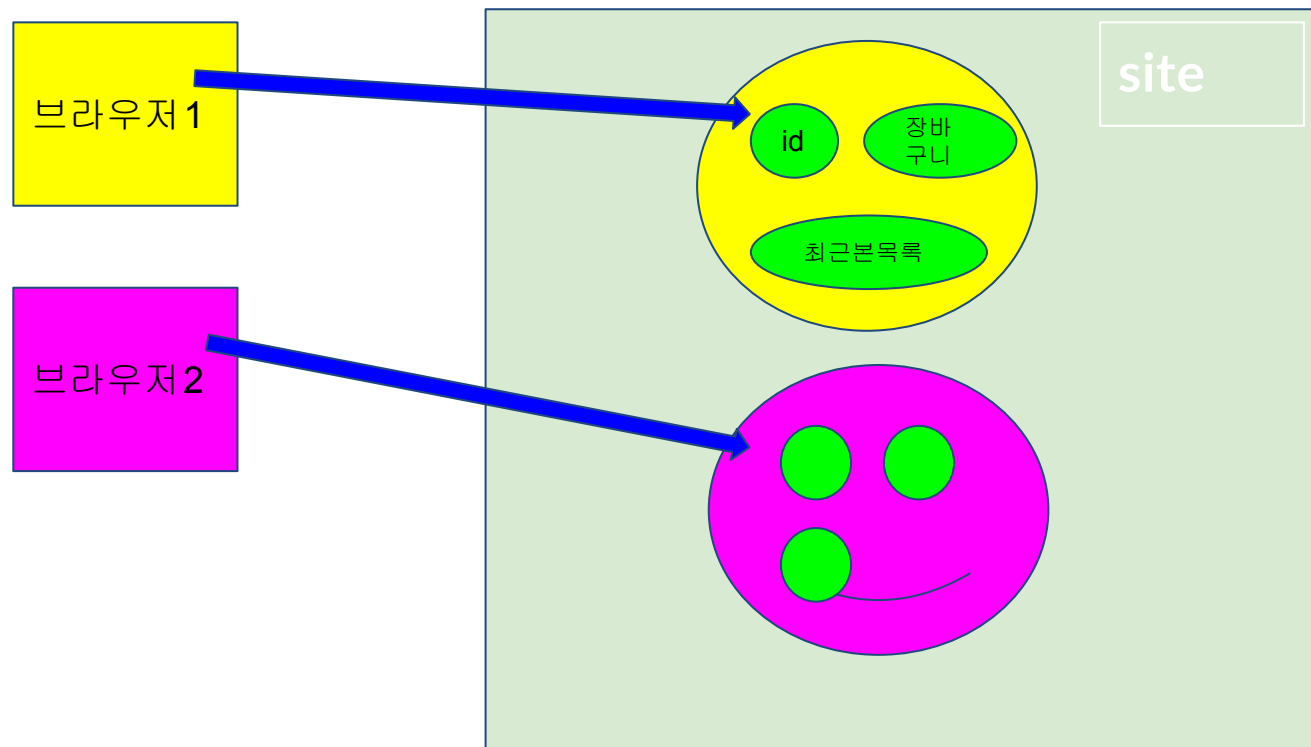


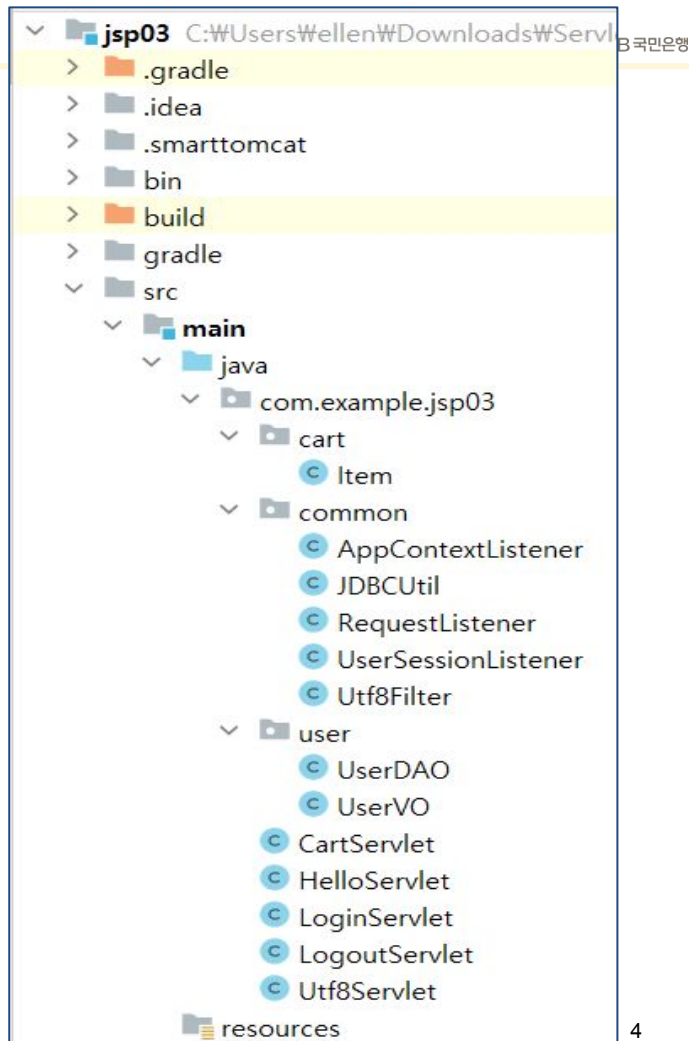
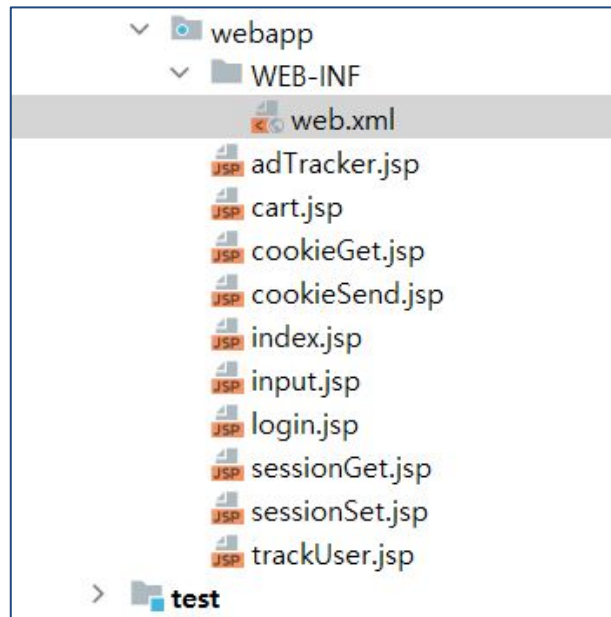
2024년 상반기 K-디지털 트레이닝

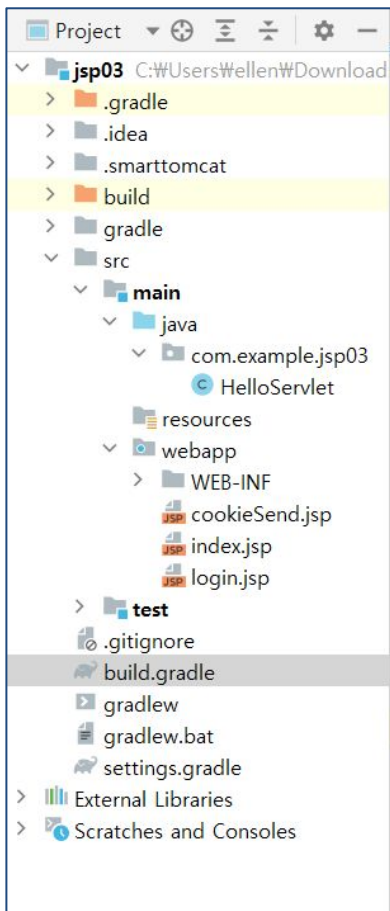
JSP/Servlet 상태정보

[KB] IT's Your Life









```
sourceCompatibility = '17'
targetCompatibility = '17'
```

```
tasks.withType(JavaCompile) {
    options.encoding = 'UTF-8'
}
```

```
dependencies {
    // JSP, SERVLET, JSTL
    implementation('javax.servlet:javax.servlet-api:4.0.1')
    compileOnly 'javax.servlet.jsp:jsp-api:2.1'
    implementation 'javax.servlet:jstl:1.2'
```

```
//lombok plugin
implementation 'org.projectlombok:lombok:1.18.26' // 최신 버전 사용
annotationProcessor 'org.projectlombok:lombok:1.18.26' // 최신 버전 사용
```

```
//driver
implementation 'mysql:mysql-connector-java:8.0.30' // MySQL 드라이버 예제
```

```
testImplementation("org.junit.jupiter:junit-jupiter-api:${junitVersion}")
testRuntimeOnly("org.junit.jupiter:junit-jupiter-engine:${junitVersion}")
}
```

쿠키

The screenshot shows the Oracle website with a cookie consent modal in the foreground. The modal is titled "해당 사이트의 쿠키 관련 선택사항" (Cookie-related selection for this site) and contains the following text:

당사의 사이트를 방문함으로써 Oracle과 당사는 **제3자 파트너**는 다음과 같은 목적으로 쿠키 및 기타 유사한 기술을 사용하여 사용자 데이터를 수집할 수 있습니다.

- 필수 쿠키: 로그인 세부 정보를 기억하고 보안 프로세스를 검증하기 위한 사이트 기능에 필요합니다.
- 기능 쿠키: 통계를 수집하거나, 소셜 미디어와 다른 기능 설정을 저장하거나, 당사의 사이트의 맞춤을 저장할 수 있도록 사이트를 최적화하는 데 사용됩니다.
- 광고 쿠키: Oracle 및 당사의 서비스와 파트너가 사용자의 브라우저 및 디바이스 관례에 걸쳐 관심 기반 광고와 맞춤형 콘텐츠를 제공할 수 있도록 사용됩니다.

자세한 내용은 당사와 **개인정보 보호정책**을 참조하십시오.

At the bottom of the modal are two buttons: "동의 쿠키 쿠키" (Accept cookies) and "쿠키 기본 설정 보기 및 변경" (View and change cookie settings).

The background of the screenshot shows the Oracle website layout with sections for "Oracle Cloud Infrastructure", "Oracle Cloud Applications", and "Oracle Cloud Free Tier".

belabef
ABOUT
STORE
REVIEW
COMMUNITY

장바구니

장바구니

Items	위사	수량	배송수단	배송비	가격	
<p> [노블레트리스] 33만입축볼 슈퍼싱글 원 독신한 하리 침대도퍼 2IN1 차막 램프머드 · 노블레트리스 33(슈퍼싱글) 1EA (한정수량특가 40% 할인/무료배송 4500원 책) / 1개 </p>	♡	1개 변경	택배	무료	₩84,000	주문 삭제
<p> 변기청소 이젠 자동으로! 변기 속 세균, 때, 냄새 자동 청소. 닥터파일 유에잇 자동 변기 살균 세정기(디지털 변기 세정기) · [8월 이벤트] 유에잇 거품세드 (하루 30개 한정) / 1개 </p>	♡	1개 변경	택배	무료	₩54,800	주문 삭제
					상품가격	₩138,800
					배송비	무료
					적합액결제 수수료	-
					결제금액	₩138,800

선택상품 삭제
위사리스트 닫기

계속 쇼핑하기
주문하기

NAVER
1월 10일 10시 00분부터
1월 10일 10시 00분부터

Pay Plus

상태정보 유지의 필요성

- 웹 프로젝트에서 상태정보 유지(State Management)는 사용자의 활동 및 애플리케이션의 현재 상태를 기억하고 유지하는 것을 의미
- 웹 애플리케이션이 사용자가 페이지 간 이동하거나 새로 고침을 해도 그 이전의 상태를 잃지 않도록 하는 중요한 기능

방법	저장 위치	특징 및 사용 예시
세션(Session)	서버	로그인 상태, 장바구니 정보
쿠키(Cookie)	클라이언트	사용자 설정, 인증 정보
로컬 스토리지(Local Storage)	클라이언트	브라우저를 닫아도 데이터 유지, 대용량 데이터 저장
세션 스토리지(Session Storage)	클라이언트	브라우저를 닫으면 데이터 삭제, 단기 데이터 저장
상태관리 라이브러리(State Management Libraries)	클라이언트	리덕스, 모부엑스 등, 복잡한 상태 관리
URL 파라미터 및 해시(Hash)	클라이언트	간단한 상태 유지, 검색 결과, 필터 설정

- 클라이언트에 저장되며, 설정된 만료 날짜까지 유지
- 주로 사용자의 설정 및 선호도, 로그인 정보 등을 저장하는 데 사용
- 비교적 보안에 취약하므로 중요한 정보는 저장하지 않는 것이 좋음.

항목	쿠키(Cookie)	세션(Session)
저장 위치	클라이언트(브라우저)	서버
생명 주기	설정된 만료 날짜까지 또는 브라우저 종료	브라우저 종료 시 또는 타임아웃 발생 시
크기 제한	약 4KB	서버 메모리/저장 공간에 따라 다름
보안	비교적 취약	비교적 안전
사용 예시	로그인 정보, 사용자 설정, 광고 추적	로그인 상태, 장바구니, 일시적인 사용자 정보
전송 방식	HTTP/HTTPS 요청 시마다 전송	세션 ID만 전송

메서드	설명	사용 예시
<code>Cookie(String name, String value)</code>	쿠키 객체를 생성	<code>Cookie cookie = new Cookie("username", "john123");</code>
<code>String getName()</code>	쿠키의 이름을 반환	<code>String name = cookie.getName();</code>
<code>String getValue()</code>	쿠키의 값을 반환	<code>String value = cookie.getValue();</code>
<code>void setValue(String newValue)</code>	쿠키의 값을 설정	<code>cookie.setValue("john456");</code>
<code>int getMaxAge()</code>	쿠키의 최대 유효 기간(초)을 반환	<code>int maxAge = cookie.getMaxAge();</code>
<code>void setMaxAge(int expiry)</code>	쿠키의 최대 유효 기간(초)을 설정	<code>cookie.setMaxAge(60*60*24*7); // 1주일</code>
<code>String getPath()</code>	쿠키의 경로를 반환	<code>String path = cookie.getPath();</code>
<code>void setPath(String uri)</code>	쿠키의 경로를 설정	<code>cookie.setPath("/app");</code>
<code>String getDomain()</code>	쿠키의 도메인을 반환	<code>String domain = cookie.getDomain();</code>
<code>void setDomain(String pattern)</code>	쿠키의 도메인을 설정	<code>cookie.setDomain("example.com");</code>
<code>boolean getSecure()</code>	쿠키가 보안 설정이 되었는지 여부를 반환	<code>boolean secure = cookie.getSecure();</code>
<code>void setSecure(boolean flag)</code>	쿠키의 보안 설정을 설정	<code>cookie.setSecure(true);</code>
<code>boolean isHttpOnly()</code>	쿠키가 HTTP 전용인지 여부를 반환	<code>boolean httpOnly = cookie.isHttpOnly();</code>
<code>void setHttpOnly(boolean isHttpOnly)</code>	쿠키를 HTTP 전용으로 설정	<code>cookie.setHttpOnly(true);</code>
<code>String getComment()</code>	쿠키의 주석을 반환	<code>String comment = cookie.getComment();</code>
<code>void setComment(String purpose)</code>	쿠키의 주석을 설정	<code>cookie.setComment("User login cookie");</code>
<code>int getVersion()</code>	쿠키의 버전을 반환	<code>int version = cookie.getVersion();</code>
<code>void setVersion(int v)</code>	쿠키의 버전을 설정	<code>cookie.setVersion(1);</code>

쿠키 설정(이름+값)

- 쿠키 정보(이름+값) 클라이언트에 저장

```
//쿠키는 브라우저에 텍스트로 저장되어야 하기 때문에 문자열만 허용한다.
Cookie c1 = new Cookie("name", "honggildong"); //name, value
Cookie c2 = new Cookie("age", "100");
response.addCookie(c1); //브라우저에게 쿠키를 심으라고 명령함.
response.addCookie(c2);
```

- 쿠키 정보(이름+값) 클라이언트로 부터 가지고 오기

```
Cookie[] cookies = request.getCookies();
for(Cookie c: cookies){
    //out: 브라우저에 텍스트로 프린트하고자 하는 경우
    out.print(c.getName() + ", " + c.getValue() + "<br>");
}
```

쿠키배열

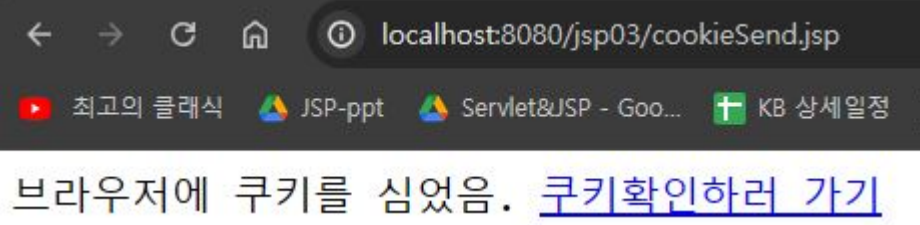
```
JSESSIONID, 6CCA2DAD80BF0C9198B6BB511B755FF1
name, honggildong
age, 100
쿠키의 개수: 3
```

The screenshot shows a web browser with the URL `localhost:8080/web30/page02.jsp`. The page content displays "honggildong 100 축구". The developer tools are open to the Application tab, showing the Cookies section expanded. The cookies table lists the following data:

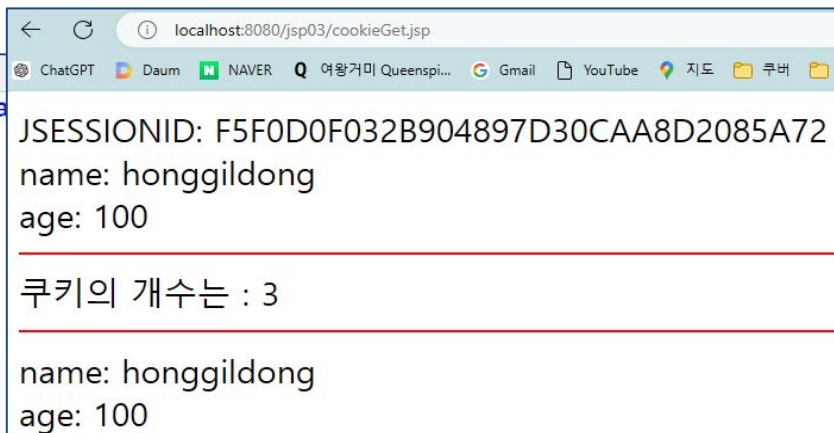
Name	Value	Domain	Path	Expires	Size	HttpOnly	Secure	SameSite
JSESSIONID	90D5B6DAA41E0ED413D6316623B1...	localhost	/web30	Session	42	✓		
type	축구	localhost	/web30	Session	10			
age	100	localhost	/web30	Session	6			
name	honggildong	localhost	/web30	Session	15			

```

1  <%@ page contentType="text/html;cha
2  <html>
3  <head>
4      <title>Title</title>
5  </head>
6  <body>
7  <%
8      Cookie c1 = new Cookie("name", "honggildong"); //name, value
9      Cookie c2 = new Cookie("age", "100"); //String, String
10     response.addCookie(c1); //브라우저에게 쿠키를 심으라고 명령함.
11     response.addCookie(c2);
12 %>
13 <body>
14     브라우저에 쿠키를 심었음. <a href="cookieGet.jsp">쿠키확인하러 가기</a>
15 </body>
16 </html>
17 </body>
18 </html>
    
```




```
cookieGet.jsp
1  <%@ page contentType="text/html; charset=UTF-8" language="java" %>
2  <!DOCTYPE html>
3  <html>
4  <head>
5      <meta charset="UTF-8">
6      <title>Insert title here</title>
7  </head>
8  <body>
9  <%
10     Cookie[] cookies = request.getCookies();
11     for(Cookie c: cookies){
12         out.print(c.getName() + ": " + c.getValue() + "<br>");
13     }
14 %>
15 <hr color="red">
16 쿠키의 개수는 : <%= cookies.length %>
17 <hr color="red">
18 <%
19     for(Cookie c: cookies){
20         if(!c.getName().equals("JSESSIONID")){
21             out.print(c.getName() + ": " + c.getValue() + "<br>");
22         }
23     }
24 %>
25 </body>
26 </html>
```



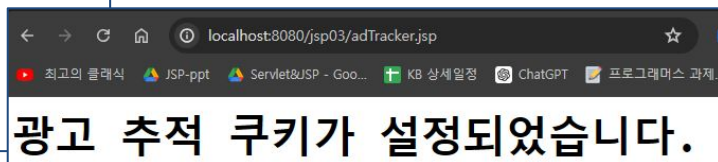
광고 추적의 예시

- 예시 시나리오
 1. 사용자가 웹사이트를 방문함.
 2. 서버는 사용자의 브라우저에 광고 추적 쿠키를 설정함.
 3. 사용자가 웹사이트 내에서 활동을 함.
 4. 사용자가 다시 웹사이트를 방문할 때, 서버는 쿠키를 읽어 사용자의 활동을 분석함.

광고 추적의 예시

- 사용자가 처음 웹사이트를 방문할 때 광고 추적 쿠키를 설정하는 페이지
- 쿠키의 이름은 **adTracker**이며, 값은 사용자 고유 식별자로 설정

```
adTracker.jsp
1  <%@ page language="java" contentType="text/html; charset=UTF-
2  <%
3      // 광고 추적 쿠키 설정
4      String cookieName = "adTracker";
5      String cookieValue = "user12345"; // 사용자 고유 식별자
6      int maxAge = 60 * 60 * 24 * 30; // 쿠키 유효기간 30일
7
8      Cookie adCookie = new Cookie(cookieName, cookieValue);
9      adCookie.setMaxAge(maxAge);
10     response.addCookie(adCookie);
11 %>
12 <html>
13 <head>
14     <title>쿠키 설정</title>
15 </head>
16 <body>
17     <h1>광고 추적 쿠키가 설정되었습니다.</h1>
18 </body>
19 </html>
```



- 사용자가 웹사이트를 다시 방문할 때 쿠키를 읽어 사용자의 활동을 추적하는 페이지
- 쿠키가 존재하면 환영 메시지와 함께 추적 메시지를 표시하고, 쿠키가 없으면 새로운 사용자로 인식

```
trackUser.jsp
1 <%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding=
2 <%
3 // 광고 추적 쿠키 읽기
4 Cookie[] cookies = request.getCookies();
5 String adTracker = null;
6
7 if (cookies != null) {
8     for (Cookie cookie : cookies) {
9         if (cookie.getName().equals("adTracker")) {
10             adTracker = cookie.getValue();
11             break;
12         }
13     }
14 }
15
16 if (adTracker != null) {
17     // 사용자가 방문했음을 추적
18     out.println("<h1>환영합니다, 사용자 " + adTracker + "!</h1>");
19     out.println("<p>당신의 활동을 추적하고 있습니다.</p>");
20 } else {
21     out.println("<h1>환영합니다, 새로운 사용자!</h1>");
22     out.println("<p>당신의 광고 추적 쿠키가 없습니다.</p>");
23 }
24 %>
25 <html>
26 <head>
27     <title>사용자 추적</title>
28 </head>
29 <body>
30 </body>
31 </html>
```



세션

- 사용자가 로그인한 후 세션에 사용자 정보를 저장하여 이후 요청에서 사용.
- 장바구니 기능을 구현할 때, 사용자가 선택한 상품들을 세션에 저장.

로그인 여부에 따라 화면 다른 흐름

로그인 화면입니다.

id : pw : apple님이 로그인 중입니다.~!

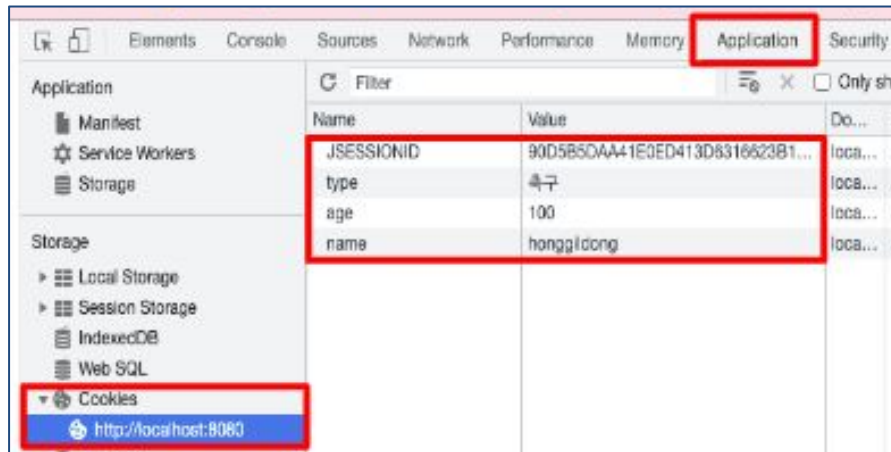
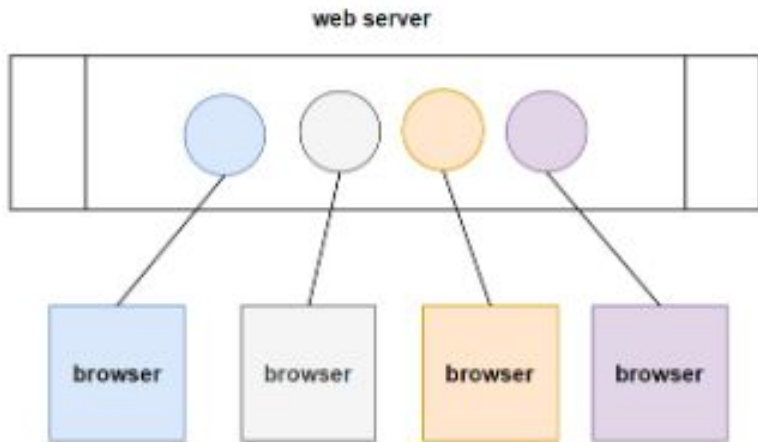
apple님! 로그인 처리 성공!



로그인페이지로



- 세션은 클라이언트 단위로 생성
- 세션은 생성시 세션아이다가 부여됨
- 톰킷은 **JSESSIONID**라는 이름의 쿠키로 브라우저 컴퓨터에 저장
- 웹 클라이언트가 가지고 있는 세션아이디를 사용하여 세션을 톰킷이 관리하게 됨.



- 서버에 저장. 세션 ID는 클라이언트의 브라우저에 쿠키로 저장
- 브라우저를 닫거나 세션 타임아웃이 발생할 때까지 유지
- 서버 설정에 따라 타임아웃 시간은 달라질 수 있음.
- 서버의 메모리나 저장 공간에 따라 제한. 클라이언트 측의 크기 제한은 없음.
- 서버에 저장. 비교적 안전
- 세션 ID가 도난당하지 않도록 **HTTPS**를 사용하는 것을 권장
- 사용 예시:
 - 사용자의 로그인 상태 유지.
 - 장바구니 정보 저장.
 - 일시적인 사용자 정보 저장(예: 양식 데이터).

메서드	설명	사용 예시
<code>void setAttribute(String name, Object value)</code>	세션에 속성(name)과 그 값을 설정	<code>session.setAttribute("username", "john");</code>
<code>Object getAttribute(String name)</code>	세션에서 속성(name)의 값을 획득	<code>String username = (String) session.getAttribute("username");</code>
<code>void removeAttribute(String name)</code>	세션에서 속성(name)을 제거	<code>session.removeAttribute("username");</code>
<code>String getId()</code>	세션의 고유 ID를 반환	<code>String sessionId = session.getId();</code>
<code>long getCreationTime()</code>	세션이 생성된 시간을 반환(밀리초 단위)	<code>long creationTime = session.getCreationTime();</code>
<code>long getLastAccessedTime()</code>	세션에 마지막으로 접근한 시간을 반환(밀리초 단위)	<code>long lastAccessedTime = session.getLastAccessedTime();</code>
<code>int getMaxInactiveInterval()</code>	세션의 최대 유효 시간(초)을 반환	<code>int maxInterval = session.getMaxInactiveInterval();</code>
<code>void setMaxInactiveInterval(int interval)</code>	세션의 최대 유효 시간(초)을 설정	<code>session.setMaxInactiveInterval(1800); // 30분</code>
<code>void invalidate()</code>	세션을 무효화하고, 모든 속성을 제거	<code>session.invalidate();</code>
<code>boolean isNew()</code>	세션이 새로운 세션인지 여부를 반환	<code>boolean isNew = session.isNew();</code>

세션 설정(이름+값)

- 세션 정보(이름+값) 서버에 저장, 이름 타입은 String, 값 타입은 Object

```
// 세션 값 설정
```

```
session.setAttribute("apple", 1);
```

Object으로
자동형변환

- 세션 정보(이름+값) 가지고 오기

```
//세션 값 확인
```

```
String userId = (String)session.getAttribute("apple");  
out.print(userId);
```

String으로
강제형변환

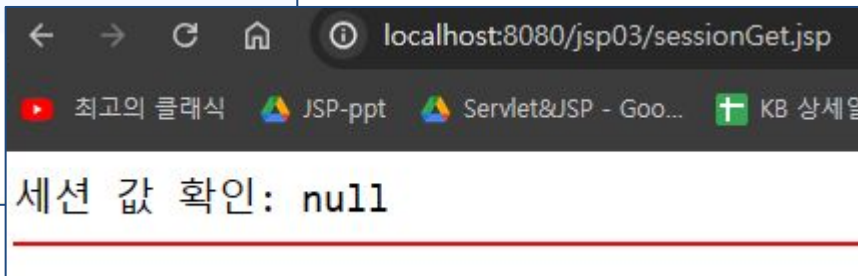
```

1  <%@ page language="java" contentType="text/html; c
2      pageEncoding="UTF-8"%>
3  <!DOCTYPE html>
4  <html>
5  <head>
6      <meta charset="UTF-8">
7      <title>Insert title here</title>
8  </head>
9  <body>
10     세션 값 확인:
11     <%
12         //세션 값 확인
13         String loginId = (String)session.getAttribute("loginId");
14         out.print(loginId);
15     %>
16     <hr color="red">
17 </body>
18 </html>
    
```

세션이름으로 설정된 것이 없으면 **null**
→ **null**로 세션이 설정되어있는지 체크

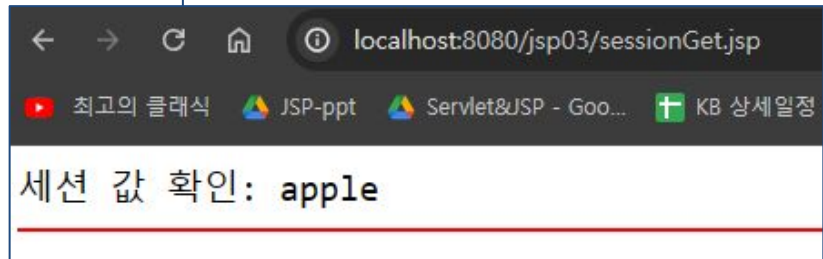
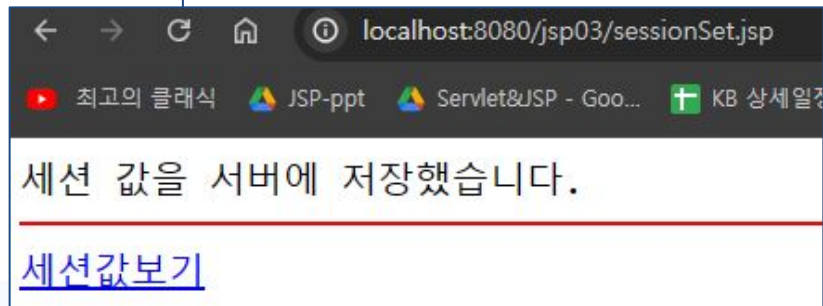
```

if (session.getAttribute("userId") == null) {
    out.print("로그인 성공"); }
else {
    out.print("로그인 실패"); }
    
```




```

1  <%@ page language="java" contentType="text/html" %>
2  <!DOCTYPE html>
3  <html>
4  <head>
5      <meta charset="UTF-8">
6      <title>Insert title here</title>
7  </head>
8  <body>
9      <%
10         // 세션 값 설정
11         session.setAttribute("loginId", "apple");
12     %>
13     세션 값을 서버에 저장했습니다.
14     <hr color="red">
15     <a href="sessionGet.jsp">세션값보기</a>
16
17 </body>
18 </html>
    
```

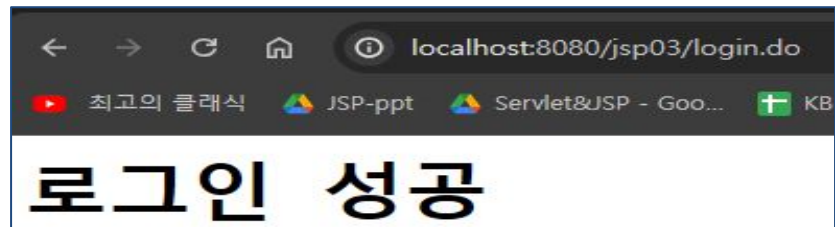
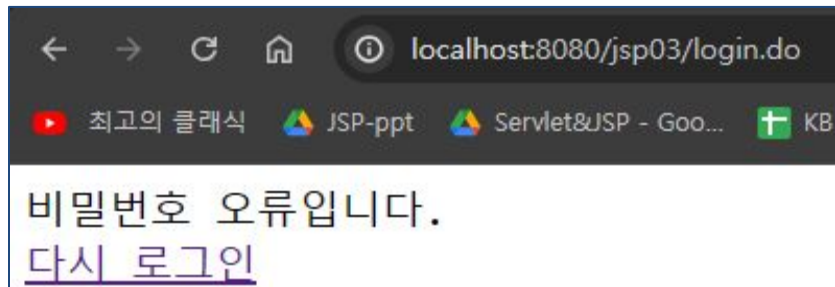


로그인 성공 후, 세션 설정

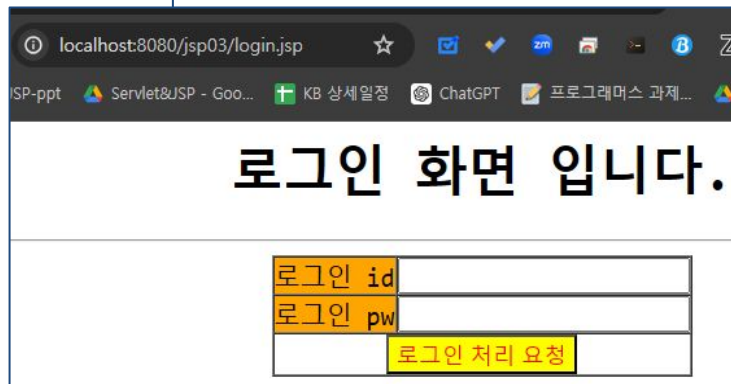
- 각 페이지를 만들 때
로그인되었을 때 화면과 로그인되지 않았을 때
화면이 다르게 만들고 싶은 경우
로그인 성공하면 세션을 설정함.
⇒ 로그인 여부를 판단하는 것은 세션으로 할 수 있음.

로그인 id

로그인 pw



```
<h1>로그인 화면 입니다.</h1>
<hr>
<form action="login.do" method="post">
  <table border="1" cellpadding="0" cellspacing="0">
    <tr>
      <td bgcolor="orange">로그인 id</td>
      <td><input type="text" name="id"/></td>
    </tr>
    <tr>
      <td bgcolor="orange">로그인 pw</td>
      <td><input type="password" name="password"/></td>
    </tr>
    <tr>
      <td colspan="2" align="center">
        <input type="submit" value="로그인 처리 요청"
          style="background: yellow; color: red"/>
      </td>
    </tr>
  </table>
</form>
```



```
private String USER_GET = "select * from users where id = ?"; 1 usage
```

```
// 회원 상세 조회
```

```
public UserVO getUser(UserVO vo) { 1 usage
    UserVO user = null;
    try {
        conn = JDBCUtil.getConnection();
        stmt = conn.prepareStatement(USER_GET);
        stmt.setString( parameterIndex: 1, vo.getId());
        rs = stmt.executeQuery();
        while(rs.next()) {
            user = new UserVO();
            user.setId(rs.getString( columnLabel: "ID"));
            user.setPassword(rs.getString( columnLabel: "PASSWORD"));
            user.setName(rs.getString( columnLabel: "NAME"));
            user.setRole(rs.getString( columnLabel: "ROLE"));
        }
    } catch (SQLException e) {
        e.printStackTrace();
    } finally {
        JDBCUtil.close(rs, stmt, conn);
    }
    return user;
}
```

UserDAO

UserVO

```
@Data 15 usag
public class UserVO {
```

```
// private 멤버변수 선언
private String id;
private String password;
private String name;
private String role;
```

LoginServlet

LoginServlet.java ×

```
@WebServlet("/login.do")
public class LoginServlet extends HttpServlet {
    private String message; no usages
```

```
public void doPost(HttpServletRequest request, HttpServletResponse
    String id = request.getParameter( name: "id");
    String password = request.getParameter( name: "password");
```

```
// 2. DB 연동 처리
```

```
UserVO vo = new UserVO();
vo.setId(id);
```

```
UserDAO dao = new UserDAO();
UserVO user = dao.getUser(vo);
System.out.println(user);
```

```
// 3. 응답 화면 구성
```

```
// 응답 메시지에 대한 인코딩 설정
```

```
response.setContentType("text/html;charset=UTF-8");
// HTTP 응답 프로토콜 message-body와 연결된 출력 스트림 획득
PrintWriter out = response.getWriter();
```

```
// 메시지 출력
```

```
if (user != null) {
    if (user.getPassword().equals(password)) {
        // 상태 정보를 쿠키에 저장하여 전송한다.
        Cookie userId = new Cookie("userId", user.getId());
        response.addCookie(userId);

        // 상태 정보를 세션에 저장한다.
        HttpSession session = request.getSession();
        session.setMaxInactiveInterval(10);
        session.setAttribute( name: "userId", user.getId());
        session.setAttribute( name: "userName", user.getName());
        session.setAttribute( name: "userRole", user.getRole());

        out.println("<h1>로그인 성공</h1>");
    } else {
        out.println("비밀번호 오류입니다.<br>");
        out.println("<a href='login.jsp'>다시 로그인</a>");
    }
} else {
    out.println("아이디 오류입니다.<br>");
    out.println("<a href='login.jsp'>다시 로그인</a>");
}
```


LoginServlet.java x

```
public class LoginServlet extends HttpServlet {
    public void doPost(HttpServletRequest request, HttpServletResponse response) {
        if (user.getPassword().equals(password)) {
            // 상태 정보를 쿠키에 저장하여 전송한다.
            // Cookie userId = new Cookie("userId", user.getId());
            // response.addCookie(userId);

            // 상태 정보를 세션에 저장한다.
            HttpSession session = request.getSession();
            session.setMaxInactiveInterval(10);
            session.setAttribute(name: "userId", user.getId());
            session.setAttribute(name: "userName", user.getName());
            session.setAttribute(name: "userRole", user.getRole());

            out.println("<h1>로그인 성공</h1>");
            out.println("<a href='logout.do'>로그아웃</a>");
        } else {
            out.println("비밀번호 오류입니다.<br>");
            out.println("<a href='login.jsp'>다시 로그인</a>");
        }
    } else {
        out.println("아이디 오류입니다.<br>");
        out.println("<a href='login.jsp'>다시 로그인</a>");
    }
}
```



```
LogoutServlet.java x
package com.example.jsp03;

import ...

@WebServlet("/logout.do")
public class LogoutServlet extends HttpServlet {
    private String message; no usages

    public void doGet(HttpServletRequest request,

        //세션 객체 획득
        HttpSession session = request.getSession();
        session.invalidate();

        // 세션 해제 후 이동할 페이지 지정
        response.sendRedirect( location: "login.jsp");
    }
}
```

localhost:8080/jsp03/login.jsp

P-ppt Servlet&JSP - Goo... KB 상세일정 ChatGPT 프로그래머스 과제...

로그인 화면 입니다.

로그인 id	<input type="text"/>
로그인 pw	<input type="password"/>
	<input type="button" value="로그인 처리 요청"/>

세션으로 장바구니 만들기

- `product.jsp`: 상품 목록을 보여주고 장바구니에 추가할 수 있는 페이지
- `CartServlet.java`: 장바구니 처리를 위한 서블릿
- `Item.java`: 상품 데이터를 저장하는 클래스

상품 목록

상품1 추가 (₩10,000)

상품2 추가 (₩20,000)

장바구니

- 장바구니가 비어있습니다.

상품 목록

상품1 추가 (₩10,000)

상품2 추가 (₩20,000)

장바구니

- product1- 10000.0
- product1- 10000.0
- product1- 10000.0
- product2- 20000.0
- product2- 20000.0
- product2- 20000.0


```
<%@ page contentType="text/html; charset=UTF-8" language="java" %>
<%@ page import="com.multi.web03.Item" %>
<%@ page import="java.util.List" %>
<html>
<head>
<title>Title</title>
</head>
<body>
<h1>상품 목록</h1>
<form action="CartServlet" method="post">
  <input type="hidden" name="itemName" value="product1">
  <input type="hidden" name="itemPrice" value="10000">
  <button type="submit">상품1 추가 (₩10,000)</button>
</form>

<form action="CartServlet" method="post">
  <input type="hidden" name="itemName" value="product2">
  <input type="hidden" name="itemPrice" value="20000">
  <button type="submit">상품2 추가 (₩20,000)</button>
</form>

<hr color="red">
<h3>장바구니</h3>
<ul>
  <%
    if(session.getAttribute("cart") != null){
      List<Item> list = (List<Item>) session.getAttribute("cart");

      for (Item item: list){
        out.print("<li>" + item.getName() + " - " + item.getPrice() + "</li>");
      }
    }else {
      out.print("<li>장바구니가 비어있습니다.</li>");
    }
  %>
</ul>
</body>
</html>
```

cart.jsp

상품 목록

상품1 추가 (₩10,000)

상품2 추가 (₩20,000)

장바구니

- 장바구니가 비어있습니다.

```
public class Item {  
    private String name;  
    private double price;  
  
    public Item(String name, double price) {  
        this.name = name;  
        this.price = price;  
    }  
  
    public String getName() {  
        return name;  
    }  
  
    public double getPrice() {  
        return price;  
    }  
}
```



Item

장바구니

- product1- 10000.0
- product1- 10000.0
- product1- 10000.0
- product2- 20000.0
- product2- 20000.0
- product2- 20000.0

CartServlet

```
@WebServlet("/CartServlet")
public class CartServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

        HttpSession session = request.getSession();
        List<Item> cart = (List<Item>) session.getAttribute("cart");

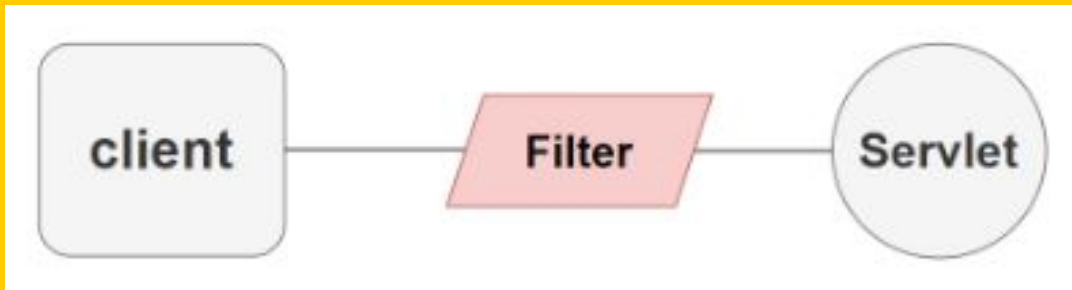
        if (cart == null) {
            cart = new ArrayList<>();
            session.setAttribute("cart", cart);
        }

        String itemName = request.getParameter("itemName");
        double itemPrice = Double.parseDouble(request.getParameter("itemPrice"));

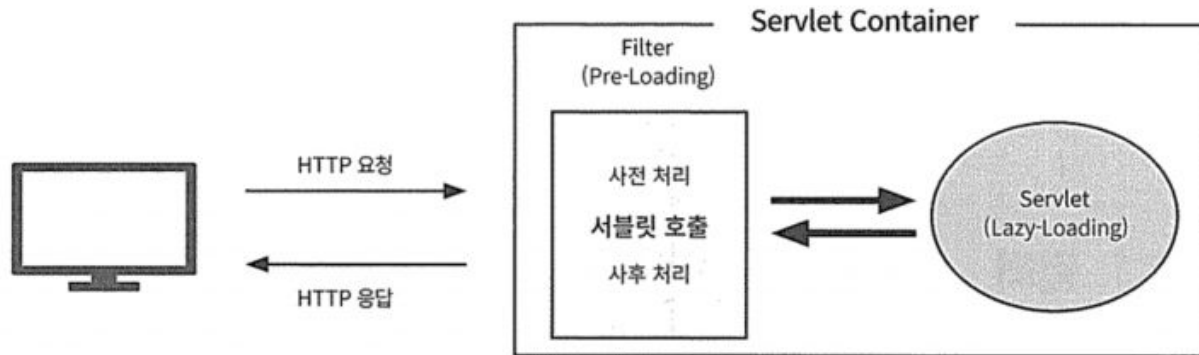
        Item item = new Item(itemName, itemPrice);
        cart.add(item);

        response.sendRedirect("cart.jsp");
    }
}
```

필터/리스너



- 웹 프로젝트에서 요청을 받기전 전처리 역할
- **post**방식의 데이터 한글 처리 역할로 사용됨.
- 필터 클래스를 만들어 **web.xml**에 등록 후 사용
- 필터 구현 : **Filter interface**
- 필터 등록 : **<filter>**
- 필터 매핑 : **<filter-mapping>**
- **@WebFilter** annotation 사용



- FilterClass implements FilterInterface
- `init(FilterConfig config)`: 필터 객체 생성시 호출, 초기화담당
- `destroy()`: 필터 객체 삭제시 호출됨, 자원해제담당
- `doFilter(ServletRequest request, ServletResponse response, FilterChain ch)`: 필터 실행시 호출됨.

```
public class FlowFilterOne implements Filter{  
    public void init(FilterConfig filterConfig ){  
        System.out.println("init() call");  
    }  
    public void doFilter(){  
        System.out.println("doFilter() call");  
    }  
    public void destroy(){  
        System.out.println("destroy() call");  
    }  
}
```

```
<filter>
    <filter-name>flow1</filter-name>
    <filter-class>com.multi.filter.FlowFilterOne</filter-class>
</filter>

<filter-mapping>
    <filter-name>flow1</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

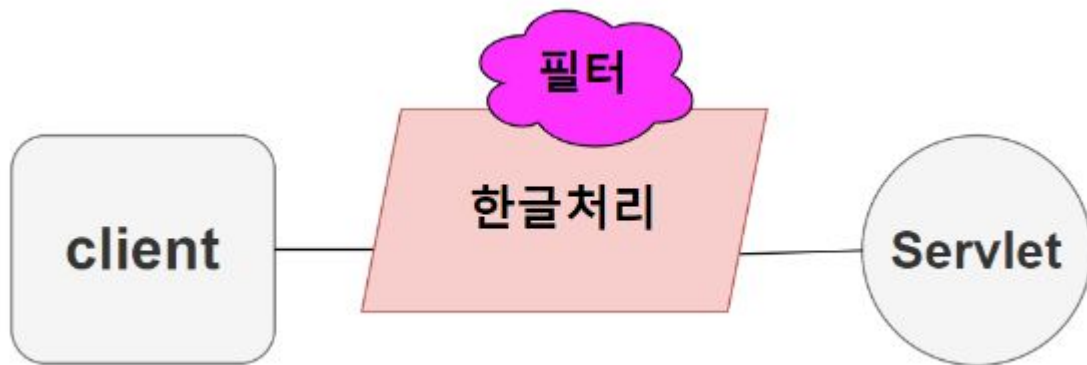
console

init() call

doFilter() call

doFilter() call

destroy() call



한글 처리 방법 1) req객체 이용

```
<form action="call1" method="post">
```

```
  name: <input name="name" value="홍길동"> <br>
```

```
  <button>한글처리요청 </button>
```

```
</form>
```



input.jsp

```
@WebServlet("call1")
```

```
public class Call1 extends HttpServlet{
```

```
    public void doPost(req, res) {
```

```
        req.setCharacterEncoding("UTF-8"); //서버에서 받을 때 한글처리
```

```
        res.setContentType("text/html; charset=UTF-8"); //서버에서 보낼 때 한글처리
```

```
        PrintWriter out = res.getWriter();
```

```
        out.print(req.getParameter("name"));
```

```
        out.close();
```

```
    }
```

```
}
```

```
web.xml x
1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app xmlns="http://xmlns.jcp.org/xml/ns/java"
3     xmlns:xsi="http://www.w3.org/2001/XMLSchema"
4     xsi:schemaLocation="http://xmlns.jcp.org
5         version="4.0">
6
7     <welcome-file-list>
8         <welcome-file>login.jsp</welcome-file>
9     </welcome-file-list>
10
11     <context-param>
12         <param-name>encoding</param-name>
13         <param-value>UTF-8</param-value>
14     </context-param>
15 </web-app>
```

```
@WebServlet("/call1")
public class Utf8Servlet extends HttpServlet {
    private String encoding; 3 usages

    public void doGet(HttpServletRequest request, HttpServletResponse response) {

        ServletContext context = getServletContext();
        encoding = context.getInitParameter(name: "encoding");
        System.out.println("---> Encoding : " + encoding);

        request.setCharacterEncoding(encoding);
        String name = request.getParameter(name: "name");

        PrintWriter out = response.getWriter();
        out.println("<html><body>");
        out.println("<h1>" + name + "</h1>");
        out.println("</body></html>");

    }
}
```

localhost:8080/jsp03/

최고의 클래식 JSP-ppt Servlet&JSP - Goo...

name : 홍길동

한글처리요청

localhost:8080/jsp03/utf8.do

최고의 클래식 JSP-ppt Servlet&JSP - Goo... KB

이름: 홍길동

input.jsp

```
<body>
  <form action="utf8.do" method="post">
    name : <input name="name" value="홍길동"> <br>
    <button>한글처리요청</button>
  </form>
</body>
</html>
```

```
public class Utf8Filter implements Filter {
```

Utf8Filter

```
private String encoding; 5 usages
```

```
@Override
```

```
public void init(FilterConfig filterConfig) throws ServletException {
```

```
    encoding = filterConfig.getInitParameter(name: "encoding");
```

```
    if (encoding == null) {
```

```
        encoding = "UTF-8"; // 기본 인코딩 설정
```

```
    }
```

```
}
```

```
@Override 1 usage
```

```
public void doFilter(ServletRequest request,  
                    ServletResponse response,  
                    FilterChain chain)
```

```
    throws IOException, ServletException {  
        request.setCharacterEncoding(encoding);  
        response.setCharacterEncoding(encoding);  
        chain.doFilter(request, response);  
    }
```

```
@Override
```

```
public void destroy() {
```

```
    // 필터 종료시 처리할 작업
```

```
}
```

```
}
```

```
//@WebFilter("/*.*do")
```

```
public class Utf8Filter implements Filter {
```

필터 등록

- @WebFilter
- web.xml

web.xml x

```
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
```

```
<filter>
```

```
<filter-name>utf8Filter</filter-name>
```

```
<filter-class>com.example.jsp03.common.Utf8Filter</filter-class>
```

```
<init-param>
```

```
<param-name>encoding</param-name>
```

```
<param-value>UTF-8</param-value>
```

```
</init-param>
```

```
</filter>
```

```
<filter-mapping>
```

```
<filter-name>utf8Filter</filter-name>
```

```
<url-pattern>*.do</url-pattern>
```

```
</filter-mapping>
```

```
<welcome-file-list>
```

```
<welcome-file>input.jsp</welcome-file>
```

```
</welcome-file-list>
```

web.xml

Utf8Servlet.java ×

```
package com.example.jsp03;
```

```
import ...
```

```
@WebServlet("/utf8.do")
```

```
public class Utf8Servlet extends HttpServlet {  
    private String encoding; no usages
```

```
    public Utf8Servlet(){ no usages  
        System.out.println("Utf8Servlet 객체 생성");  
    }
```

```
    public void doPost(HttpServletRequest request, HttpServletResponse response)  
        throws ServletException, IOException {  
        String name = request.getParameter(name: "name");  
        response.setContentType("text/html; charset=UTF-8");  
        response.getWriter().write(s: "<html><body>");  
        response.getWriter().write(s: "이름: " + name);  
        response.getWriter().write(s: "</body></html>");  
    }
```

```
}
```

스프링에서 post방식 한글 처리

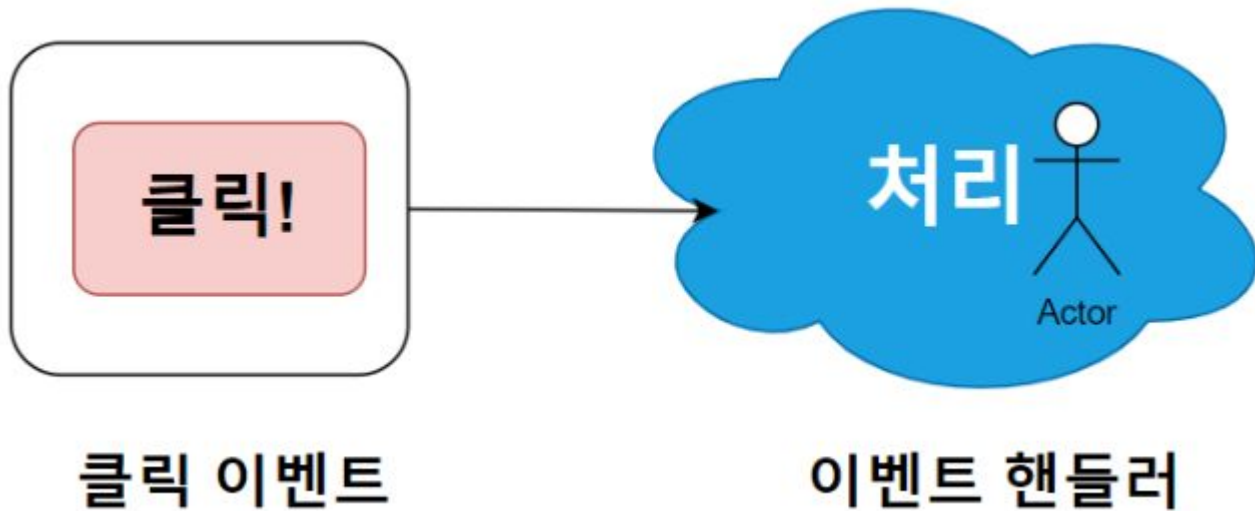
```
<filter>
  <filter-name>encodingFilter</filter-name>
  <filter-class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>
  <init-param>
    <param-name>encoding</param-name>
    <param-value>UTF-8</param-value>
  </init-param>
  <init-param>
    <param-name>forceEncoding</param-name>
    <param-value>true</param-value>
  </init-param>
</filter>

<filter-mapping>
  <filter-name>encodingFilter</filter-name>
  <url-pattern>/*</url-pattern>
</filter-mapping>
```

스프링 프레임워크에서 post방식
한글처리 설정
→ 스프링 프레임워크내에 한글 처리
라이브러리 포함되어 있음.
→ 필터로 web.xml에 등록 후 사용

리스너 (Listener)

- 이벤트 발생을 기다렸다가 자동 호출되게 하는 클래스
- 이벤트핸들러 : 지정한 이벤트가 발생하면 처리하는 클래스



리스너 종류



리스너	사용 시점	예시
ServletContextListener	웹 애플리케이션 시작 시점, 종료 시점	애플리케이션 시작 시 데이터베이스 연결 초기화, 종료 시 리소스 정리
HttpSessionListener	새로운 세션 생성 시점, 기존 세션 소멸 시점	새로운 세션 생성 시 사용자 초기 설정, 세션 종료 시 데이터 정리
ServletRequestListener	클라이언트 요청 도달 시점, 요청 처리 완료 시점	요청 초기화 시 로깅, 요청 종료 시 리소스 정리

ServletContextListener

```
public class AppContextListener implements
    ServletContextListener {
    public void contextInitialized(ServletContextEvent sce) {
        // 애플리케이션 초기화 로직
        System.out.println("Web application started");
    }

    public void contextDestroyed(ServletContextEvent sce) {
        // 애플리케이션 종료 로직
        System.out.println("Web application stopped");
    }
}
```

HttpSessionListener

```
public class UserSessionListener implements HttpSessionListener {
    public void sessionCreated(HttpSessionEvent se) {
        // 세션 생성 시 로직
        System.out.println("Session created: " + se.getSession().getId());
    }

    public void sessionDestroyed(HttpSessionEvent se) {
        // 세션 종료 시 로직
        System.out.println("Session destroyed: " + se.getSession().getId());
    }
}
```

```
public class RequestListener implements ServletRequestListener {
    public void requestInitialized(ServletRequestEvent sre) {
        // 요청 초기화 시 로직
        System.out.println("Request initialized: " + sre.getServletRequest().getRemoteAddr());
    }

    public void requestDestroyed(ServletRequestEvent sre) {
        // 요청 종료 시 로직
        System.out.println("Request destroyed: " + sre.getServletRequest().getRemoteAddr());
    }
}
```

ServletRequestListener




@WebListener

@WebListener

```
public class RequestListener implements ServletRequestListener {  
    public void requestInitialized(ServletRequestEvent sre) { 1 usage
```

@WebListener

```
public class UserSessionListener implements HttpSessionListener {  
    public void sessionCreated(HttpSessionEvent se) { 1 usage
```



web.xml

```
<listener>
```

```
    <listener-class>com.example.jsp03.common.AppContextListener</listener-class>
```

```
</listener>
```

리스너 실행

localhost:8080/jsp03/

최고의 클래식 JSP-ppt Servlet&JSP - Goo...

name : 홍길동

한글처리요청

localhost:8080/jsp03/login...

로그인 화면 입니다.

로그인 id	spring
로그인 pw	****
	로그인 처리 요청

최고의 클래식

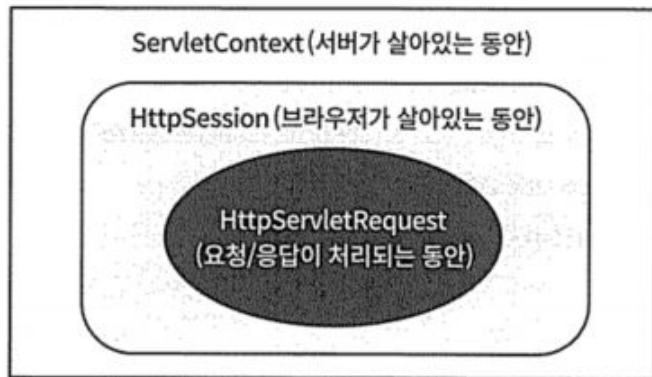
이름: 홍길동

로그인 성공

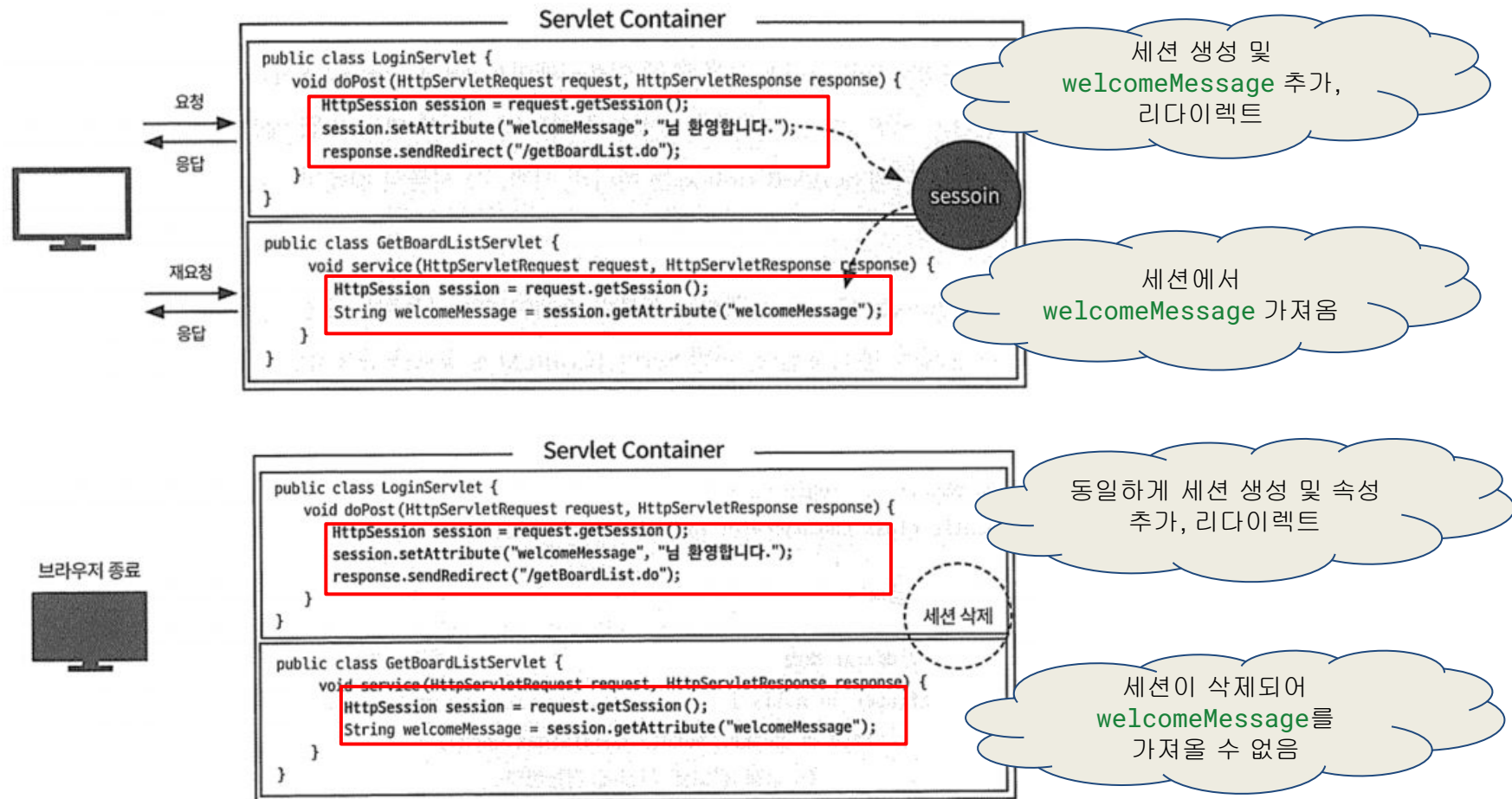
로그아웃

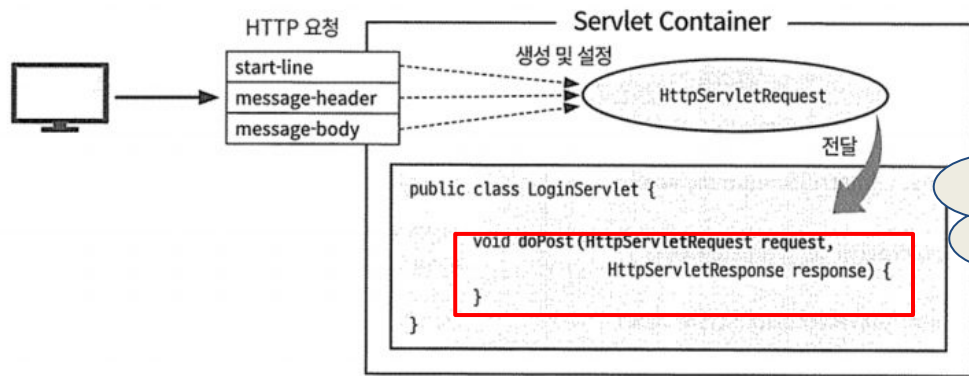
web application started
14-Jul-2024 22:41:35.258 WARNING [localhost-startStop-1]
14-Jul-2024 22:41:35.298 INFO [localhost-startStop-1] o
14-Jul-2024 22:41:35.305 INFO [main] org.apache.coyote..
14-Jul-2024 22:41:35.553 INFO [main] org.apache.catalin
<http://localhost:8080/jsp03>
Request initialized: 0:0:0:0:0:0:1
Session created: 2EB9FCC7EE76EBF9B19077656A45165A
Request destroyed: 0:0:0:0:0:0:1
Request initialized: 0:0:0:0:0:0:1
Utf8Servlet 객체 생성
Request destroyed: 0:0:0:0:0:0:1
Request initialized: 0:0:0:0:0:0:1
Request destroyed: 0:0:0:0:0:0:1
Request initialized: 0:0:0:0:0:0:1
Request destroyed: 0:0:0:0:0:0:1
Request initialized: 0:0:0:0:0:0:1
Request destroyed: 0:0:0:0:0:0:1
Request initialized: 0:0:0:0:0:0:1
----- called -----
1. 드라이버 로딩 성공
Request initialized: 0:0:0:0:0:0:1
----- called -----
1. 드라이버 로딩 성공
2. 커넥션 연결
2. 커넥션 연결
UserVO(id=spring, password=1234, name=한글, role=USER)
UserVO(id=spring, password=1234, name=한글, role=USER)
Request destroyed: 0:0:0:0:0:0:1
Request destroyed: 0:0:0:0:0:0:1
Request initialized: 0:0:0:0:0:0:1
Session destroyed: 2EB9FCC7EE76EBF9B19077656A45165A
Request destroyed: 0:0:0:0:0:0:1
Request initialized: 0:0:0:0:0:0:1
Session created: A1E1611786ADE68F8CE2F773D1E4B141
Request destroyed: 0:0:0:0:0:0:1
Request initialized: 0:0:0:0:0:0:1
Request destroyed: 0:0:0:0:0:0:1
Request initialized: 0:0:0:0:0:0:1
Request destroyed: 0:0:0:0:0:0:1
Request initialized: 0:0:0:0:0:0:1
Request destroyed: 0:0:0:0:0:0:1
Request initialized: 0:0:0:0:0:0:1
Request destroyed: 0:0:0:0:0:0:1
14-Jul-2024 22:44:00.170 INFO [Thread-1] org.apache.coy
14-Jul-2024 22:44:00.213 INFO [Thread-1] org.apache.cat
web application stopped

서블릿 객체와 정보 공유

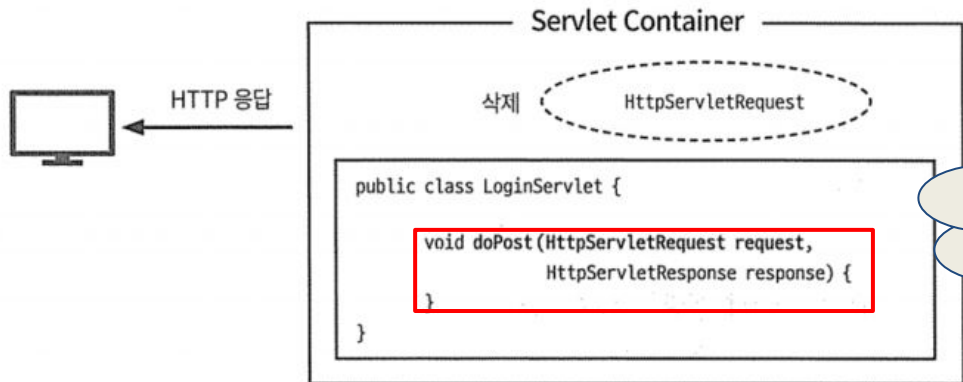


객체	설명	주요 용도
ServletContext	애플리케이션 시작/종료 시 생성	애플리케이션 전역 설정, 자원 공유, 이벤트 처리
HttpServletRequest	클라이언트의 요청 시 생성	요청 정보 읽기, 요청 범위 데이터 저장, 요청 포워딩/인클루드
HttpSession	사용자 세션 동안 유지	사용자 세션 정보 저장, 세션 설정 및 관리

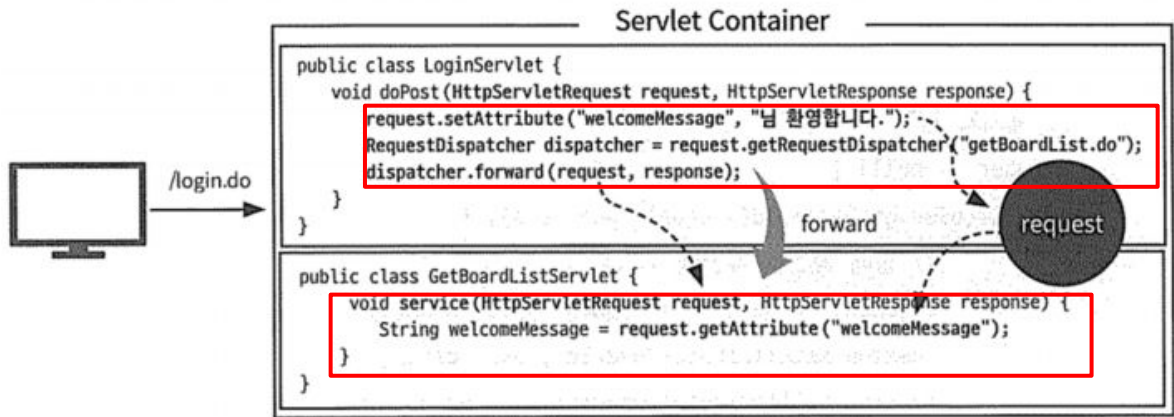




HttpServletRequest 객체 생성
및 설정, 요청 정보 설정 후
Servlet에 전달
→ Servlet의 doPost
메서드에서 요청을 처리

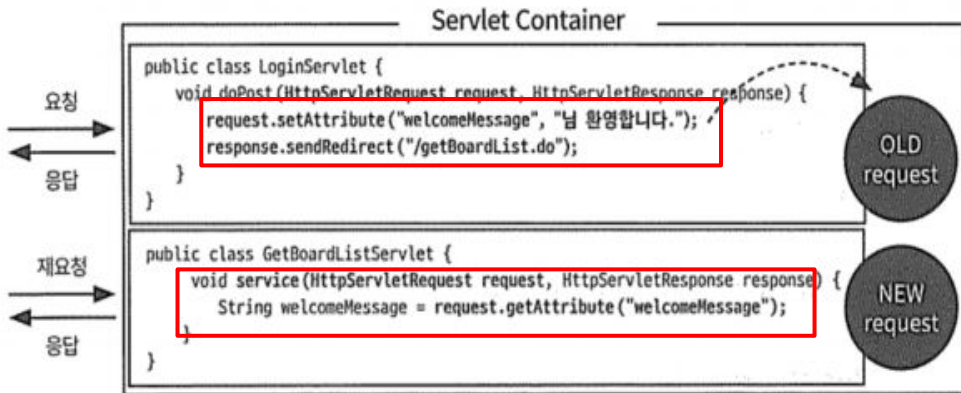


처리된 응답이
HttpServletResponse 객체를
통해 클라이언트로 반환,
HttpServletRequest 객체 삭제



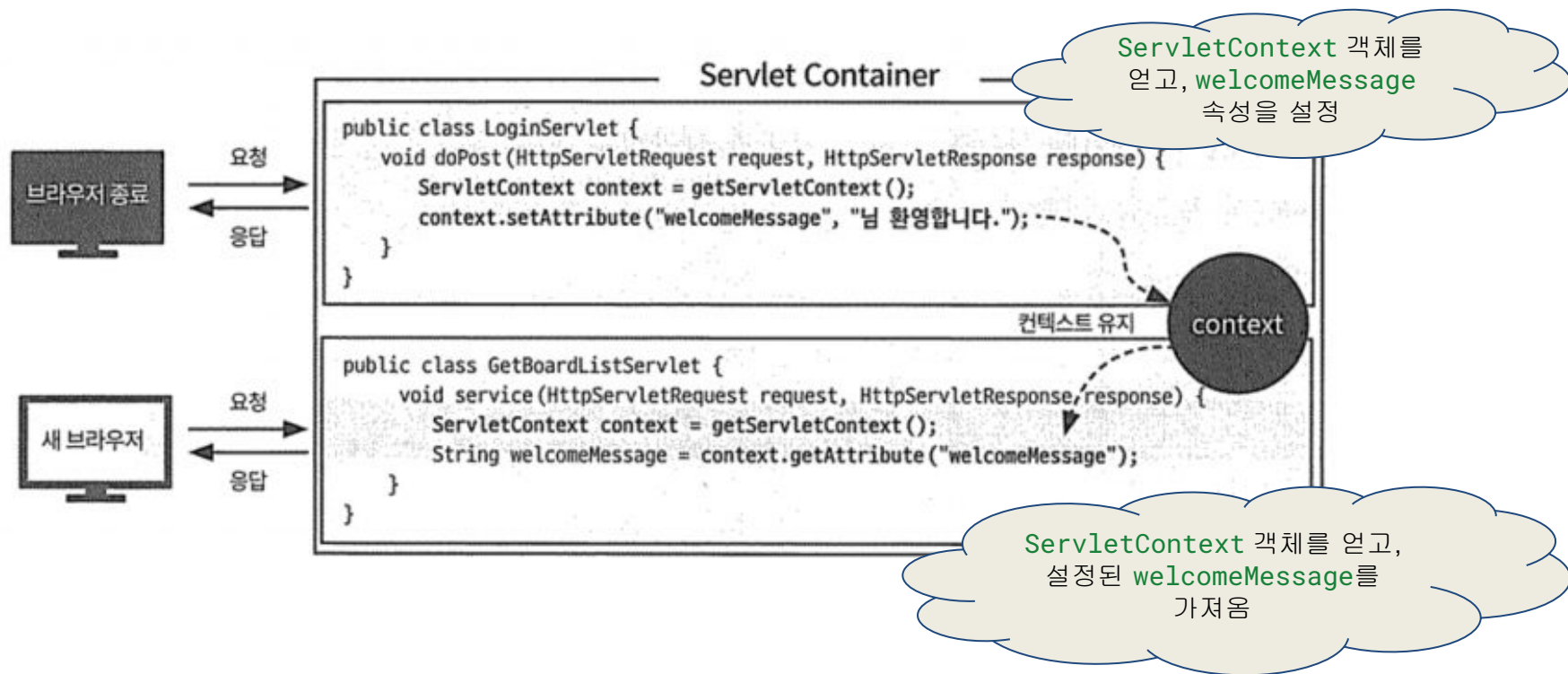
forward

- 동일한 요청 객체 사용
- **RequestDispatcher**로 포워드
- 클라이언트 요청 횟수 : 한 번
- 브라우저 URL 변화 : 변하지 않음



redirect

- 새로운 요청 객체 생성
- **sendRedirect**로 리다이렉트
- 새로운 **request** 객체 사용, 이전 속성 접근 불가
- 클라이언트 요청 횟수 : 두 번
- 브라우저 URL 변화 : 변함



- 상태정보의 필요성
 - 세션
 - 쿠키
- 필터
 - 생성 방법
 - 등록 방법
 - 필터 적용 패턴 적용
- 리스너
 - 생성 방법
 - 등록 방법
 - 리스너 3가지 이벤트