

PART3 -세션/쿠키/필터/리스너

학습 목표

- 서버와 브라우저간의 세션 트래킹
- HttpSession을 이용하는 로그인 처리
- Cookie를 생성하고 전송하는 방법
- 필터를 이용해서 공통의 코드를 적용하는 방법
- 리스너를 이용해서 특정한 이벤트에 반응하는 방법

무상태와 세션트래킹

- 웹은 기본적으로 상태를 유지하지 않는다(stateless)
- 지난번 서버에 방문한 사용자를 어떻게 추적할 것인가? – 세션트래킹
 - 서버에서 정보를 보관하는 방식 – 세션(Session)
 - 브라우저에서 정보를 보관하는 방식 – 쿠키(Cookie)

쿠키(Cookie)

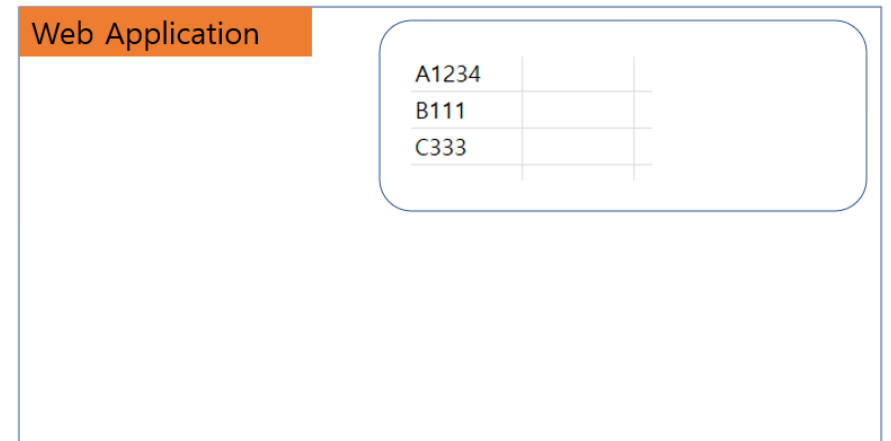
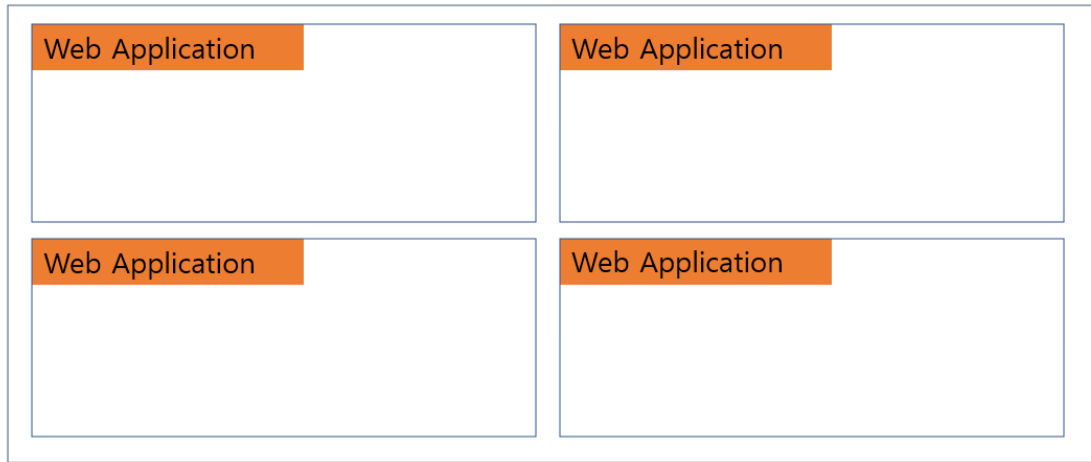
- 브라우저와 서버사이에서 주고 받는 작은 문자열
- 특정 서버에서 발행한 쿠키는 지정된 경로에 다시 접근할 때 반드시 전송하도록 규정
- Http 응답 헤더중 'Set-Cookie'
 - 브라우저에서 쿠키를 보관하라고 지시
- 쿠키의 만료기간에 따라서 보관
 - 만료 기간이 없는 경우 - 메모리상에서만 보관
 - 만료 기간이 있는 경우 - 파일 형태로 보관
- 쿠키는 기본적으로 '이름'과 '값'으로 구성
 - 추가적으로 만료기한, 경로 등 설정
 - 한글을 그대로 값으로 사용할 수 없음(URL인코딩필요)

쿠키를 생성하는 방법

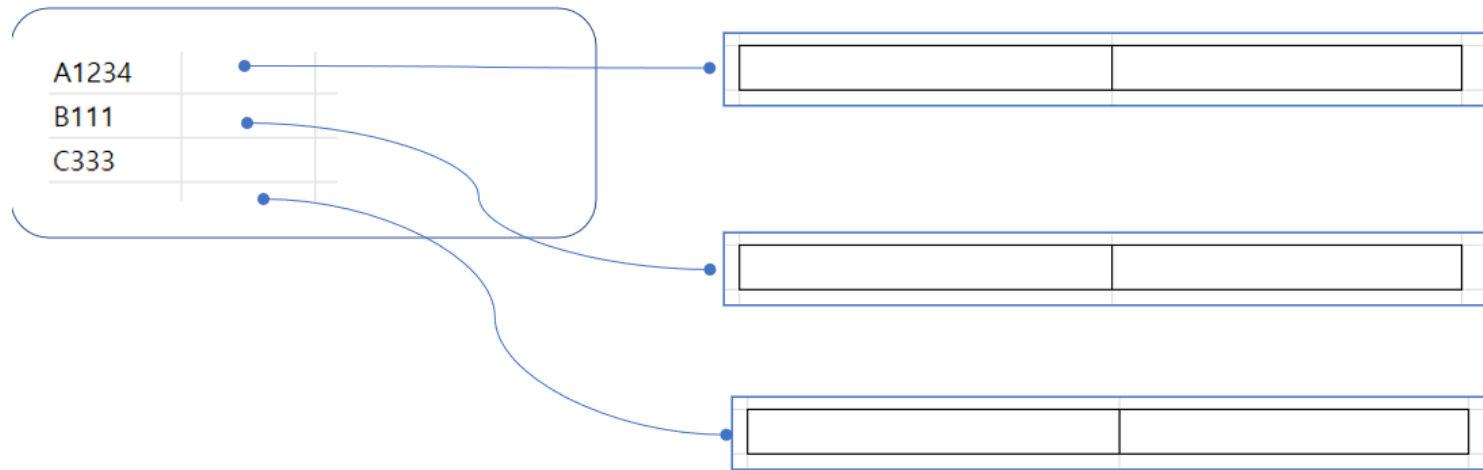
- 자동으로 발행되는 쿠키 – 서버에서 지난번 방문 브라우저를 인식하기 위해서 생성되는 쿠키로 흔히 '세션 쿠키'라고 함
 - Tomcat에서는 'JSESSIONID'라는 이름으로 발행
- 개발자가 작성하는 쿠키 – 직접 응답(response)에 포함시켜야 함
 - `response.addCookie()`

세션과 세션 유지

- WAS내 웹 프로젝트가 실행되면 WAS내 독립적인 메모리 공간이 만들어진다. – ServletContext 혹은 Application(JSP)
- Web Application 공간에는 세션 컨텍스트(Session Context)라는 별도의 공간을 이용해서 JSESSIONID와 같은 쿠키를 관리하는데 이를 '세션 저장소'라고 한다.

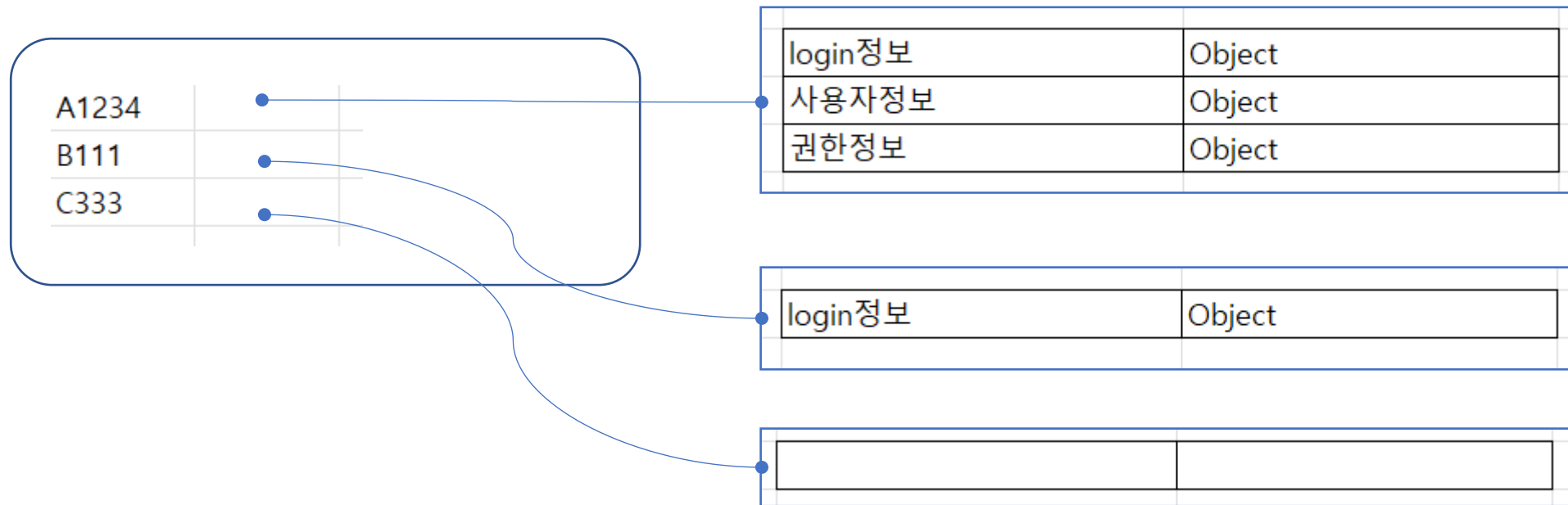


- 세션 저장소는 기본적으로 키(key)와 값(value)의 구조로 이루어지며 맵(map) 구조로 생성
- JSESSIONID가 키가 되고 이에 해당하는 별도의 저장구조가 생성된다.
- Servlet에서는 HttpSession API를 이용해서 이를 접근할 수 있다.



로그인 여부의 판단

- 현재 요청에서 전송한 JSESSIONID값을 가지고 세션 저장소내 공간을 조회해서 특정한 이름과 값이 있는 데이터가 있는지를 확인
- 존재하는 경우에는 로그인한 사용자로 판단



HttpServletRequest의 getSession()

- 요청시 모든 정보는 HttpServletRequest를 통해서 접근하고 getSession()을 이용해서 HttpSession 객체에 접근
- getSession()호출시 JSESSIONID가 없는 경우에는 새로운 공간을 생성하고 이에 대한 참조를 HttpSession으로 반환
 - isNew()를 이용해서 새로 생성된 여부를 확인할 수 있음
- JSESSIONID가 있는 경우에는 해당 공간에 대한 접근이 가능하도록 HttpSession으로 반환

세션을 이용한 로그인 체크

- 사용자가 로그인을 하면 현재 요청의 HttpSession을 찾아서 특정한 키(key)와 값(value)을 저장
 - HttpSession의 setAttribute(키,값)
 - 키(key)는 문자열, 값(value)은 Object타입
- 다음 방문시 가지고 온 세션 쿠키의 값을 이용해서 HttpSession 내에 로그인시에 저장한 정보가 있는지를 확인
 - 존재하는 경우 로그인한 사용자로 인정

로그인 체크

- org.zerock.w2
 - controller
 - TodoListController
 - TodoModifyController
 - TodoReadController
 - TodoRegisterController**
 - TodoRemoveController

```
@Override
protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {

    log.info("/todo/register GET .....");

    HttpSession session = req.getSession();

    if(session.isNew()) { //기존에 JSESSIONID가 없는 새로운 사용자
        log.info("JSESSIONID 쿠키가 새로 만들어진 사용자");
        resp.sendRedirect("/login");
        return;
    }

    //JSESSIONID는 있지만 해당 세션 컨텍스트에 loginInfo라는 이름으로 저장된
    //객체가 없는 경우
    if(session.getAttribute("loginInfo") == null) {
        log.info("로그인한 정보가 없는 사용자.");
        resp.sendRedirect("/login");
        return;
    }

    //정상적인 경우라면 입력 화면으로
    req.getRequestDispatcher("/WEB-INF/todo/register.jsp").forward(req, resp);
}
```

```
01:11:03.595 [http-nio-8080-exec-4] INFO org.zerock.w2.controller.TODORegisterController - /todo/register GET .....
01:11:03.595 [http-nio-8080-exec-4] INFO org.zerock.w2.controller.TODORegisterController - JSESSIONID 쿠키가 새로 만들어진 사용자
```

HTTP 상태 404 - 찾을 수 없음

타입 상태 보고

메시지 요청된 리소스 [/login]은(는) 가용하지 않습니다.

설명 Origin 서버가 대상 리소스를 위한 현재의 representation을 찾지 못했거나, 그것이 존재하는지를 밝히려 하지 않습니다.

Apache Tomcat/9.0.59

프로젝트의 루트 폴더를 식별하여 Visual Studio Code 소스 파일을 열고 변경 내용을 동기화합니다. **루트 폴더 설정** 다시 표시 안 함

시작 요소 콘솔 소스 네트워크 성능 메모리 **응용 프로그램** x 보안 Lighthouse

응용 프로그램

필터

이름 값

JSESSIONID 105808A6ECD50706C8E9AD9C87F045F

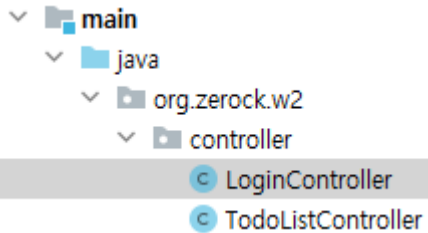
저장소

매니페스트 Service Workers 저장소

문제가 있는 쿠키만 표시

로그인

로그인 처리



```
package org.zerock.w2.controller;

import lombok.extern.log4j.Log4j2;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

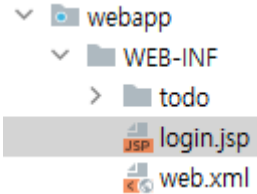
@WebServlet("/login")
@Log4j2
public class LoginController extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {

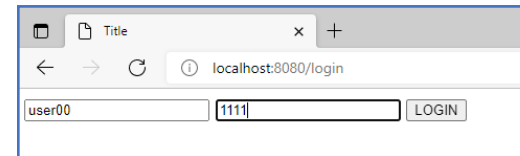
        log.info("login get.....");

        req.getRequestDispatcher("/WEB-INF/login.jsp").forward(req, resp);

    }
}
```



```
<%@ page contentType="text/html; charset=UTF-8" language="java" %>
<html>
<head>
    <title>Title</title>
</head>
<body>
    <form action="/login" method="post">
        <input type="text" name="mid">
        <input type="text" name="mpw">
        <button type="submit">LOGIN</button>
    </form>
</body>
</html>
```



controller

LoginController

```
@Override
protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException,
IOException {

    log.info("login post.....");

    String mid = req.getParameter("mid");
    String mpw = req.getParameter("mpw");

    String str = mid+mpw;

    HttpSession session = req.getSession();

    session.setAttribute("loginInfo", str);

    resp.sendRedirect("/todo/list");

}
```

localhost:8080/login

user00 1111 LOGIN

Todo List

localhost:8080/todo/list

동기화하지 않음

1 Sample Title... 2021-12-31 DONE

2 Test... 2022-12-31 DONE

3 Not Yet... 2022-12-31 NOT YET

4 Test... 2022-12-31 DONE

5 Test... 2022-12-31 DONE

7 JDBC Test Title 2022-03-07 NOT YET

8 JDBC Test Title 2022-03-07 NOT YET

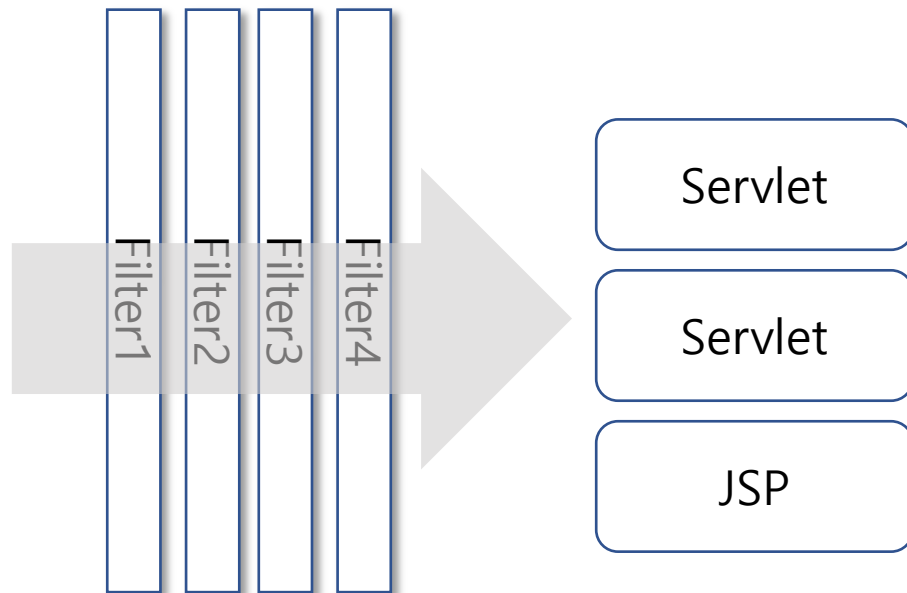
9 JDBC Test Title 2022-03-07 NOT YET

10 JDBC Test Title 2022-03-07 NOT YET

11 Modify.....Todo Test 2022-04-24 DONE

필터를 이용한 로그인 체크

- 로그인 처리가 필요한 모든 컨트롤러에 동일한 코드를 적용하는 대신 필터를 이용
- 특정한 경로로 접근할때 지정된 필터를 통해 제어
- 여러 개의 필터를 등록해서 사용할 수 있음



▼ org.zerock.w2

> controller

> dao

> domain

> dto

▼ filter

● LoginCheckFilter

> service

> util

@WebFilter 어노테이션을 이용해서 필터 추가
urlPatterns 속성값으로 특정 URL 접근시 필터 동작 설정

doFilter()의 마지막 FilterChain을 이용해서 다음 필터나 목적지로 이동하고자 하는 경우 chain.doFilter()

```
package org.zerock.w2.filter;

import lombok.extern.log4j.Log4j2;

import javax.servlet.*;
import javax.servlet.annotation.WebFilter;
import java.io.IOException;

@WebFilter(urlPatterns = {"/todo/*"})
@Log4j2
public class LoginCheckFilter implements Filter {
    @Override
    public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) throws IOException, ServletException {

        log.info("Login check filter....");

        chain.doFilter(request, response);
    }
}
```

로그인 처리 필터

```
package org.zerock.w2.filter;

import lombok.extern.log4j.Log4j2;

import javax.servlet.*;
import javax.servlet.annotation.WebFilter;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import java.io.IOException;

@WebFilter(urlPatterns = {"/todo/*"})
@Log4j2
public class LoginCheckFilter implements Filter {

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

        log.info("Login check filter....");

        HttpServletRequest req = (HttpServletRequest)request;
        HttpServletResponse resp = (HttpServletResponse)response;

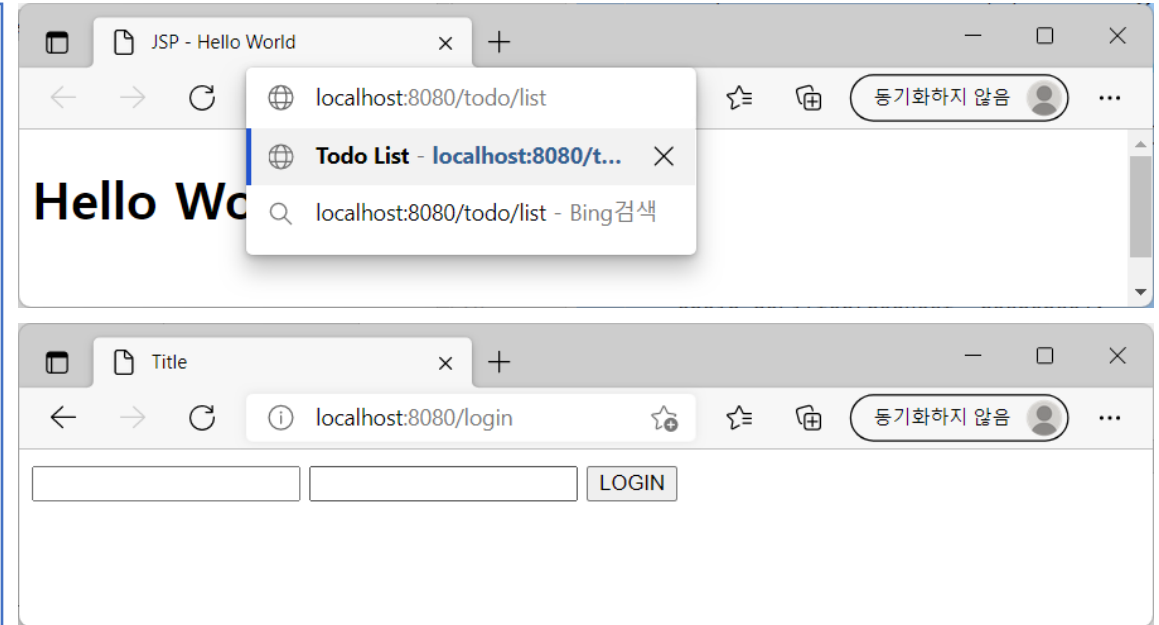
        HttpSession session = req.getSession();

        if(session.getAttribute("loginInfo") == null){

            resp.sendRedirect("/login");

            return;
        }

        chain.doFilter(request, response);
    }
}
```



```
[http-nio-8080-exec-7] INFO org.zerock.w2.filter.LoginCheckFilter - Login check filter....
[http-nio-8080-exec-8] INFO org.zerock.w2.controller.LoginController - login get.....
```


UTF-8필터

- POST방식의 한글 처리시에 한글이 깨지는 문제 해결

filter

LoginCheckFilter

UTF8Filter

```
package org.zerock.w2.filter;

import lombok.extern.log4j.Log4j2;

import javax.servlet.*;
import javax.servlet.annotation.WebFilter;
import javax.servlet.http.HttpServletRequest;
import java.io.IOException;

@WebFilter(urlPatterns = {"//*"})
@Log4j2
public class UTF8Filter implements Filter {

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

        log.info("UTF8 filter...");

        HttpServletRequest req = (HttpServletRequest) request;

        req.setCharacterEncoding("UTF-8");

        chain.doFilter(request, response);
    }
}
```

세션을 이용하는 로그아웃 처리

- HttpSession의 invalidate()를 이용해서 무효화 처리

```
package org.zerock.w2.controller;

import lombok.extern.log4j.Log4j2;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
import java.io.IOException;

@WebServlet("/logout")
@Log4j2
public class LogoutController extends HttpServlet {

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {

        log.info("log out.....");

        HttpSession session = req.getSession();

        session.removeAttribute("loginInfo");
        session.invalidate();

        resp.sendRedirect("/");

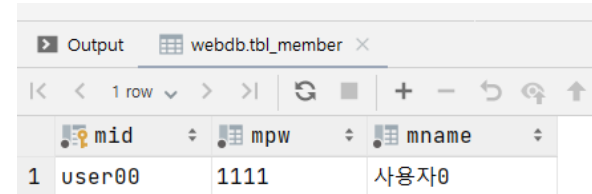
    }
}
```

데이터베이스내 회원 정보와 연동-DB

```
create table tbl_member (  
    mid varchar(50) primary key,  
    mpw varchar(50) not null,  
    mname varchar(100) not null  
);
```

```
insert into tbl_member (mid, mpw, mname) values ('user00','1111','사용자0');  
insert into tbl_member (mid, mpw, mname) values ('user01','1111','사용자1');  
insert into tbl_member (mid, mpw, mname) values ('user02','1111','사용자2');
```

```
select * from tbl_member where mid='user00' and mpw = '1111';
```



	mid	mpw	mname
1	user00	1111	사용자0

MemberVO, MemberDAO

▼ org.zerock.w2
> controller
> dao
▼ domain

MemberVO

TodoVO

```
package org.zerock.w2.domain;

import lombok.*;

@Getter
@ToString
@Builder
@AllArgsConstructor
@NoArgsConstructor
public class MemberVO {

    private String mid;
    private String mpw;
    private String mname;
}
```

▼ org.zerock.w2
> controller
▼ dao
ConnectionUtil
MemberDAO
TodoDAO

```
package org.zerock.w2.dao;

import lombok.Cleanup;
import org.zerock.w2.domain.MemberVO;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;

public class MemberDAO {

    public MemberVO getWithPassword(String mid, String mpw) throws Exception {

        String query = "select mid, mpw, mname from tbl_member where mid =? and mpw = ?";

        MemberVO memberVO = null;

        @Cleanup Connection connection = ConnectionUtil.INSTANCE.getConnection();
        @Cleanup PreparedStatement preparedStatement =
            connection.prepareStatement(query);
        preparedStatement.setString(1, mid);
        preparedStatement.setString(2, mpw);

        @Cleanup ResultSet resultSet = preparedStatement.executeQuery();

        resultSet.next();

        memberVO = MemberVO.builder()
            .mid(resultSet.getString(1))
            .mpw(resultSet.getString(2))
            .mname(resultSet.getString(3))
            .build();

        return memberVO;
    }
}
```

MemberDTO, MemberService

▼ org.zerock.w2
 > controller
 > dao
 > domain
 ▼ dto

MemberDTO

TodoDTO

```
package org.zerock.w2.dto;

import lombok.AllArgsConstructor;
import lombok.Builder;
import lombok.Data;
import lombok.NoArgsConstructor;

@Data
@Builder
@AllArgsConstructor
@NoArgsConstructor
public class MemberDTO {

    private String mid;
    private String mpw;
    private String mname;
}
```

▼ org.zerock.w2
 > controller
 > dao
 > domain
 > dto
 > filter

▼ service

MemberService

TodoService

```
package org.zerock.w2.service;

import lombok.extern.log4j.Log4j2;
import org.modelmapper.ModelMapper;
import org.zerock.w2.dao.MemberDAO;
import org.zerock.w2.util.MapperUtil;

@Log4j2
public enum MemberService {

    INSTANCE;

    private MemberDAO dao;
    private ModelMapper modelMapper;

    MemberService() {

        dao = new MemberDAO();
        modelMapper = MapperUtil.INSTANCE.get();

    }

}
```

```
public MemberDTO login(String mid, String mpw) throws Exception
{

    MemberVO vo = dao.getWithPassword(mid, mpw);

    MemberDTO memberDTO = modelMapper.map(vo,
    MemberDTO.class);

    return memberDTO;
}
```

컨트롤러와 서비스 연동

▼ org.zerock.w2

▼ controller

Ⓢ LoginController

```
@WebServlet("/login")
@Log4j2
public class LoginController extends HttpServlet {

    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {

        log.info("login get.....");

        req.getRequestDispatcher("/WEB-INF/login.jsp").forward(req, resp);
    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {

        log.info("login post.....");

        String mid = req.getParameter("mid");
        String mpw = req.getParameter("mpw");

        try {
            MemberDTO memberDTO = MemberService.INSTANCE.login(mid, mpw);

            HttpSession session = req.getSession();

            session.setAttribute("loginInfo", memberDTO);

            resp.sendRedirect("/todo/list");

        } catch (Exception e) {
            resp.sendRedirect("/login?result=error");
        }
    }
}
```

사용자 정의 쿠키

- 사용자(개발자)가 서버에서 쿠키를 생성하고 이를 이용하는 방식
- `new Cookie(이름,값)` 으로 객체 생성
- `response.addCookie()`를 이용해서 응답시 쿠키 추가
- 추가 지정항목
 - Domain
 - Path
 - Expire (seconds)

쿠키와 세션의 비교

	사용자 정의 쿠키	WAS에서 발행하는 쿠키(세션 쿠키)
생성	개발자가 직접 <code>newCookie()</code> 로 생성 경로도 지정 가능	자동
전송	반드시 <code>HttpServletResponse</code> 에 <code>addCookie()</code> 를 통 해야만 전송	
유효기간	쿠키 생성시에 초단위로 지정할 수 있음	지정불가
브라우저의 보관 방식	유효기간이 없는 경우에는 메모리상에만 보관 유효기간이 있는 경우에는 파일이나 기타 방식으 로 보관	메모리상에만 보관
쿠키의 크기	4kb	4kb

쿠키를 사용하는 경우

- 세션은 서버에서 데이터를 보관하기 때문에 좀 더 안전하다는 인식
- 쿠키의 경우 매번 브라우저와 서버사이를 오고가는 방식이므로 상대적으로 중요도가 떨어지는 경우에 사용
- 대표적인 사용처
 - 오늘 본 상품(조회목록)
 - 오늘 하루 이창 열지 않기
- 모바일 시대가 되면서 쿠키의 활용 증가
 - 사용자의 입력이 불편한 문제 – 로그인 기억하기(remember-me)

쿠키와 세션을 같이 활용하기

- 로그인 처리시에 세션내에 로그인 정보가 있는지 확인하고
- 없는 경우 로그인 관련 쿠키가 있는지 조사
- 쿠키가 존재하는 경우 이를 이용해서 다시 세션에 사용자의 정보를 추가하는 방식

```
HttpSession session = req.getSession();

if(session.getAttribute("loginInfo") == null){

    //쿠키를 체크
    Cookie cookie = findCookie(req.getCookies(), "remember-me");

    if(cookie != null){

        log.info("cookie는 존재하는 상황");
        String uuid = cookie.getValue();

        try {
            MemberDTO memberDTO = MemberService.INSTANCE.getByUUID(uuid);

            log.info("쿠키의 값으로 조회한 사용자 정보: " + memberDTO );

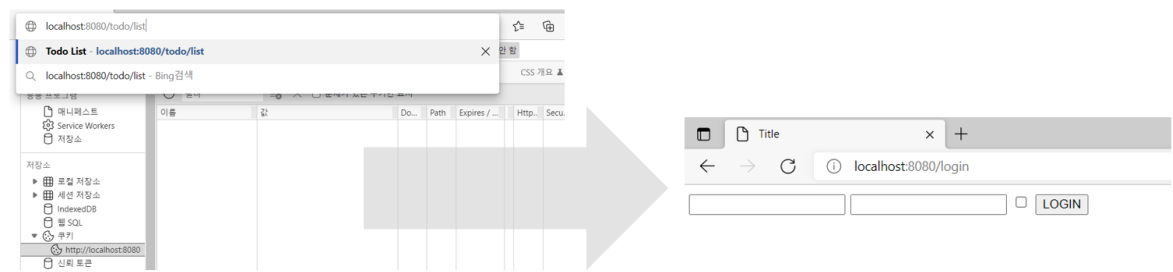
            session.setAttribute("loginInfo", memberDTO);
        } catch (Exception e) {
            e.printStackTrace();
        }

        chain.doFilter(request, response);
        return;
    }

    resp.sendRedirect("/login");

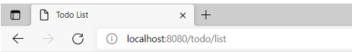
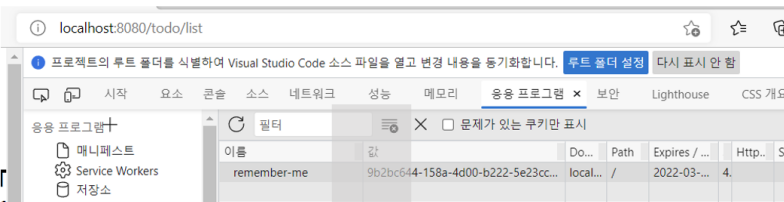
    return;
}
```

세션과 쿠키에 양쪽 모두 로그인 정보가 없는 경우



세션에만 정보가 있고 쿠키가 없는 경우 -> 로그인 성공

세션에만 정보가 없고 쿠키가 있는 경우 -> 로그인 성공



Todo List

MemberDTO(mid=user00, mpw=1111, mname=사용자0, uuid=9b2bc644-158a-4d00-b222-5e23ccf7d3bb)

사용자0

- 1 Sample Title... 2021-12-31 DONE
- 2 Test... 2022-12-31 DONE
- 3 Not Yet... 2022-12-31 NOT YET
- 4 Test... 2022-12-31 DONE
- 5 Test... 2022-12-31 DONE
- 7 JDBC Test Title 2022-03-07 NOT YET
- 8 JDBC Test Title 2022-03-07 NOT YET
- 9 JDBC Test Title 2022-03-07 NOT YET
- 10 JDBC Test Title 2022-03-07 NOT YET
- 11 Modify.....Todo Test 2022-04-24 DONE

LOGOUT

리스너(Listener)

- 특정한 정보(event)가 발생하면 이를 처리하는 객체를 '리스너(Listener)'라고 함
- Servlet API에는 다양한 종류의 리스너가 존재
 - ServletContext 관련
 - HttpSession관련
 - HttpServletRequest관련

ServletContextListener

- ServletContext가 현재 실행되는 웹 애플리케이션을 의미하므로 현재 프로젝트의 실행/종료를 감지하고 특정 작업을 수행하는 용도로 사용

▼ org.zerock.w2

- > controller
- > dao
- > domain
- > dto
- > filter
- ▼ listener
 - W2AppListener

```
package org.zerock.w2.listener;

import lombok.extern.log4j.Log4j2;

import javax.servlet.ServletContextEvent;
import javax.servlet.ServletContextListener;
import javax.servlet.annotation.WebListener;

@WebListener
@Log4j2
public class W2AppListener implements ServletContextListener {

    @Override
    public void contextInitialized(ServletContextEvent sce) {

        log.info("-----init-----");
        log.info("-----init-----");
        log.info("-----init-----");
    }

    @Override
    public void contextDestroyed(ServletContextEvent sce) {

        log.info("-----destroy-----");
        log.info("-----destroy-----");
        log.info("-----destroy-----");
    }

}
```

ServletContext의 활용

- ServletContext의 `setAttribute()`를 이용해서 현재 웹 애플리케이션내에 공유되는 객체를 지정할 수 있음
- EL에서는 별도의 처리없이 사용가능. EL의 경우 단계별 객체 검색
 - page
 - request
 - session
 - application

세션 관련 리스너

- HttpSessionListener/HttpSessionAttributeListener
- HttpSession 객체의 라이프사이클 혹은 setAttribute(), removeAttribute() 등의 상태 변경을 감지
- 활용 용도
 - 로그인/아웃 시점 기록
 - 사용자 접속 기록