



It's Your Life

with





톡캣 서버의 실행 구조

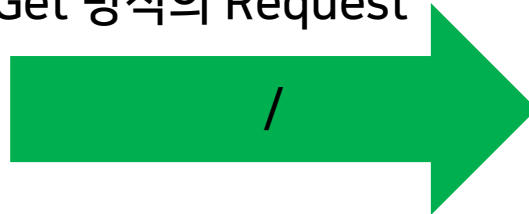


jsp 페이지 요청



Client

Get 방식의 Request



Tomcat
Server

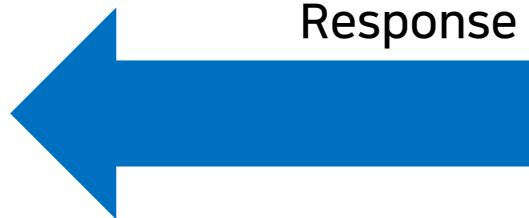


Servlets

index.jsp

```
<html>
<% String str = "Hello, World" %>
<h1><%= str %></h1>
</html>
```

서버에서 보내는
Response



```
<html>
<h1>Hello, Wolrd</h1>
</html>
```

http://localhost:8080/

Hello, World

Web Browser



서블릿 요청



Client

Get 방식의 Request

/login?username=tetz

Tomcat
Server

LoginServlet

@WebServlet(/login)

```
@Override ㄹ Tetz +1  
protected void doGet(HttpServletRequest request, HttpServletResponse response)  
    throws ServletException, IOException {  
    String username = request.getParameter("username");  
    String password = request.getParameter("password");  
}
```

<http://localhost:8080/login?username=tetz>

로그인 성공!

서버에서 보내는
Response

```
<html>  
<h1>로그인 성공!</h1>  
</html>
```

Web Browser





그란데 말입니다

jsp 페이지 요청과 서블릿 요청 중에서
어떤 것이 우선 실행이 될까요!?





톡캣 서버의 REAL 실행 구조

Tomcat Server



Web Server

정적인 처리 수행



Web Container

동적인 처리 수행



정적 페이지
Request

정적 페이지
Response

Tomcat Server



Web Server

정적인 처리 수행



Web Container

동적인 처리 수행



동적 처리 요청

동적 처리 응답

동적 페이지
Request

동적 페이지
Response



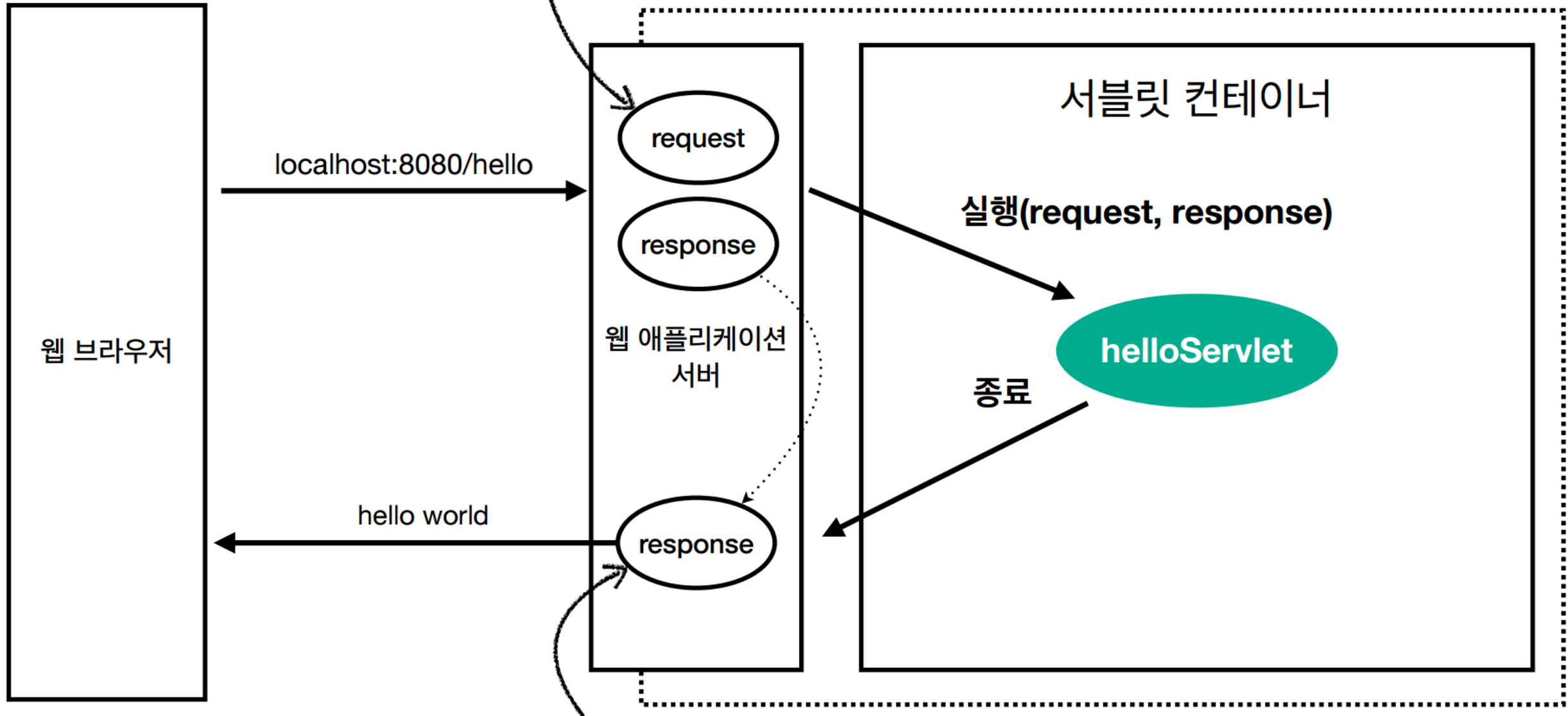
서블릿 요청의

구조

HTTP 요청 메시지를 기반으로 생성

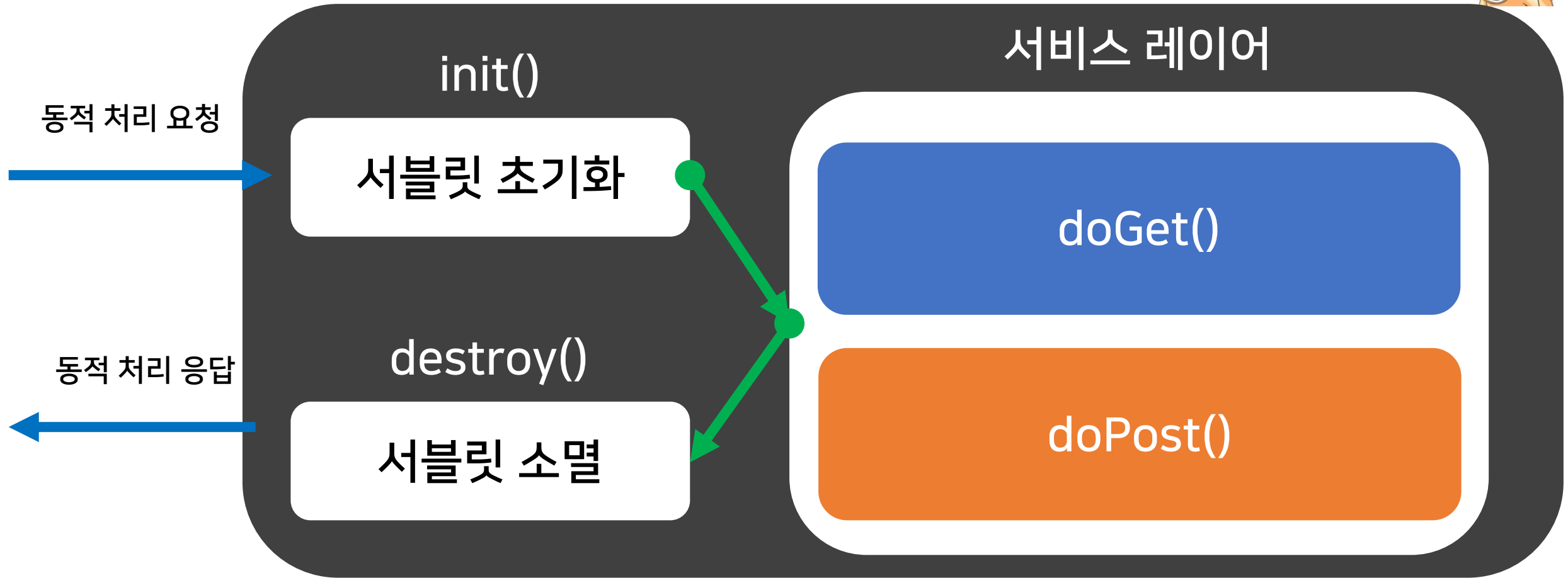


웹 애플리케이션 서버



Response 객체 정보로 HTTP 응답 생성

Servlet





눈으로
확인하기!



LoginServlet

```
@WebServlet("/login")
public class LoginServlet extends HttpServlet {
    private static final String...
    private static final String...
    private static final String...
```

Generate

- Constructor
- Getter
- toString()
- Override Methods... **Ctrl+O**
- Delegate Methods...
- Test...
- Copyright

login 서블릿에 가서
Alt + Insert(맥은 Cmd + N) 을 누른 뒤
Override Methods 선택!



↓a

⌚

↑i

⌵

×

Ⓜ ⚡ doHead(req:HttpServletRequest, resp:HttpSe

Ⓜ ⚡ doPut(req:HttpServletRequest, resp:HttpServ

Ⓜ ⚡ doDelete(req:HttpServletRequest, resp:HttpS

Ⓜ ⚡ doOptions(req:HttpServletRequest, resp:Http

Ⓜ ⚡ doTrace(req:HttpServletRequest, resp:HttpSe

Ⓜ ⚡ service(req:HttpServletRequest, resp:HttpSer

Ⓜ 🌿 service(req:ServletRequest, res:ServletRespor

▼ Ⓜ 🌿 javax.servlet.GenericServlet

Ⓜ 🌿 destroy():void

Ⓜ 🌿 getInitParameter(name:String):String

Ⓜ 🌿 getInitParameterNames():Enumeration<String

Ⓜ 🌿 getServletConfig():ServletConfig

Ⓜ 🌿 getServletContext():ServletContext

Ⓜ 🌿 getServletInfo():String

Ⓜ 🌿 init(config:ServletConfig):void

Ⓜ 🌿 init():void

Ⓜ 🌿 log(msg:String):void

Ⓜ 🌿 log(message:String, t:Throwable):void

Ⓜ 🌿 getServletName():String

☐ Copy JavaDoc

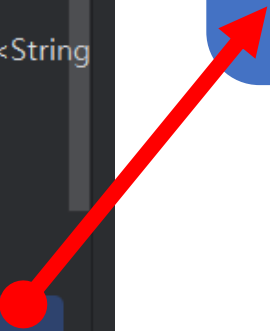
☐ Generate missed JavaDoc

☒ Insert @Override

OK

Cancel

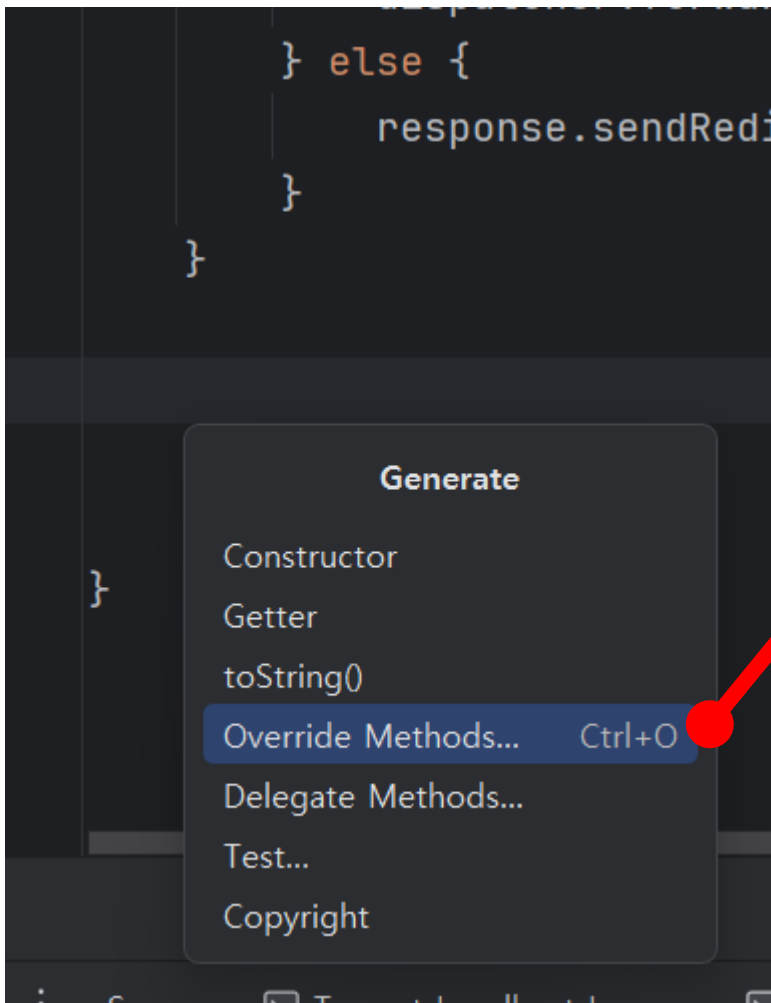
다양한 메소드 중에서
init() 선택 → OK



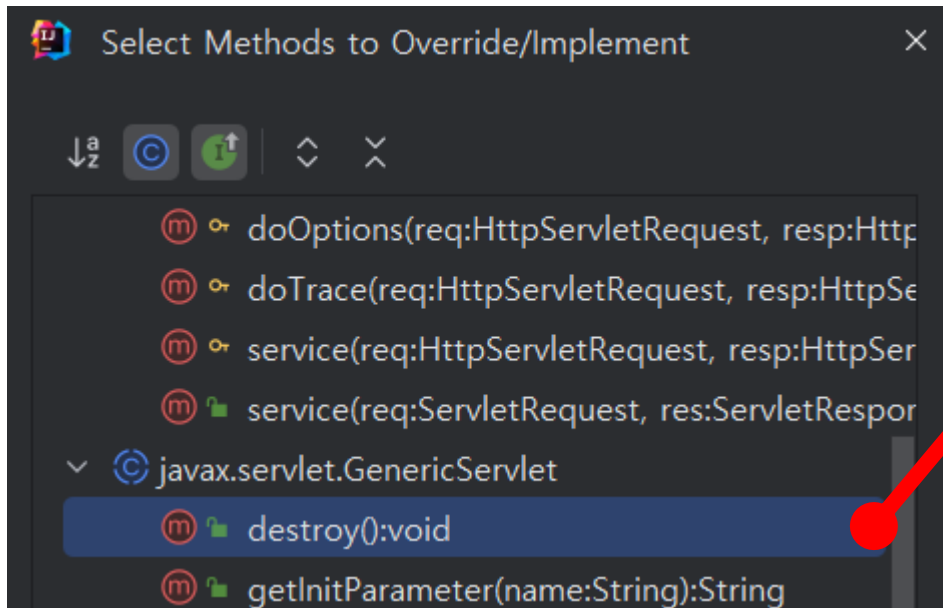


```
@Override new *  
public void init() throws ServletException {  
    System.out.println("##### init() 메서드 호출, 서블릿 초기화! #####");  
}
```

서버 로그를 통해 실행 시점을 확인하기 위해
sout 으로 문자를 출력!



doPost 아래 부분에서 동일하게
메서드 오버라이딩 선택



이번엔 destroy() 선택



로그 확인용 sout 문자 출력

```
@Override new *  
public void destroy() {  
    System.out.println("##### destroy() 메서드 호출, 서블릿 소멸! #####");  
}
```

[HOME](#) [LOGIN](#) [BOARD](#)

로그인

아이디 :

비밀번호 :

로그인 페이지로 이동하면
/login 요청이 들어가므로
LoginServlet 이 생성 됩니다!



```
[2024-07-30 08:59:37,181] Artifact Gradle : org.example :  
kb-jsp-servlet-lecture-1.0-SNAPSHOT.war: Deploy took 671 milliseconds  
##### init() 메서드 호출, 서블릿 초기화! #####
```

톰캣 서버 빌드가 끝나고
LoginServlet 이 생성 되면서
init() 메서드가 수행!



Tomcat 9.0.91 ▾



한번 생성 된 서블릿 인스턴스를
굳이 없앨 필요가 없으므로
보통 서버를 종료하면 destroy() 메서드가 실행

destroy() 메서드 호출, 서블릿 소멸!

30-Jul-2024 09:06:49.413 [main] org.apache

.destroy ["http-nio-8080"]

Disconnected from server

서버 종료를 하면
먼저 서블릿을 소멸 시킨 이 후
서버를 종료 합니다!



그란데 말입니다

그렇다면 `init()` 과 `destroy()` 는
언제 쓰면 좋을까요!?



`init()` : 서블릿 생성 시 필요한 초기화 작업
→ 리소스 자원 로드 등등

`destroy()` : 서블릿 종료 시 필요한 정리 작업
→ 리소스 자원 해제, 연결 해제 등등



당장 하자



LoginServlet

수정



LoginServlet 에서 어떤 부분을
init() / destroy() 에 적용하면 좋을까요!?



그란데 말입니다

```
private static Connection conn = null;
```

```
@Override
```

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

```
    System.out.println("##### init() 메서드 호출, 서블릿 초기화 #####");
```

```
    try {
```

```
        Class.forName("com.mysql.cj.jdbc.Driver");
```

```
        conn = DriverManager.getConnection(JDBC_URL, JDBC_USER, JDBC_PASSWORD);
```

```
        System.out.println("##### MySQL 연결 성공 #####");
```

```
    } catch (Exception e) {
```

```
        e.printStackTrace();
```

```
    }
```

```
}
```

접속 정보를 이제는 클래스 자체가
가지도록 수정!

서블릿이 생성되는 시점에
바로 JDBC 연결을 하고
해당 접속 정보를 클래스가 가지고 있는
conn 에 저장하여 사용!



```
try {  
    String sql = "SELECT * FROM users WHERE username = ? AND password = ?";  
  
    try (PreparedStatement pstmt = conn.prepareStatement(sql)) {  
        pstmt.setString(parameterIndex: 1, username);  
        pstmt.setString(parameterIndex: 2, password);  
  
        try (ResultSet rs = pstmt.executeQuery()) {  
            if (rs.next()) {  
                isLoginSuccess = true;  
            }  
        }  
    }  
}
```

JDBC 접속은 이제 init() 메서드가
담당하므로 로그인 로직 자체가
단순화되어 가독성이 올라갑니다!

@Override kdtTetz *

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

```
    try {
```

```
        conn.close();
```

```
        System.out.println("##### MySQL 접속 종료 #####");
```

```
    } catch (Exception e) {
```

```
        e.printStackTrace();
```

```
    }
```

```
    System.out.println("##### destroy() 메서드 호출, 서블릿 소멸! #####");
```

```
}
```

서블릿이 소멸 될 때,
자동으로 접속 정보도 종료하도록
destroy() 에서 close() 메서드 수행



실습, BoardServlet 에 init(), destroy() 적용

- 이제 게시판도 MySQL 과 연결하여 작업할 것이므로, Board 데이터베이스에 어제 제공된 SQL 코드를 사용하여 posts 테이블을 생성해 주세요
- BoardServlet 에도 init(), destroy() 메서드를 오버라이딩하고 각각의 메서드에서 Board 데이터 베이스에 연결, 해제하는 기능을 추가해 주세요
- Board 페이지에 들어가면 + 톨 킷 종료 시, 아래와 같은 메시지 로그가 인텔리제이에 출력이 되어야 합니다!



```
##### BoardServlet init() 메서드 호출, 서블릿 초기화 #####  
##### MySQL 연결 성공 #####
```

```
##### MySQL 접속 종료 #####  
##### BoardServlet destroy() 메서드 호출, 서블릿 소멸! #####
```

```
Disconnected from server
```

도전 실습, JDBC 추가



- posts 테이블에 주어진 SQL 구문을 사용해서 데이터 1개를 추가해 주세요
- init 메서드 구동 시, posts 테이블의 모든 데이터를 받아서 sout 으로 content 컬럼의 데이터를 출력해 주세요
- 게시판 페이지에 들어가면 인텔리제이 로그에 아래와 같은 출력이 뜨면 됩니다

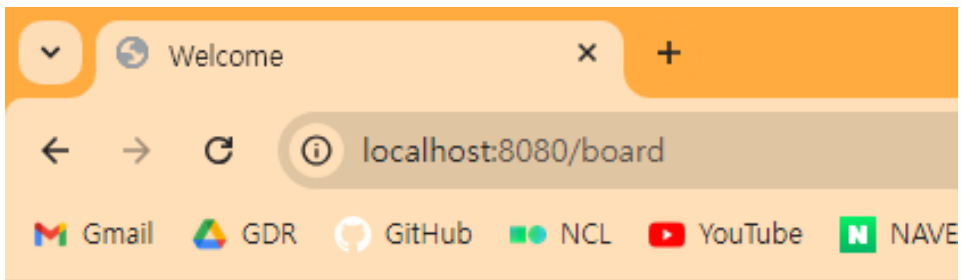
```
##### BoardServlet init() 메서드 호출, 서블릿 초기화 #####  
##### MySQL 연결 성공 #####  
테스트 글입니다! 안녕하세요
```




Servlet Filter



어제 POST 메서드로 데이터 전송 시
글자가 깨지던 것 기억 나시나요!?



[HOME](#) [LOGIN](#) [BOARD](#)

작성하신 글의 내용은

ê°ëëëëë¼

request 요청 인코딩을
UTF-8 로 변경해 줘야 합니다!

요걸 해결 하기 위해서는 !



```
@Override no usages Tetz *  
protected void doPost(HttpServletRequest request, HttpServletResponse response)  
{  
    request.setCharacterEncoding("UTF-8");  
  
    String post = request.getParameter("post");  
    request.setAttribute("post", post);  
    RequestDispatcher dispatcher = request.getRequestDispatcher("/board/write");  
    dispatcher.forward(request, response);  
}
```



그란데 말입니다

그렇다면 모든 doPost() 메서드에는
저 코드를 추가해줘야만 할까요!?

우리가 운영하는 서블릿이 1000 개 라면!?

갑자기 미국 서비스를 위해서 인코딩을
변경해야 한다면!?





Eyebrows

Bigger Eyes

Face-lift

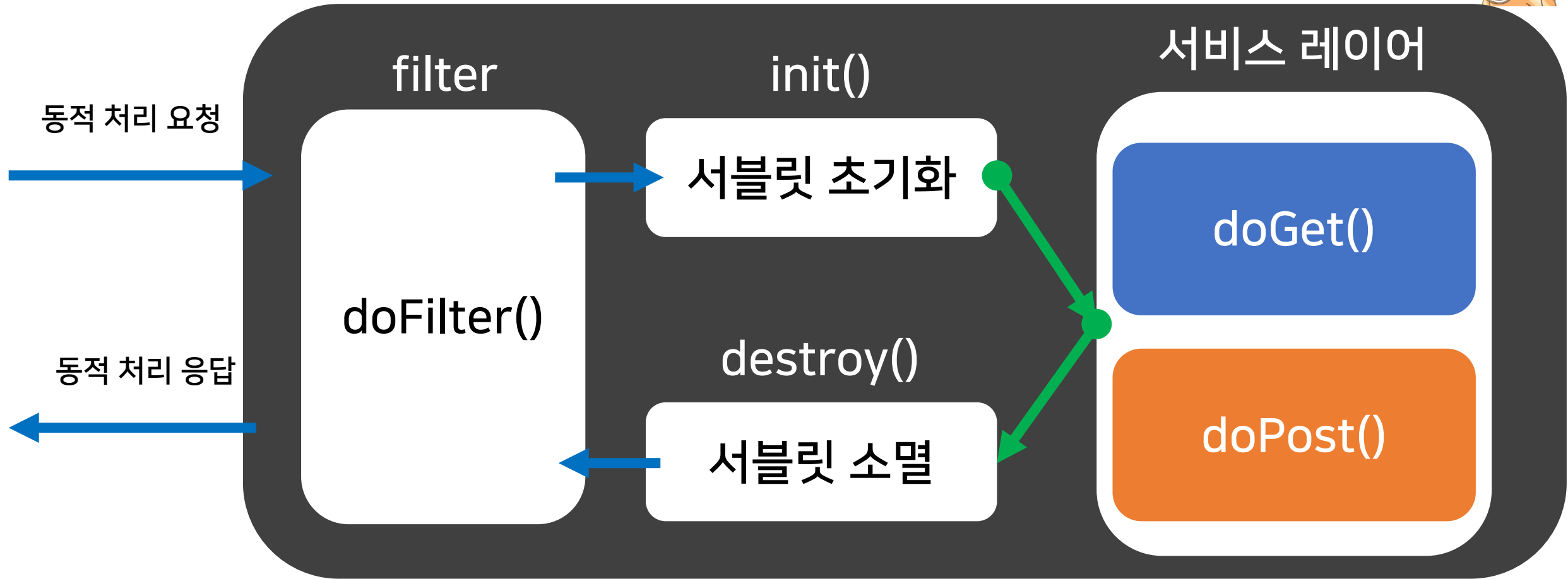
Makeup

Filter



이 몸, 등장

Servlet







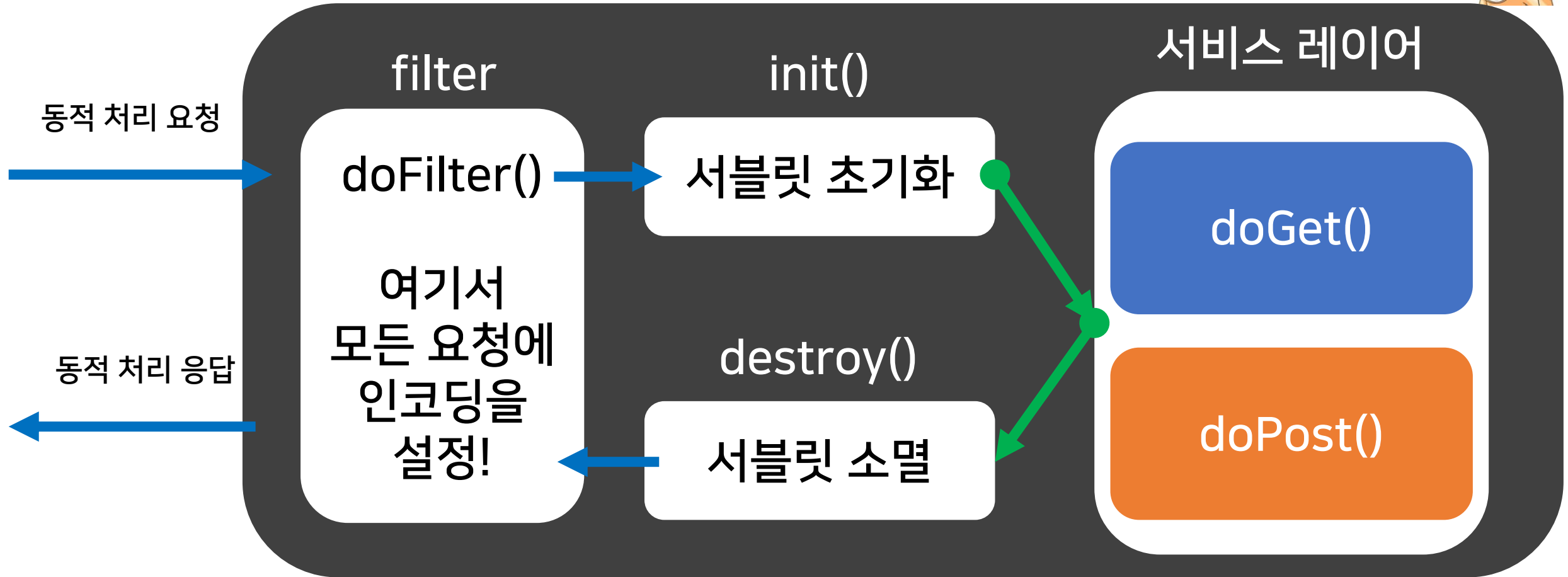
Req

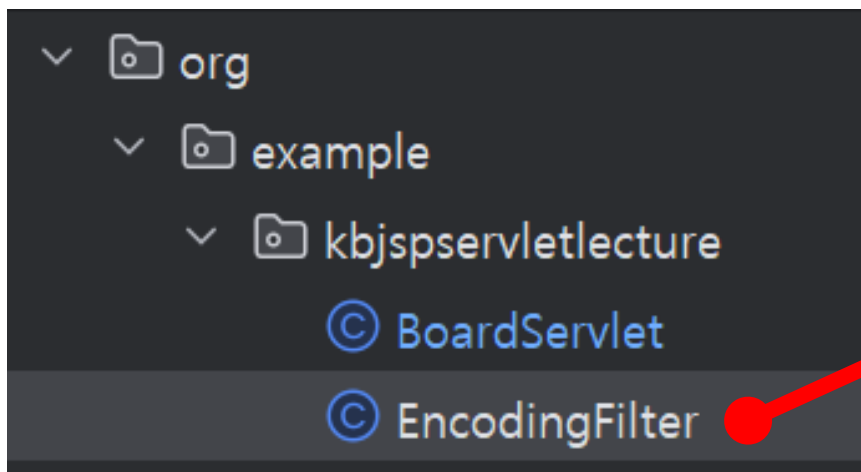
filter

- 안 돼! 못가 !! 가려거든 나 밟고 지나가!!



Servlet





인코딩 필터 적용을 위해서
EncodingFilter 클래스 생성

```
package org.example.kbjspServletlecture;
```

```
public class EncodingFilter implements Filter { 2 usages
```

```
}
```

① Filter java.util.logging

① Filter<T> java.nio.file.

① Filter javax.imageio.sp

① Filter javax.servlet

서블릿의 Filter 인터페이스를
구현하여 필터를 만듭니다!

```
public class EncodingFilter impl
```

```
}
```

Generate

Constructor

toString()

Override Methods... Ctrl+O

인터페이스의 메서드 구현을 위해
Override 메서드 선택

Select Methods to Override/Implement

↓ a
z



▼ javax.servlet.Filter

init(filterConfig:FilterConfig):void

destroy():void

doFilter(servletRequest:ServletRequest, servletResponse:ServletResponse):void

모든 메서드를 오버라이딩!

```
public class EncodingFilter implements Filter { 2 usages
    @Override ① kdtTetz *
    public void init(FilterConfig filterConfig) throws Se
        Filter.super.init(filterConfig);
    }
    @Override ② kdtTetz *
    public void destroy() {
        Filter.super.destroy();
    }
    @Override no usages new *
    public void doFilter(ServletRequest servletRequest, S

}
```

자동으로 생성된
코드를 보면 어떤걸 알 수 있나요?

질문1) 3개의 메서드 중
특성이 다른 하나의 메서드는?

질문2) 그럼 나머지 2개의
메서드는 xx 메서드입니다.
xx 에 들어갈 말은!?

```
public class EncodingFilter implements Filter { 2 usages
    @Override ① kdtTetz *
    public void init(FilterConfig filterConfig) throws ServletException {
        Filter.super.init(filterConfig);
    }
    @Override ① kdtTetz *
    public void destroy() {
        Filter.super.destroy();
    }
    @Override no usages new *
    public void doFilter(ServletRequest servletRequest, ServletResponse servletResponse, FilterChain chain) throws IOException, ServletException {
    }
}
```

init(), destroy() 메서드는
구현된 부분이 부모에 존재합니다!

즉, 둘은 추상메서드입니다!
따라서 삭제해도 문제가 안됩니다!

```
public class EncodingFilter implements Filter {  
    @Override  
    public void doFilter(ServletRequest servletRequest,  
        ServletResponse servletResponse, FilterChain chain)  
        throws IOException, ServletException {  
    }  
}
```

우리한테 필요한 +
순수 인터페이스인 doFilter 만
구현합니다!


```
public class EncodingFilter implements Filter { 2 usages kdtTetz
    @Override no usages kdtTetz *
    public void doFilter(ServletRequest request, ServletResponse
        throws IOException, ServletException {
        request.setCharacterEncoding("UTF-8");
        response.setContentType("text/html; charset=UTF-8");

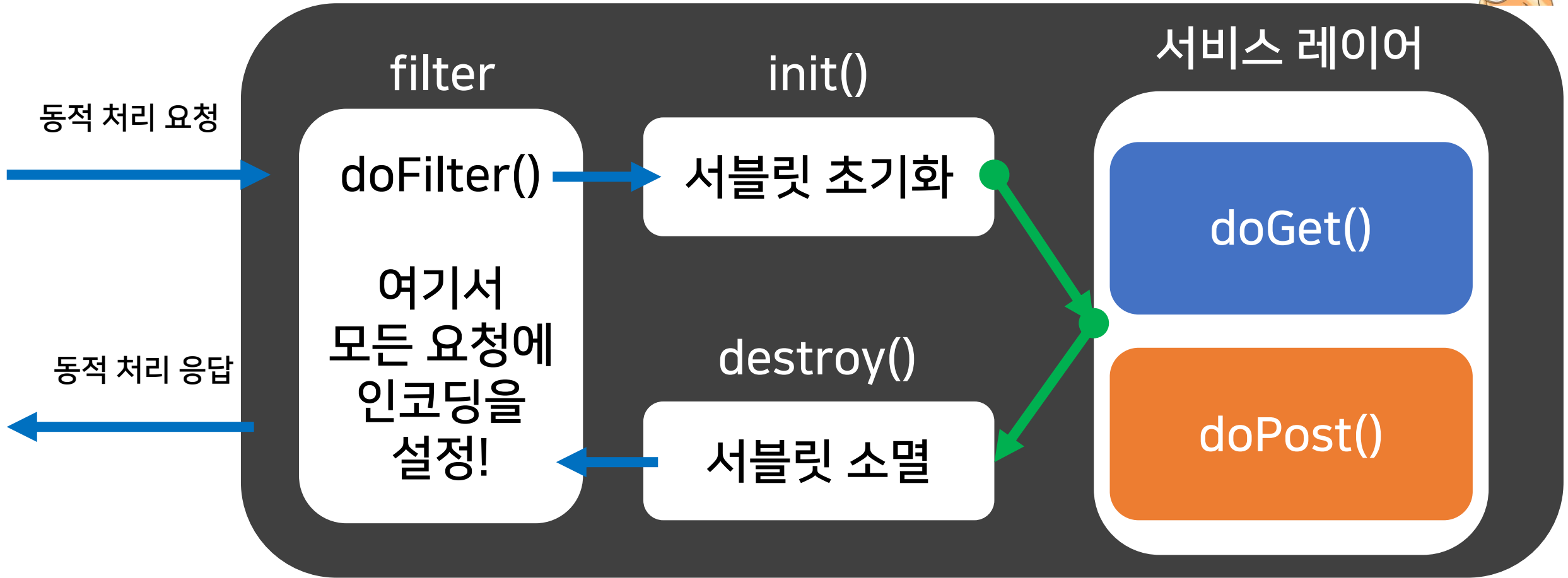
        chain.doFilter(request, response);
    }
}
```

request 에
필요한 인코딩 및
컨텐츠타입을 설정 합니다

또 다른 필터 혹은,
필터를 지나서
실제 비즈니스 로직을 처리하는
서블릿에 필터 된 req, res 를 전달



Servlet





저 필터의 존재를 톰캣 서버가 알고 있나요!?



그란데 말입니다



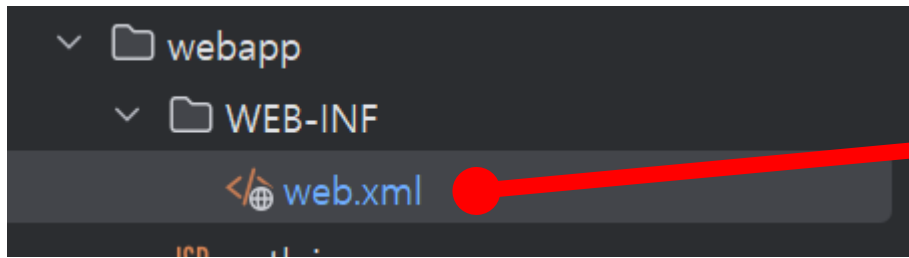


툼캣 서버

모르는데 어떻게 가요!



필터



설정은 web.xml 을 통해 합니다!





```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee http://xmlns.jcp.org/xml
  version="4.0">
  <filter>
    <filter-name>EncodingFilter</filter-name>
    <filter-class>org.example.kbjspServletlecture.EncodingFilter</filter-class>
  </filter>

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

필터를 등록!

어떤 주소 요청에
필터를 적용할지 설정



귀찮다!

요즘 개발자



@WebFilter
어노테이션으로 간단하게 등록!

```
@WebFilter(urlPatterns = "/*") kdtTetz *  
public class EncodingFilter implements Filter {  
    @Override no usages kdtTetz *
```



실습, CountFilter 만들기!



- 서버에 총 몇번의 요청이 왔는지를 기록할 수 있는 CountFilter 를 만들어 주세요
- 해당 필터는 요청이 총 몇 번 들어왔는지를 멤버 변수 count 에 저장하고 있으며, 서버에 들어오는 모든 요청 마다 필터가 작동 count 값이 +1 됩니다
- 그리고 필터가 동작하면 현재 자신의 count 값을 출력합니다
- 서비스를 구동하고 페이지를 이동하면 아래와 같은 내용이 인텔리제이에 출력 되도록 CountFilter 를 만들어 주세요
- 실습 문제가 어려운 분들은 다다음 장의 코드를 보고 빈칸을 채워주세요!



```
Filter 가 작동한 횟수는 : 1
Filter 가 작동한 횟수는 : 2
Filter 가 작동한 횟수는 : 3
##### LoginServlet init() 메서드 호출,
##### MySQL 연결 성공 #####
Filter 가 작동한 횟수는 : 4
##### BoardServlet init() 메서드 호출,
##### MySQL 연결 성공 #####
테스트 글입니다! 안녕하세요
Filter 가 작동한 횟수는 : 5
Filter 가 작동한 횟수는 : 6
Filter 가 작동한 횟수는 : 7
Filter 가 작동한 횟수는 : 8
```



```
// 모든 주소 요청에 작동하기 위해서 /* 주소 패턴 적용
```

```
@WebFilter(urlPatterns = "/*") new *
```

```
public class CountFilter implements Filter {
```

```
    // 왜 static 이 붙었을까요?
```

```
    private static int count = 0;          no usages
```

```
@Override no usages new *
```

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

```
    // ##### 해당 부분을 완성해 주세요
```

```
    // 다음 필터 or 서블릿으로 요청을 넘기기 위한 Chain 전달
```

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

```
}
```

```
}
```

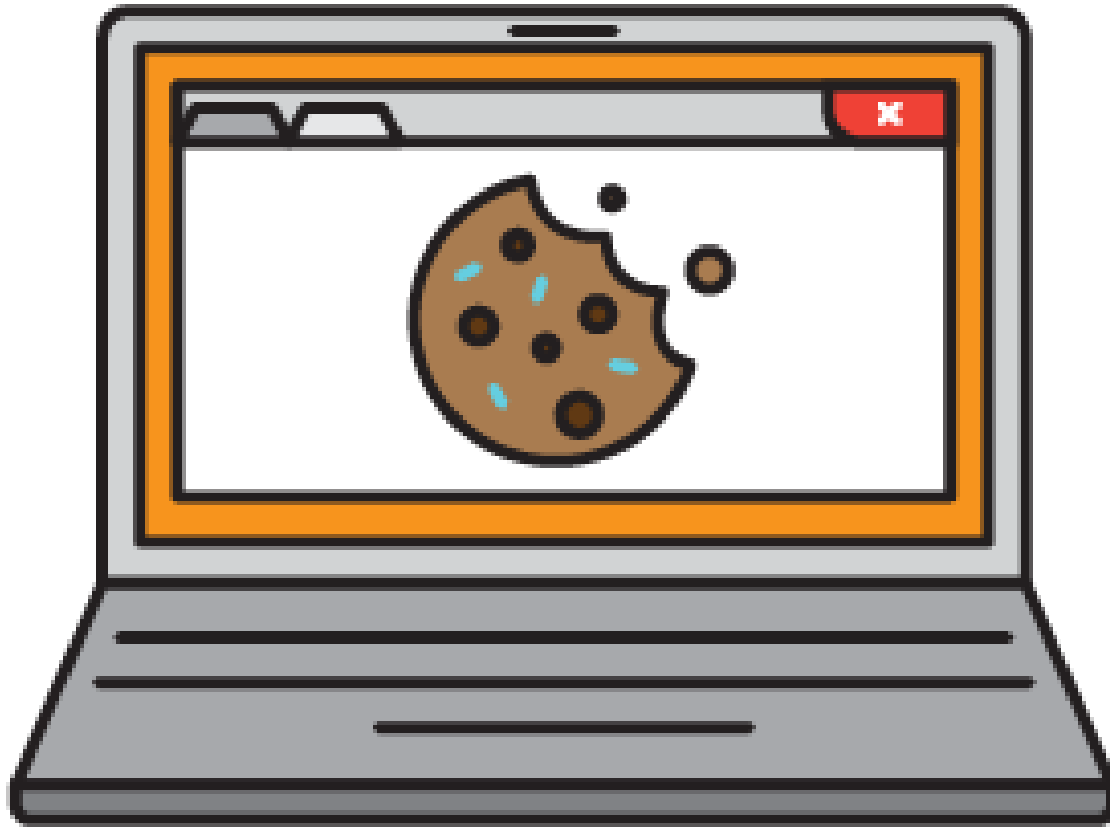
이 부분을 완성해 주세요!



쿠키



Cookie



WEBSITE COOKIES

오리온

달지않고 바삭한

초코칩 쿠키

Original

초코칩 15% 함유 과자 104g (533 kcal)



오해 나만 시켜!

글·그림 이주희

별아!



BACK-END


별아!





장바구니

☒



100000011

유리 토끼 바디모찌 쿠션
인형

10,500 원

- 1 +

×

상품금액 10,500 원 / 수량 1 개

총 10,500 원

☒



658745636

짱구 파자마
의류

28,000 원

- 1 +

×

상품금액 28,000 원 / 수량 1 개

총 28,000 원

☒

주문 정보

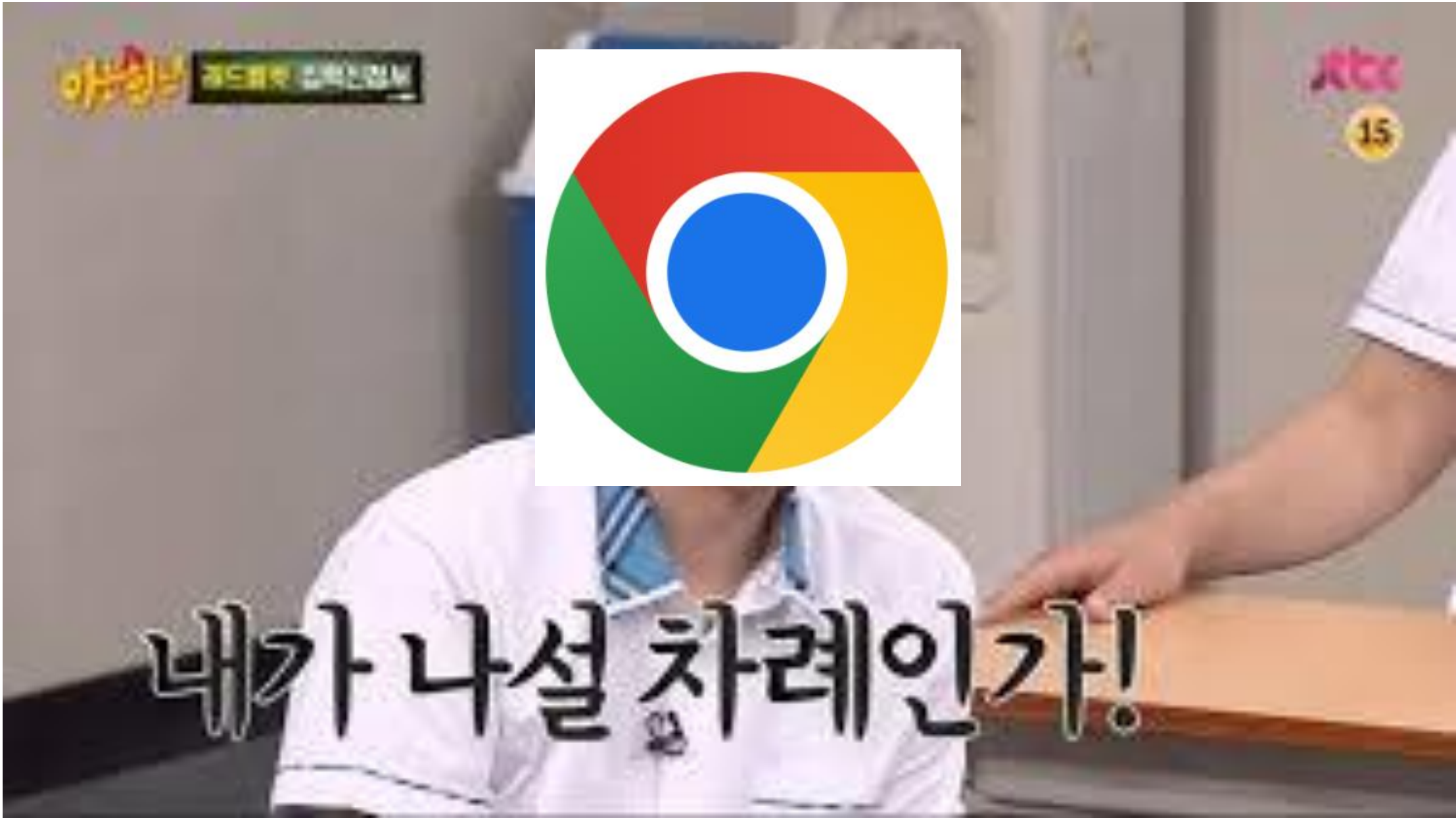
총 수량	6 개
총 상품 금액	88,300 원
배송비	3,000 원
총 주문금액	91,300 원

주문하기

전체삭제



효율이 없다 아니까, 효율이 .



아는힘
공드림부 김희진장세

15

내가 나설 차례인가!

Tomcat Server



첫 Reqeust

Response





**A FEW
MOMENTS LATER**



Tomcat Server



두번째 Reqeust





웹 브라우저

1. HTTP 요청



2. 쿠키를 HTTP 헤더(Set-Cookie)에 담아서 응답



3. 쿠키 저장 / HTTP 재요청 (쿠키 포함)



4. 응답 (쿠키 업데이트 시 전달)



웹 서버


```
@WebServlet("/setCookie")
```

/setCookie 주소 매핑



```
public class CookieServlet extends HttpServlet {
```

```
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
```

```
        throws ServletException, IOException {
```

```
        // 쿠키 생성
```

```
        Cookie userCookie = new Cookie(name: "username", value: "tetz");
```

```
        // 쿠키의 유효 기간을 설정
```

```
        userCookie.setMaxAge(7 * 24 * 60 * 60); // 7일
```

쿠키 생성 및 설정

```
        // 응답에 쿠키 추가
```

```
        response.addCookie(userCookie);
```

응답에 추가

```
        // 응답 내용 작성
```

```
        response.setContentType("text/html; charset=UTF-8");
```

```
        response.setCharacterEncoding("UTF-8");
```

```
        response.getWriter().println("<html><body>");
```

```
        response.getWriter().println("<h1>쿠키가 설정되었습니다.</h1>");
```

```
        response.getWriter().println("</body></html>");
```

응답 전송!

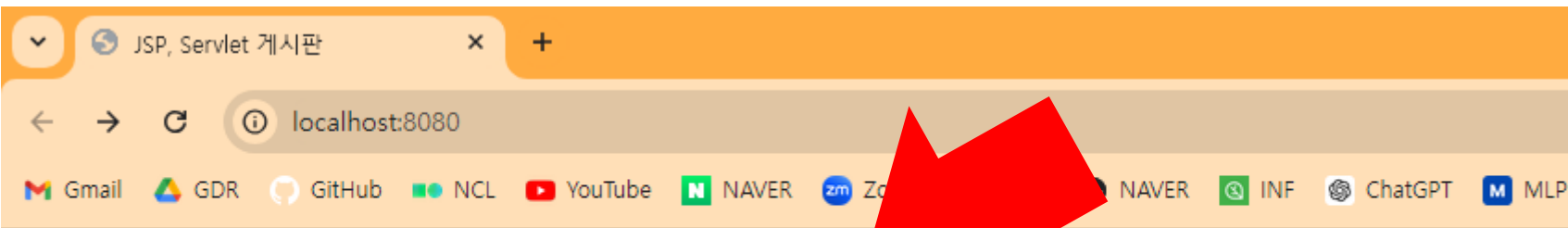
```
    }
```

```
}
```



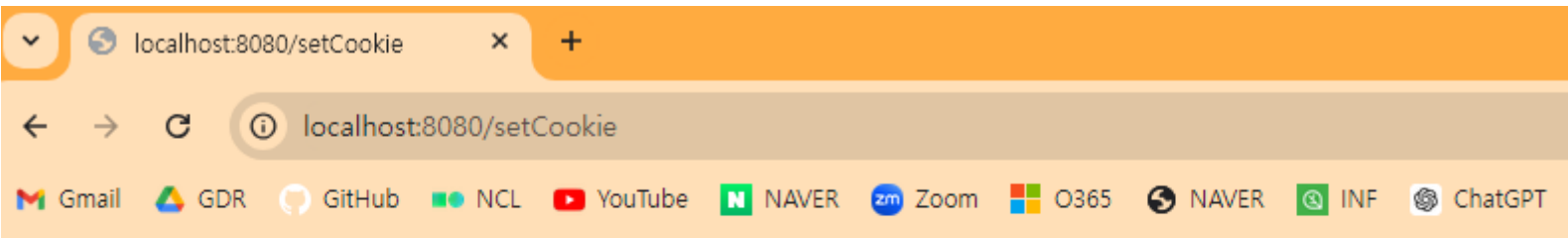

```
<%@ page contentType="text/html;charse  
<header>  
  <a href="/">HOME</a>  
  <a href="login">LOGIN</a>  
  <a href="board">BOARD</a>  
  <a href="setCookie">COOKIE</a>  
</header>
```

헤더 메뉴에
쿠키 요청을 위한
링크 추가



[HOME](#) [LOGIN](#) [BOARD](#) [COOKIE](#)

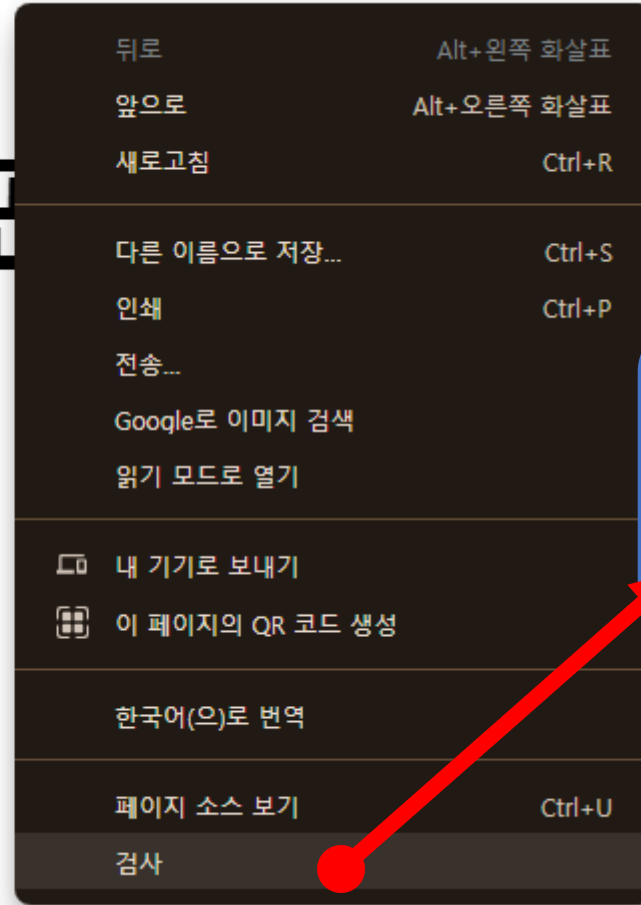
JSP, Servlet 게시판 입니다



쿠키가 설정되었습니다.



ASP, Servlet 게시판



검사 창 띄우기!



DevTools is now available in Korean! [Always match Chrome's language](#) [Switch DevTools to Korean](#) [Don't show again](#)

Elements Console Sources Network Performance Memory **Application** Security >> ⚙️ ⋮ ✕

Application

- Manifest
- Service workers
- Storage

Storage

- Local storage
- Session storage
- IndexedDB
- ▼ Cookies
- http://localhost:8080
- Private state tokens

Filter Only show cookies with an issue

Name	Value	D...	Path	Ex...	Size	Ht...	Se...	St...	Pa...	Cr...	Pri...
JSESSIONID	5130D7E48E6FFB7820F45...	lo...	/	Se...	42	✓					M...
username	tetz	lo...	/	20...	12						M...

어플리케이션 선택

쿠키 탭에서 현재 보고 있는
웹페이지의 주소를 클릭!

DevTools is now available in Korean! [Always match Chrome's language](#) [Switch DevTools to Korean](#) [Don't show again](#)

Elements Console Sources Network Performance Memory **Application** Security >> ⚙️ ⋮ ✕

Application

- Manifest
- Service workers
- Storage

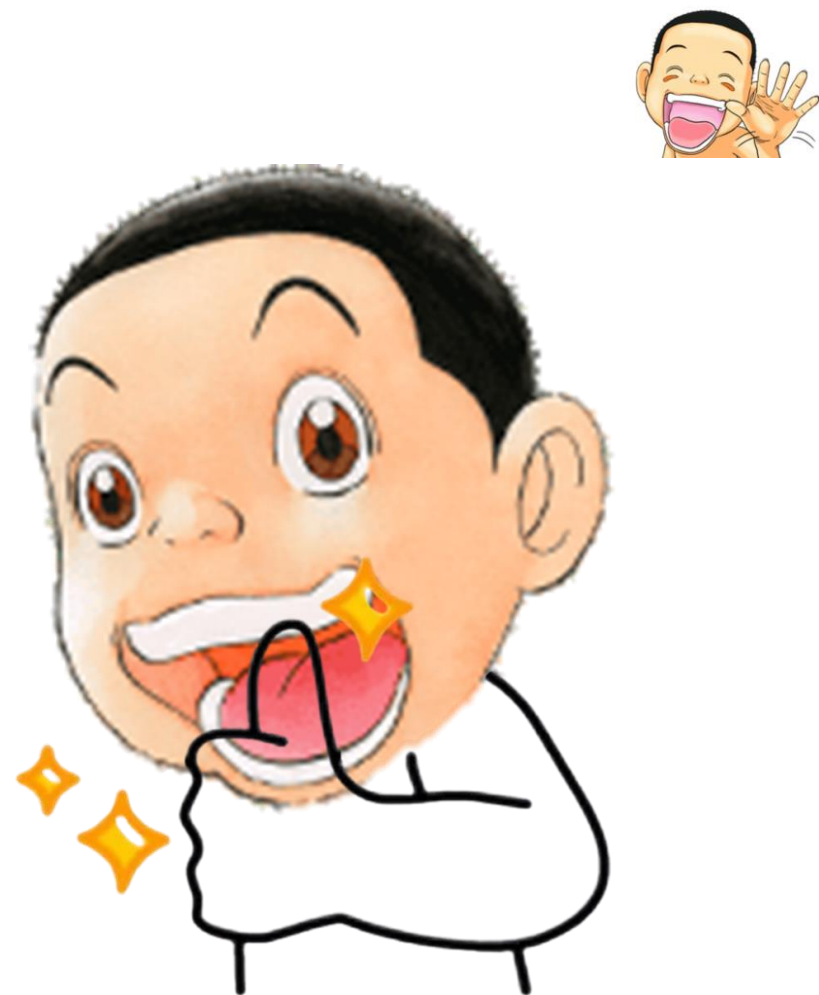
Storage

- Local storage
- Session storage
- IndexedDB
- ▼ Cookies
- http://localhost:8080
- Private state tokens

Filter Only show cookies with an issue

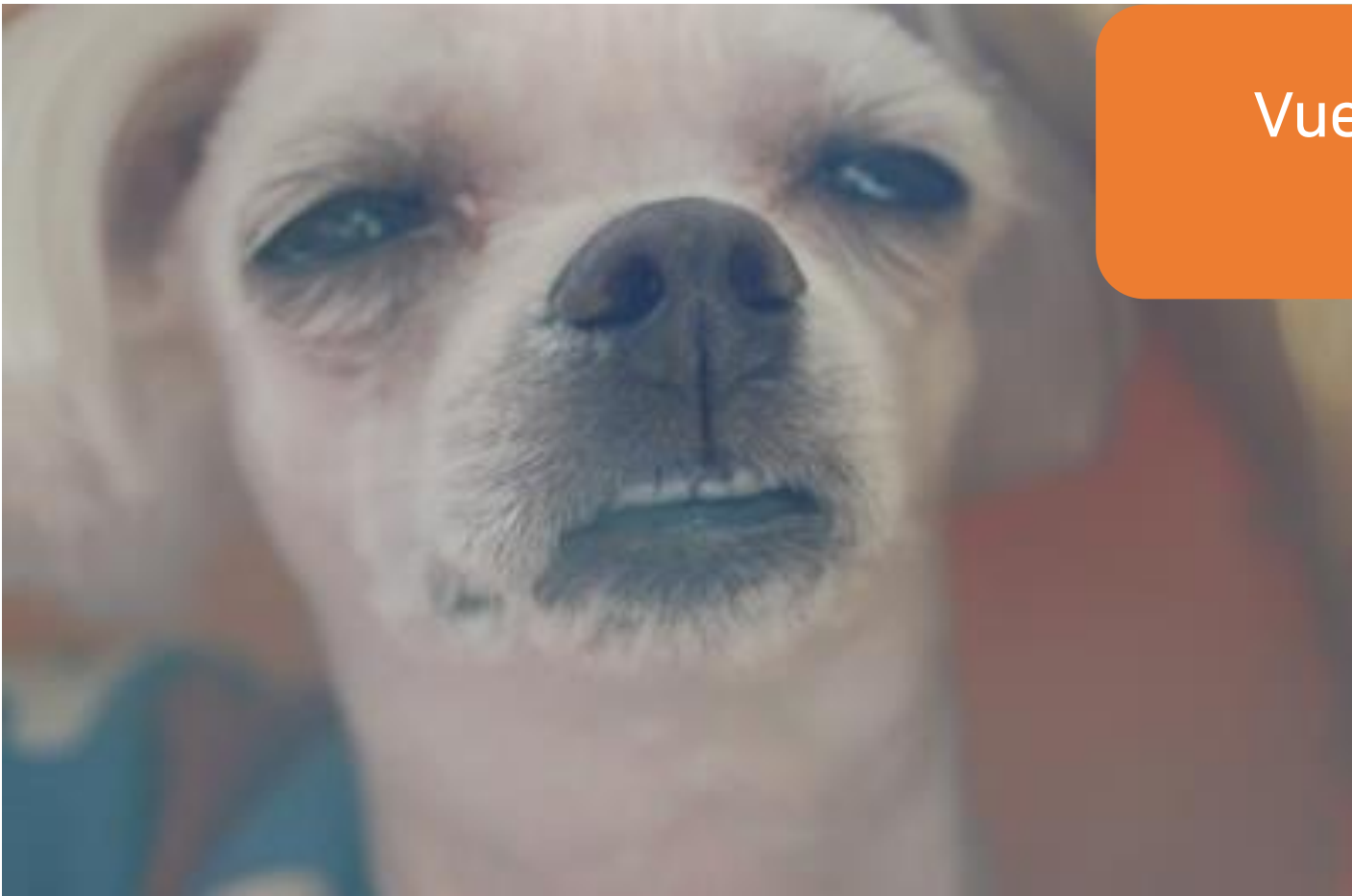
Name	Value	D...	Path	Ex...	Size	Ht...	Se...	Sa...	Pa...	Cr...	Pri...
JSESSIONID	5130D7E48E6FFB7820F45...	lo...	/	Se...	42	✓					M...
username	tetz	lo...	/	20...	12						M...

우리가 보낸 쿠키 확인!!





그란데 말입니다



Vue 에서 쿠키 비스무리한거 배웠었는데
기억 하시나요!?



[Info] 로컬 스토리지 VS 세션 스토리지



이현우 ▶

웃긴걸 찾는 사람들 (웃찾사-유머, 개그, 짤, ...)

3일 • 🌐

가짜사나이 3기 촬영 현장
악플 달지마



좋아요



공유하기

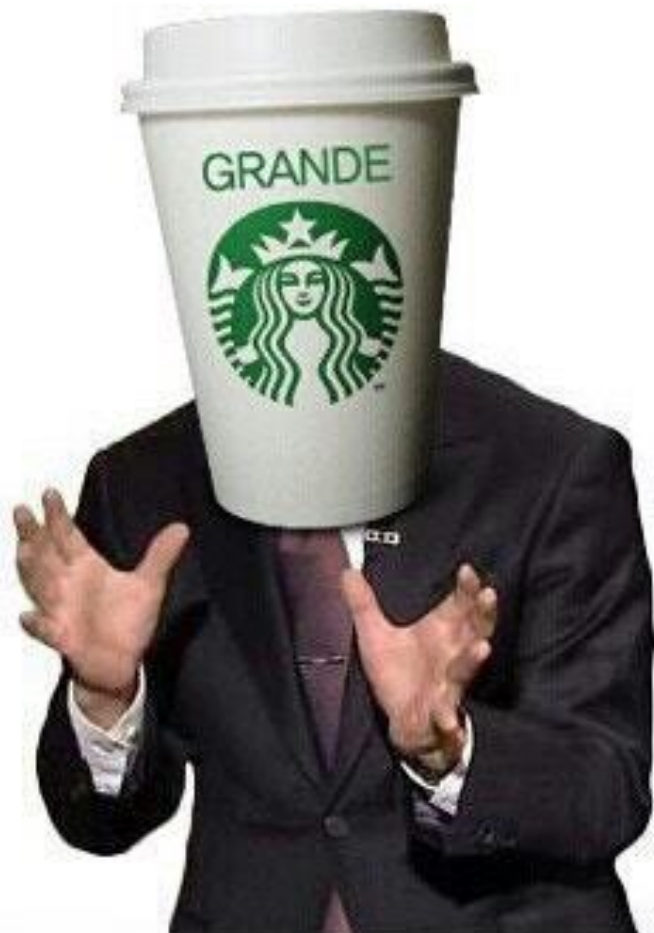




아~ 완벽히 이해했어!



이해못했음



우리가 보낸 쿠키 말고 위에 있는 애는 누구죠?



DevTools is now available in Korean! [Always match Chrome's language](#) [Switch DevTools to Korean](#) [Don't show again](#)

Elements Console Sources Network Performance Memory **Application** Security >> ⚙️ ⋮ ✕

Application 🔄 ⌵ ✕ ☐ Only show cookies with an issue

	Name ▲	Value	D...	Path	Ex...	Size	Ht...	Se...	Sa...	Pa...	Cr...	Pri...
Service workers	JSESSIONID	5130D7E48E6FFB7820F45...	lo...	/	Se...	42	✓					M...
	username	tetz	lo...	/	20...	12						M...
Storage												
	▶ Local storage											
	▶ Session storage											
	IndexedDB											
Cookies	▼											
	http://localhost:8080											
Private state tokens												

그란데 말입니다



세션

→ 서버의 쿠키

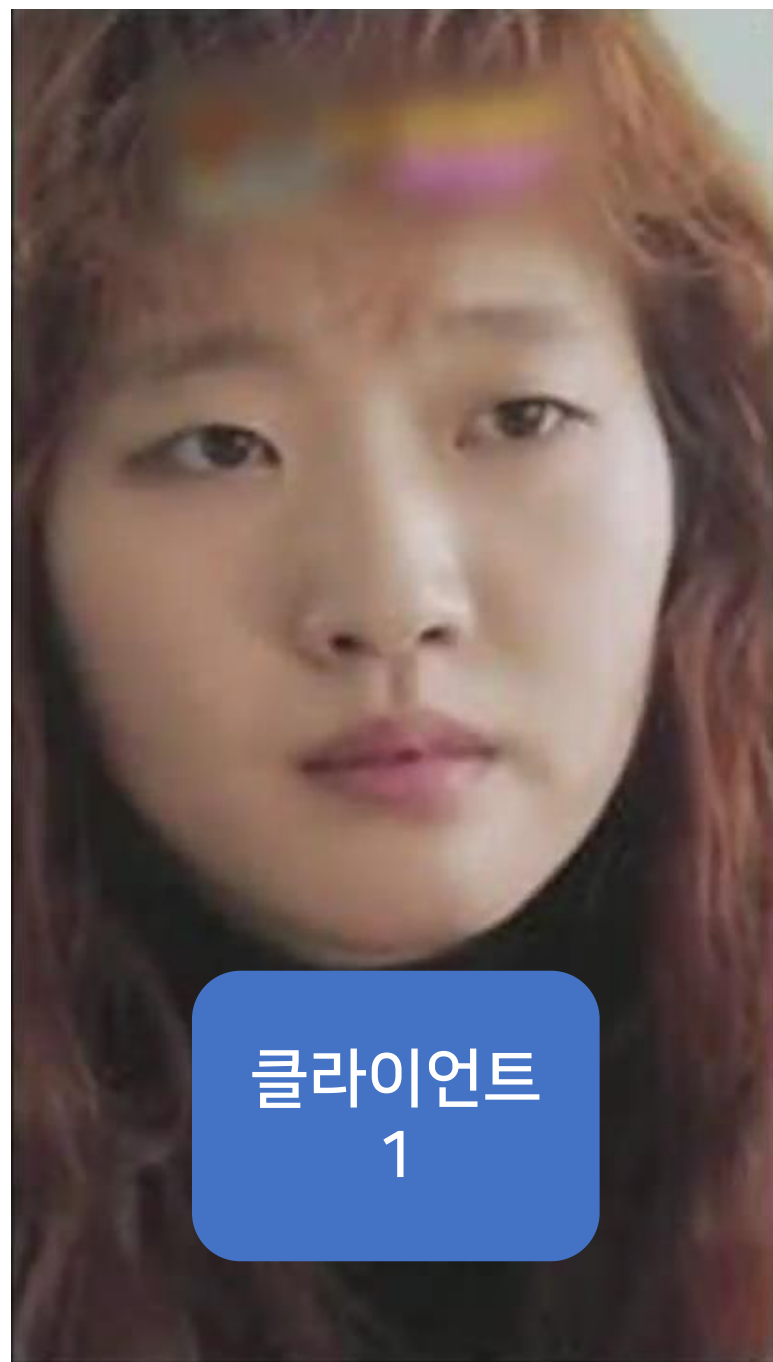




그런 건
없다.



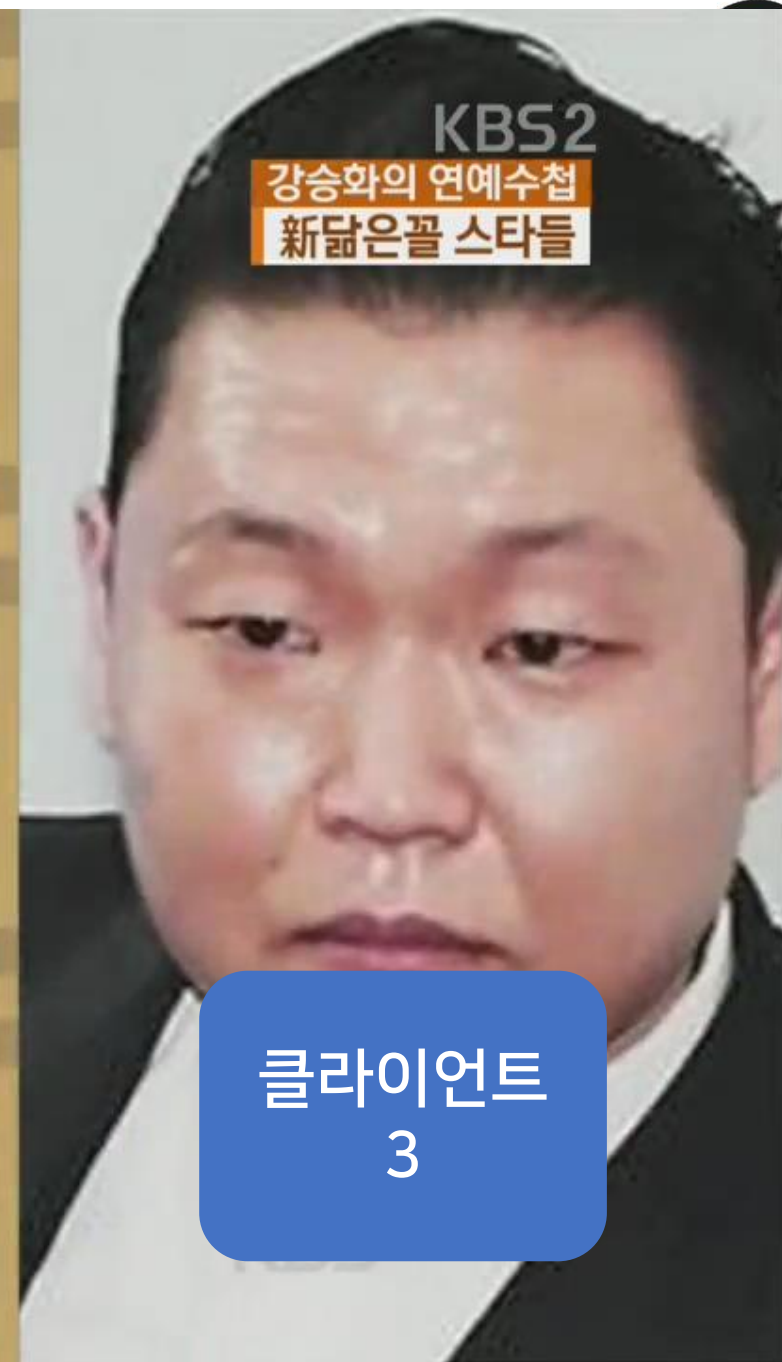




클라이언트
1



클라이언트
2



클라이언트
3

KBS2

강승화의 연예수첩

新답은꼴 스타들





서버 (Server)



한 집에서 시작된 거대한 거대한 현의 이야기

너 이름 뭐야

어디서나 전번파라

그대 만난 날 모든 것을,
그대 없으면 사라진다

2017년 1월 이름다운 현피가 시작된다



서버는 클라이언트를
구분하기 위해서
이름표(=세션 아이디)를
부여 합니다!



Application

Manifest

Service workers

Storage

Storage

Local storage

Session storage

IndexedDB

Cookies

http://localhost:8080

Private state tokens

Filter

Only show cookies with an issue

Name	Value	D...	Path	Ex...	Size	Ht...	Se...	Sa...	Pa...	Cr...	Pri...
JSESSIONID	5130D7E48E6FFB7820F45...	lo...	/	Se...	42	✓					M...
username	tetz	lo...	/	20...	12						M...

요게 서버에서 클라이언트를 구분하기 위해서 임의로 부여한 SessionID 문자열 입니다!

Tomcat Server



첫 Reqeust

Response



abcdefg



임의로 발급된 세션
abcdefg

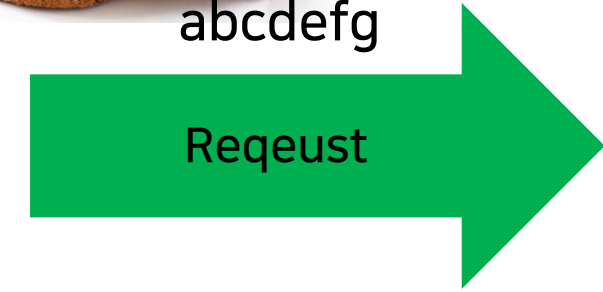


Tomcat Server



abcdefg

Reqeust



Response



임의로 발급된 세션
abcdefg





HTTP Session 의 특징!

- 브라우저가 아닌 서버에 저장되는 쿠키
- 사용자가 서버에 접속한 시점부터 연결을 끝내는 시점을 하나의 상태로 보고 유지하는 기능을 함 → 로그인 유지
- 서버는 각 사용자에게 대한 세션을 발행하고 서버로 접근(Request)한 사용자를 식별하는 도구로 사용
- 쿠키와 달리 저장 데이터에 제한이 없음
- 만료 기간 설정이 가능하지만, 브라우저가 종료되면 바로 삭제



그란데 말입니다

그럼 세션을 어느 기능을 구현하는데 쓰면 좋을까요?



카카오 로그인

N

네이버 로그인



그란데 말입니다

그럼 세션과 쿠키는 어떤 차이를 가질까요!?

질문1) 보안은 누가 더 좋을까요?

질문2) 접근 및 처리 속도는 누가 더 빠를까요?





세션으로

로그인 구현하기!

© LoginServlet

로그인 비즈니스 로직을 처리하는
로그인 서블릿을 수정 합니다!



로그인이 성공하면 지금은 request
요청의 속성 값에 사용자 정보를 저장!

→ 세션을 이용하여 세션에 저장

```
if (isLoginSuccess) {  
    request.setAttribute(s: "username", username);  
    RequestDispatcher dispatcher = request.getRequestDispatcher(s: "welcome.jsp");  
    dispatcher.forward(request, response);  
} else {  
    response.sendRedirect(s: "loginFailed.jsp");  
}  
}
```

서버가 가지고 있는 세션을 가지고 옵니다!

```
if (isLoginSuccess) {  
    HttpSession session = request.getSession();  
    session.setAttribute(s: "username", username);  
  
    RequestDispatcher dispatcher = request.getRequestDispatcher(s: "welcome.jsp");  
    dispatcher.forward(request, response);  
} else {  
    response.sendRedirect(s: "loginFailed.jsp");  
}
```

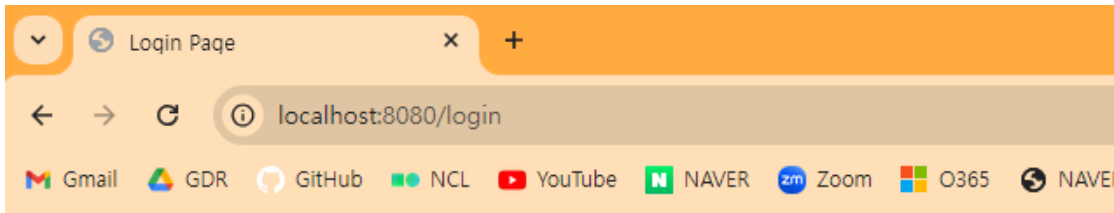
세션 객체에 속성으로
사용자 정보를 저장!

welcome.jsp 파일



```
<%@ page contentType="text/html; charset=UTF-8" language="java" pageEncoding="UTF-8"%>
<html>
<head>
    <title>Welcome</title>
</head>
<%@ include file="header.jsp" %>
<body>
    <h1>로그인 성공!</h1>
    <h2>환영합니다! <%= session.getAttribute("userid") %> 님</h2>
    <h2>환영합니다! ${username} 님</h2>
</body>
</html>
```

정보를 세션에서 가져 오도록 수정!



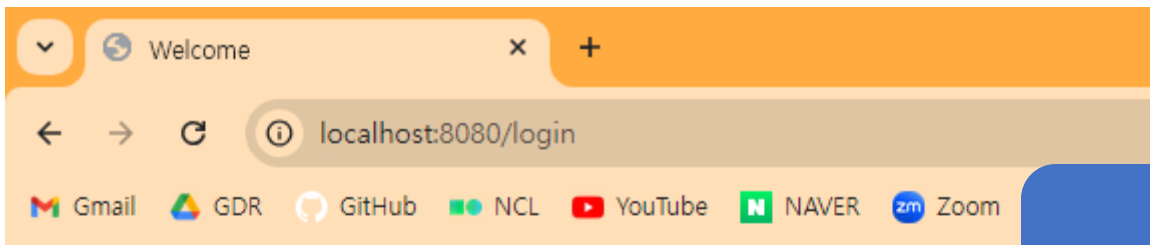
[HOME](#) [LOGIN](#) [BOARD](#) [COOKIE](#)

로그인

아이디 :

비밀번호 :

로그인



세션에서 정보를 성공적으로 가져와서 출력!

[HOME](#) [LOGIN](#) [BOARD](#) [COOKIE](#)

로그인 성공!

환영합니다! tetz 님

환영합니다! tetz 님





세션과 필터로

찐 로그인 기능 구현!



Session

Filter





그란데 말입니다

이제 게시판은 로그인한 사용자만
접근이 가능하도록 수정할 예정입니다!

어떻게 하면 될까요!?

필요사항1) 해당 클라이언트가 로그인이 되었는지
확인이 가능해야 함

필요사항2) 게시판으로 접속하는 모든 요청에
대해서 로그인 된 사용자임을 확인 필요



어디서든
특정 클라이언트가
로그인이 되었는지
확인 가능!

Session

게시판으로 가능
모든 요청에 대한
특정 작업 수행 가능!

Filter





Tetz



로그인이 성공하면 이제 게시판 페이지로
이동을 시킵니다!

```
if (isLoginSuccess) {  
    HttpSession session = request.getSession();  
    session.setAttribute(s: "username", username);  
  
    response.sendRedirect(s: "/board");  
} else {  
    response.sendRedirect(s: "loginFailed.jsp");  
}
```



EyeBrows

Bigger Eyes

Face-lift

Makeup

Filter

/board 요청이 들어오면 일단 LoginFilter 를
거쳐야만 합니다!

```
@WebFilter("/board")
public class LoginFilter implements Filter {
    @Override
    public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
        throws IOException, ServletException {
        HttpServletRequest httpRequest = (HttpServletRequest) request;
        HttpServletResponse httpResponse = (HttpServletResponse) response;

        HttpSession session = httpRequest.getSession(false);
        boolean loggedIn = (session != null && session.getAttribute("username") != null);

        if (loggedIn) {
            chain.doFilter(request, response);
        } else {
            httpResponse.sendRedirect("/auth.jsp");
        }
    }
}
```

```
@WebFilter("/board") ① kdtTetz *
public class LoginFilter implements Filter {
    ② @Override no usages kdtTetz *
    public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
        throws IOException, ServletException {
        HttpServletRequest httpRequest = (HttpServletRequest) request;
        HttpServletResponse httpResponse = (HttpServletResponse) response;

        HttpSession session = httpRequest.getSession(b: false);
        boolean loggedIn = (session != null && session.getAttribute(s: "username") != null);

        if (loggedIn) {
            chain.doFilter(request, response);
        } else {
            httpResponse.sendRedirect(s: "/auth.jsp");
        }
    }
}
```

세션 사용이 가능하도록 타입을 캐스팅!

세션에 사용자가 로그인 되었는지를 확인!



```
@WebFilter("/board") ① kdtTetz *
public class LoginFilter implements Filter {
    @Override  no usages ② kdtTetz *
    public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
        throws IOException, ServletException {
        HttpServletRequest httpRequest = (HttpServletRequest) request;
        HttpServletResponse httpResponse = (HttpServletResponse) response;

        HttpSession session = httpRequest.getSession(b: false);
        boolean loggedIn = (session != null && session.getAttribute(s: "username") != null);

        if (loggedIn) {
            chain.doFilter(request, response);
        } else {
            httpResponse.sendRedirect(s: "/auth.jsp");
        }
    }
}
```

로그인이 되었으면 기존 요청으로 보내기

로그인이 안되었으면 로그인 필요하다는
내용을 보여주는 페이지로 보내기

auth.jsp 파일

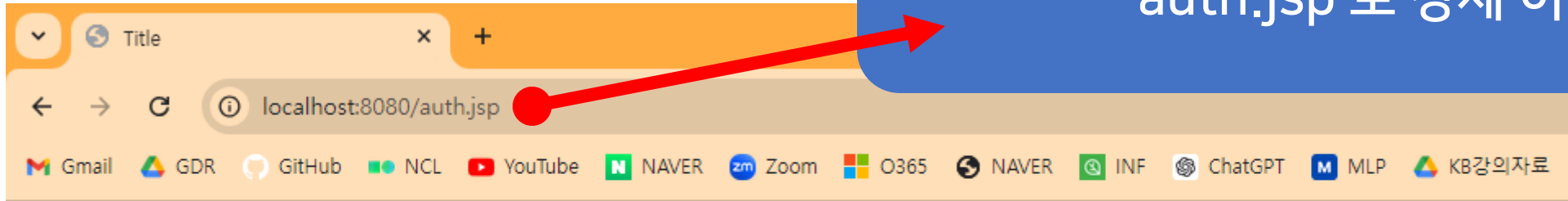


```
<%@ page contentType="text/html; charset=UTF-8" language="java" pageEncoding="UTF-8"%>
<html>
<head>
    <title>Title</title>
</head>
<%@ include file="header.jsp" %>
<body>
    <h1>로그인이 필요한 서비스 입니다</h1>
    <a href="login"><button type="button">로그인 페이지로</button></a>
</body>
</html>
```




자~ 드가자~

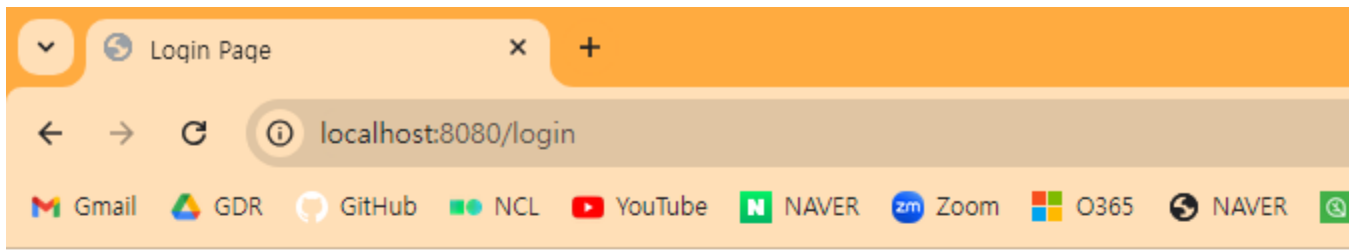
바로 BOARD 로 이동을 하면 로그인 필터에 걸려서
auth.jsp 로 강제 이동 됩니다!



[HOME](#) [LOGIN](#) [BOARD](#) [COOKIE](#)

로그인이 필요한 서비스 입니다

로그인 페이지로



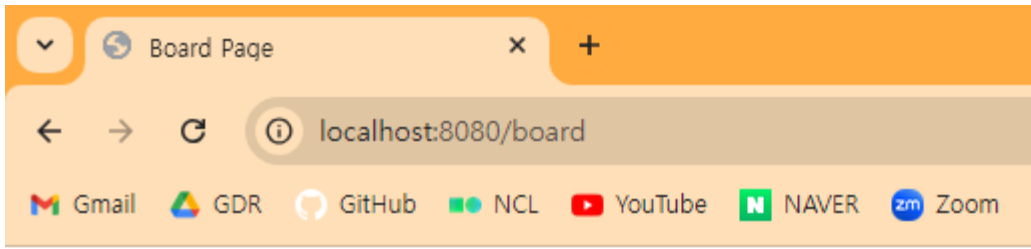
[HOME](#) [LOGIN](#) [BOARD](#) [COOKIE](#)

로그인

아이디 :

비밀번호 :

로그인



로그인이 성공하면 자동으로 /board 요청으로 Redirect 되고 게시판 페이지로 이동 합니다!

[HOME](#) [LOGIN](#) [BOARD](#) [COOKIE](#)

게시판 페이지 입니다

글 내용 :

세션이 살아 있는 동안에는
= 브라우저가 종료 안된 상태
= 세션의 MaxAge 기간 내

다른 곳에 이동을 했다가 와도
게시판 페이지에 접근이 가능합니다!

