



It's Your Life

with





Login 의 개념



그란데 말입니다

로그인을 구현할 때
제가 xx 를 쓴다고 했었는데 기억 하시는 분!?

로그인 접속한 클라이언트를 구분한 상태에서
진행해야 하므로 Session 을 사용합니다!

그란데 말입니다



그란데 말입니다

그란데 말입니다

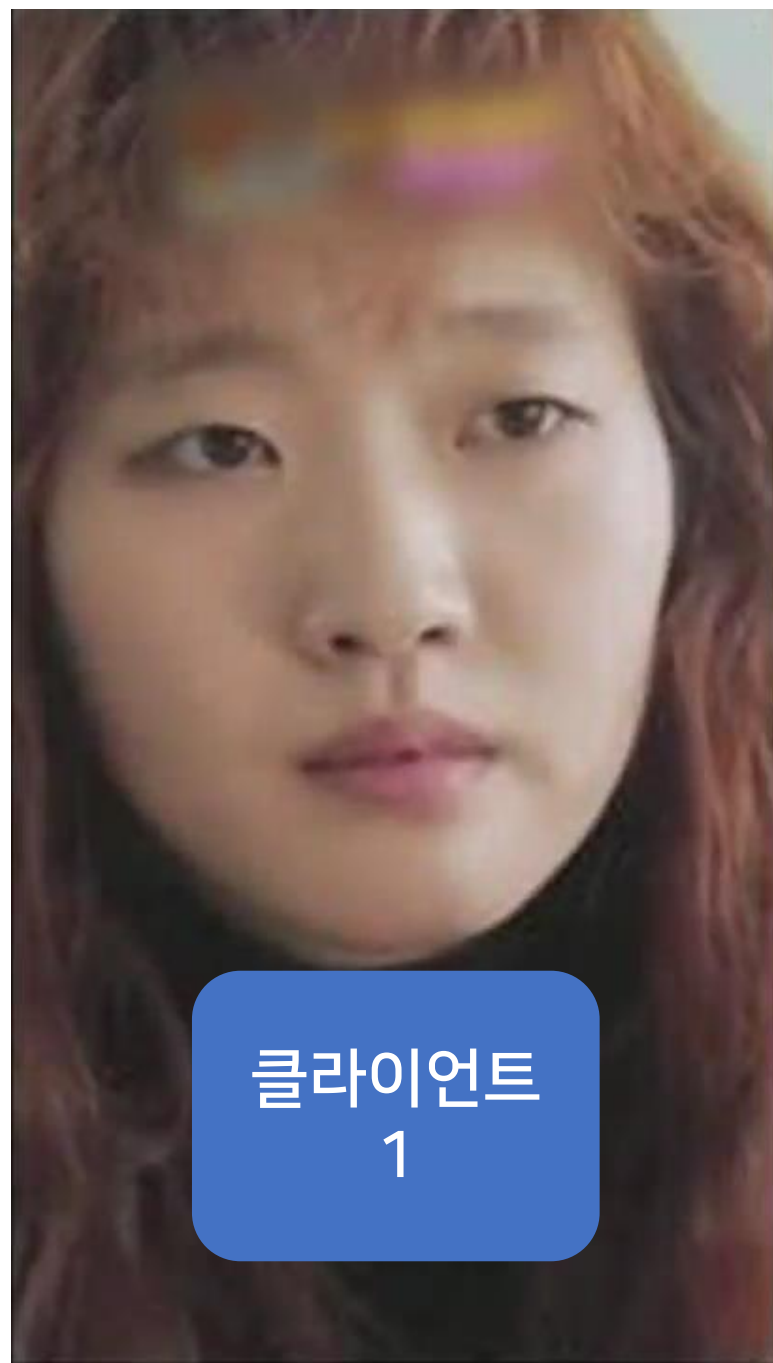
세션이 뭐였죠!?





세션

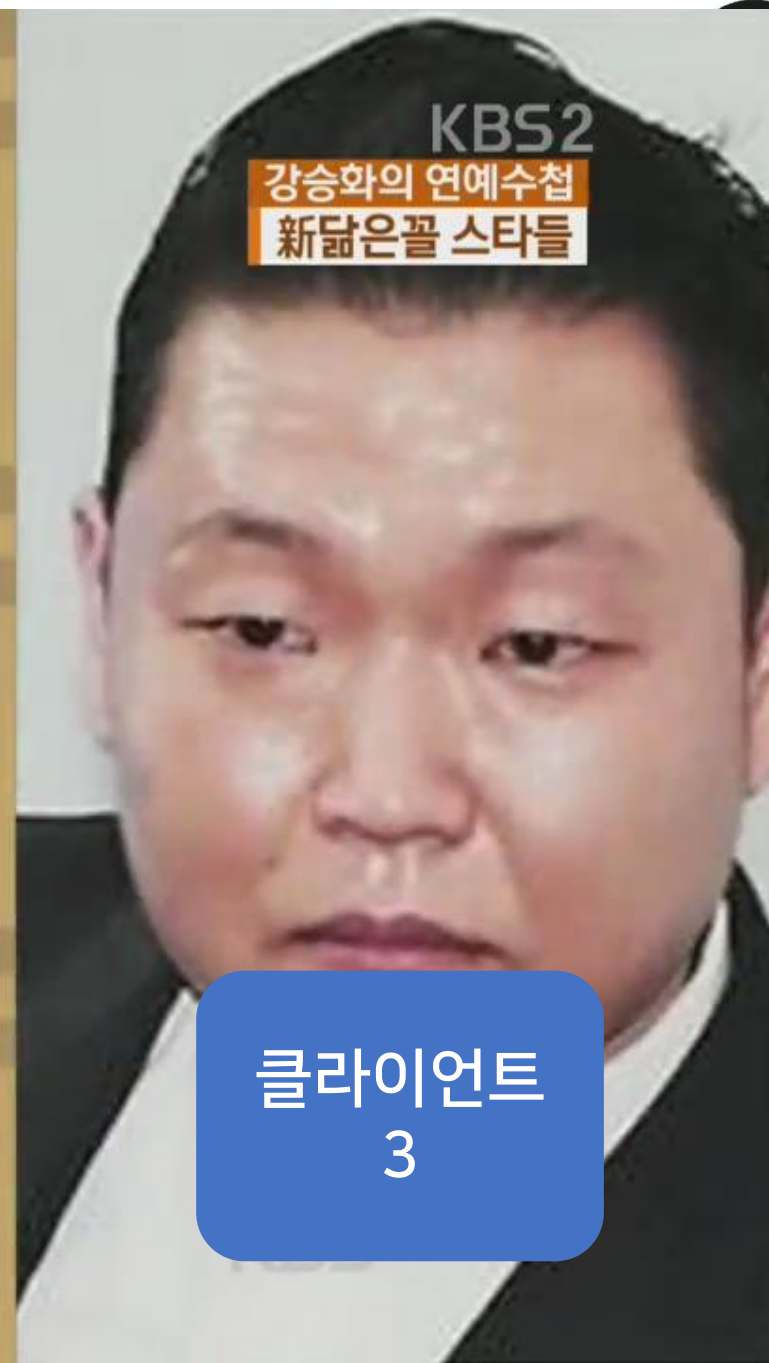
→ 서버의 쿠키



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
 - http://localhost:8080
- Session storage
- IndexedDB
- Cookies
 - http://localhost:8080

Name	Value	D...	P...	E...	Si...	H...	S
JSESSIONID	13785CC991353BBE08...		/	S...	42	✓	

크롬에서 확인 가능한 세션 쿠키

응용 프로그램

- 매니페스트
- Service workers
- 저장소

저장소

- 로컬 저장소
- 세션 저장소
- IndexedDB
- 쿠키
 - http://localhost:8080

이름	값	D...	P...	E...	크
JSESSIONID	4131721237B49EE67A...		/	세...	4

엣지에서 확인 가능한 세션 쿠키



Tomcat Server



첫 Reqeust

임의로 발급된 세션
abcdefg



Response



abcdefg

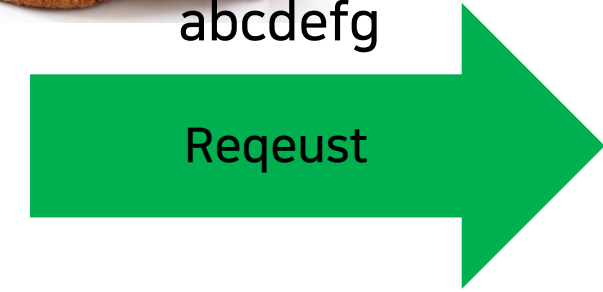


Tomcat Server



abcdefg

Reqeust



Response



이미 발급된 세션
abcdefg





Tomcat Server



첫 Reqeust

임의로 발급된 세션
abcdefg



Response



abcdefg



The screenshot shows the Chrome DevTools Application tab. The left sidebar has a tree view with 'Application' expanded, showing 'Manifest', 'Service workers', and 'Storage'. Under 'Storage', 'Cookies' is selected, and 'http://localhost:8080' is chosen. The main pane displays a table of cookies. A red dot is placed on the 'JSESSIONID' row, with a red arrow pointing to the Korean text box on the right.

Name	Value	D...	P...	E...	Si...	H...	S
JSESSIONID	13785CC991353BBE08...		/	S...	42	✓	

크롬에서 확인 가능한 세션 쿠키

The screenshot shows the Edge DevTools Application tab. The left sidebar has a tree view with '응용 프로그램' (Application) expanded, showing '매니페스트' (Manifest), 'Service workers', and '저장소' (Storage). Under '저장소', '로컬 저장소' (Local storage) is expanded, and '세션 저장소' (Session storage) is selected. The main pane displays a table of cookies. A red dot is placed on the 'JSESSIONID' row, with a red arrow pointing to the Korean text box on the right.

이름	값	D...	P...	E...	크...
JSESSIONID	4131721237B49EE67A...		/	세...	4

엣지에서 확인 가능한 세션 쿠키



HTTP Session 의 특징!

- 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 로그인에 왜 세션이 필요하죠!?

클라이언트가 누군지를 알아야 해당 클라이언트가
로그인에서 문제가 발생하지 않기 때문입니다!

그란데 말입니다



서버 (Server)

JSESSIONID

13785CC991353BBE08...



JSESSIONID

4131721237B49EE67A..

그런 건
없다.

JSESSIONID

13785CC991353BBE08...



서버 (Server)

조승상의
통행증이 없으면
강을 건널 수
없다.





JSESSIONID

4131721237B49EE67A..





Login 구현 V1



Login 을 위한

DB 만들기



	id	username	password	roles
▶	1	12	12	ROLE_MEMBER
	2	tetz	12	ROLE_MEMBER
	3	siwan	12	ROLE_MEMBER
✱	NULL	NULL	NULL	NULL

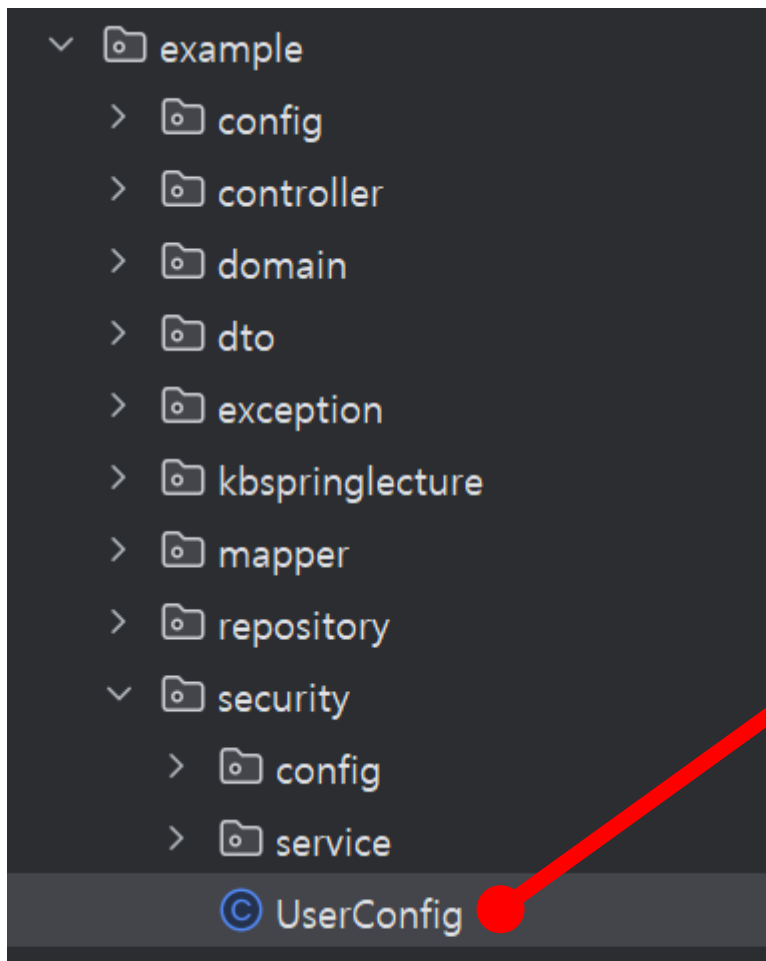
<https://github.com/xenosign/spring-code-repo/blob/main/sql/users.sql>

```
1 • USE mybatis;
2
3 • CREATE TABLE users (
4     id INT AUTO_INCREMENT PRIMARY KEY,
5     username VARCHAR(20) NOT NULL UNIQUE,
6     password VARCHAR(100) NOT NULL,
7     roles VARCHAR(100) NOT NULL default 'ROLE_MEMBER'
8 );
9
10 • INSERT INTO users (username, password)
11     VALUES ("12", "12"),
12     ("tetz", "12"),
13     ("siwan", "12");
14
15 • SELECT * FROM users;
```



Login 을 위한

기본 세팅하기



사용자 로그인 세팅을 위해
security 패키지 만들기

security 하위에 UserConfig 만들기!

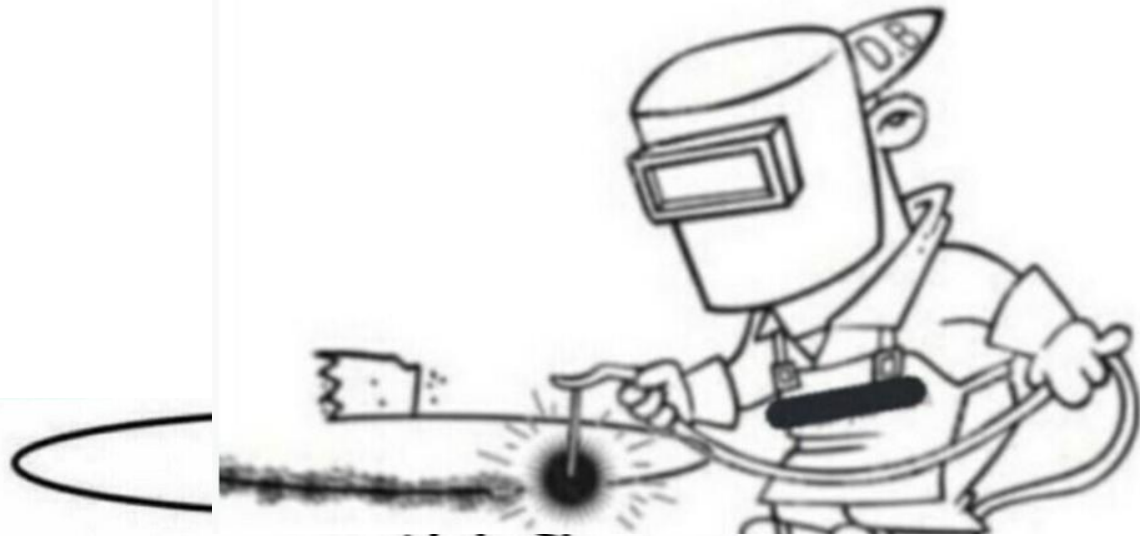
클라이언트의 요청이 들어올 때
해당 요청을 훔쳐서(= Intercept)
로그인 여부를 확인하는
Authentication 과정을 추가!

```
@Configuration  👤 kdtTetz *
public class UserConfig implements WebMvcConfigurer {
    // 인터셉터 추가
    @Override  no usages  👤 kdtTetz
    public void addInterceptors(InterceptorRegistry registry) {
        System.out.println("시큐리티 인터셉터가 등록되었습니다.");
        registry.addInterceptor(new AuthenticationInterceptor())
            .addPathPatterns("/**")
            .excludePathPatterns("/", "/user/**", "/resources/**");
    }
}
```

일단 모든 주소에 대한 접근에 로그인 여부를 확인

단, 기본 주소와 로그인을 위한 /user 의 하위 주소는
예외로 등록하여 접근이 가능하도록 설정!

어림도 없다!

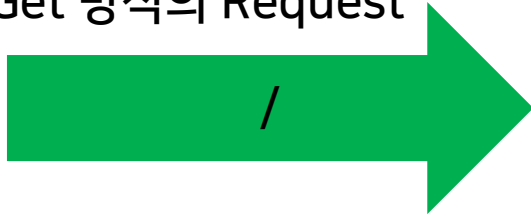


신경 쓰지 마, 난
그냥 위의 찰을
흠치고있어





Get 방식의 Request



Tomcat Server

index.jsp

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

서버에서 보내는
Response



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

Get 방식의 Request

/

Tomcat Server

index.jsp

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

인터넷에 의해 요청 된 요청

/login



Get 방식의 Request



Tomcat Server

index.jsp

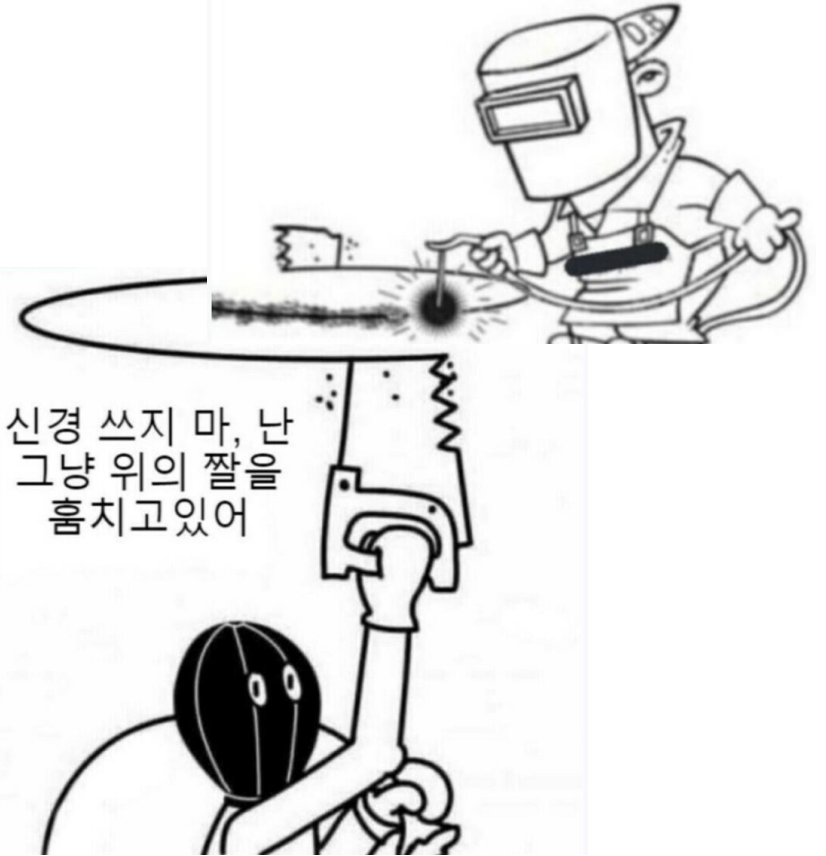
```
<html>  
<% String str = "Hello, World" %>  
<h1><%= str %></h1>  
</html>
```

서버에서 보내는
Response



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

어림도 없다!



신경 쓰지 마, 난
그냥 위의 찰을
흠치고있어



- example
 - config
 - controller
 - domain
 - dto
 - exception
 - kbspringlecture
 - mapper
 - repository
 - security
 - config
 - service

© AuthenticationInterceptor

© UserConfig

클라이언트의 요청을 가로채서
실제 로그인 과정을 처리하는
AuthenticationInterceptor 클래스 추가



@Component kdtTetz

```
public class AuthenticationInterceptor implements HandlerInterceptor {  
    @Override no usages kdtTetz  
    public boolean preHandle(HttpServletRequest request, HttpServletResponse response, Object handler  
        throws Exception {  
        String requestUri = request.getRequestURI();  
        System.out.println("요청 URI: " + requestUri);  
  
        HttpSession session = request.getSession();  
        if (session.getAttribute(s: "loginUser") == null) {  
            System.out.println("로그인 안됨. 리다이렉트 중: /user/login");  
            response.sendRedirect(s: "/user/login");  
            return false;  
        }  
        return true;  
    }  
}
```

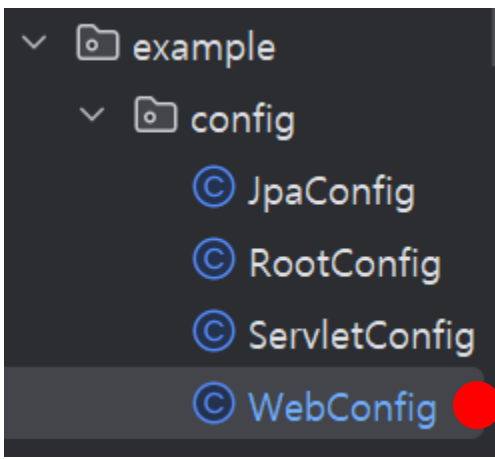
클라이언트의 요청이
어디로 오는지 확인하는 코드

세션에 사용자가 로그인 되었는지를 확인하고
안되었으면 로그인 페이지로 돌려보내는 코드



WebConfig 에

설정 적용



방금 만든 설정을
WEbConfig 에 적용시켜 봅시다!

지금 부터는 사용 및 관리가 편리한
@Import 어노테이션을 통해서
설정 클래스를 추가!

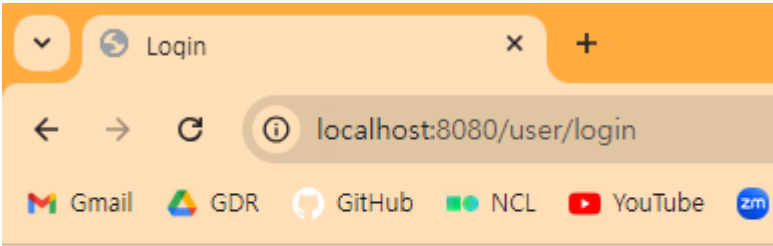
```
@Configuration  👤 kdtTetz
@Import(UserConfig.class)
public class WebConfig extends AbstractAnnotationConfig
{
    @Override  no usages  👤 kdtTetz
    protected Class<?>[] getRootConfigClasses() {
        return new Class[]{
            RootConfig.class, JpaConfig.class
        };
    }
}
```

기존에는 설정 적용을 위해서
getRootConfigClasses 에
설정 클래스를 추가하는 방식을 사용!



Interceptor

기능 확인



V1 MyBatis

[HOME](#) [게시글 목록](#) [404 error](#)

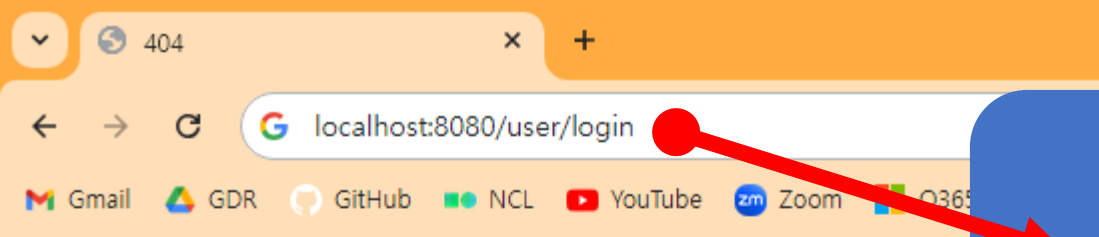
V1 REST

[HOME](#) [게시글 목록](#)

V1 JPA

[HOME](#) [게시글 목록](#)

아무 링크나 클릭을 해봅시다!



해당 주소 요청은 인터셉터에 의해 인터셉트
→ 로그인 안되었으므로 설정한
/user/login 으로 리다이렉트 됩니다!!

V1 MyBatis

[HOME](#) [게시글 목록](#) [404 error](#)

V1 REST

[HOME](#) [게시글 목록](#)

V1 JPA

[HOME](#) [게시글 목록](#)

회원 기능

[회원가입](#) [로그인](#) [로그아웃](#) [admin](#)

404, 존재하지 않는 페이지 입니다

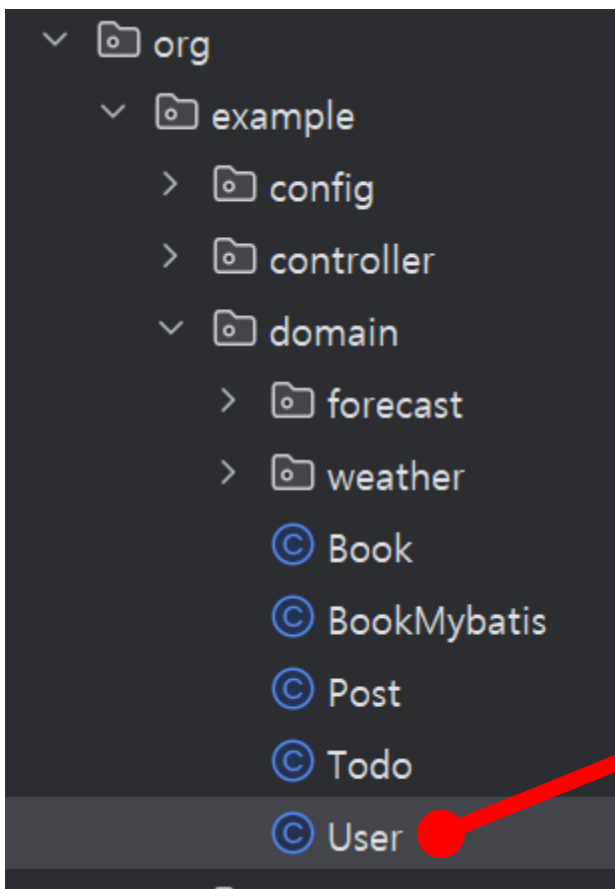
[홈 페이지로 돌아가기](#)

해당 주소는 매핑된 컨트롤러가 없으므로
404 페이지가 리턴



Login 기능 구현

User Entity 만들기



로그인 처리를 위한
User Entity 클래스 생성



최초 생성한 users 테이블에 맞는
JPA Entity 클래스를 작성!

@Entity 18 usages kdtTetz

@Data

@NoArgsConstructor

@AllArgsConstructor

@Table(name = "users")

public class User {

@Id

@GeneratedValue(strategy = GenerationType.IDENTITY)

private Long id;

@Column(unique = true, nullable = false)

private String username;

@Column(nullable = false)

private String password;

@Column(nullable = false)

private String roles;

}



UserRepository

만들기



▼ repository

> book

> post

> todo

▼ user

© UserRepository

사용자 DB 와 통신을 담당하는
UserRepository 클래스 생성



```
@Repository  👤 kdtTetz *
```

```
@Transactional
```

```
@RequiredArgsConstructor
```

```
public class UserRepository {
```

```
    private final EntityManager em;
```

필요 어노테이션 추가하기!

데이터 관련 처리를 담당하는
EntityManager 주입 받기



```
public User findByUsername(String username) { 2 usages kdtTetz
    String jpql = "SELECT u FROM User u WHERE u.username = :username";
    List<User> users = em.createQuery(jpql, User.class)
        .setParameter(s: "username", username)
        .getResultList();

    return users.isEmpty() ? null : users.get(0);
}
```

DB 에 해당 id 를 가지는 사용자가
존재하는지를 찾는 메서드

리스트가 비었으면 null 을 리턴하여
해당 사용자가 없음을 표기



UserService

만들기



그란데 말입니다

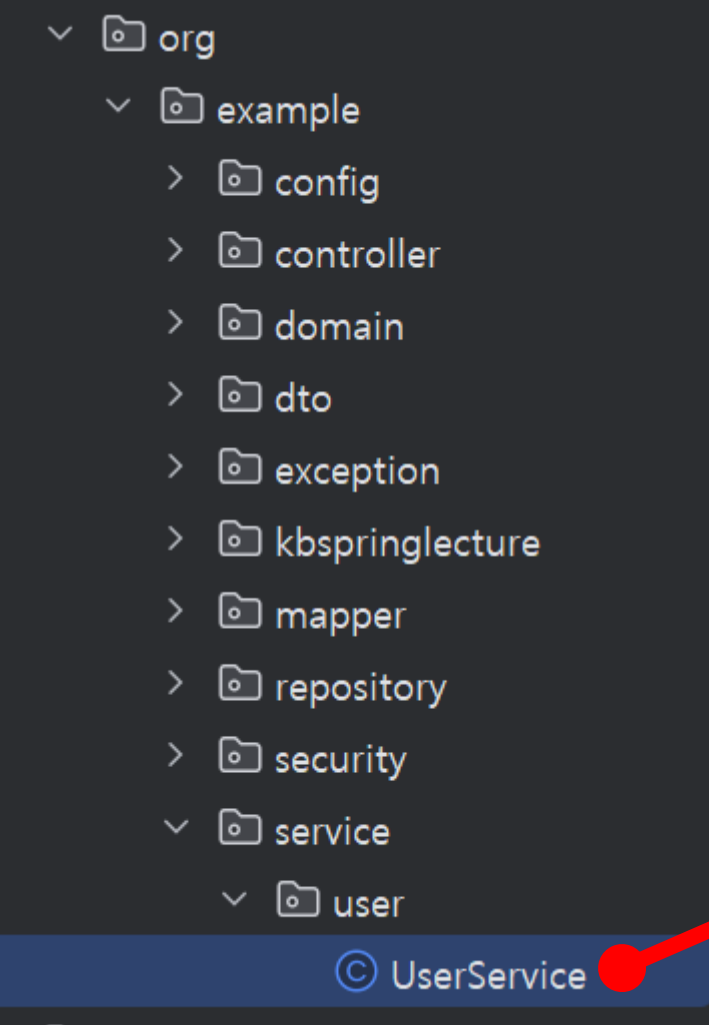
왜 이렇게 구조를 복잡하게 가져가나요!?

관심사에 따라 레이어를 구분하면 나중에 편리해 집니다!

Data 와 통신하는 부분 → Repository
도메인의 비즈니스 로직을 처리 → Service
사용자의 요청에 따른 화면 처리 → Controller


요렇게 나누어 놓으면 해당 관심사의 변경이 필요할 때
하나의 클래스만 변경하면 됩니다!

그란데 말입니다



사용자의 비즈니스 로직을 처리하는 Service 클래스 생성

필요 어노테이션 추가하기!
Service 컴포넌트 이므로 → @Service
UserRepository 를 써야하므로 빈 주입!



@Service 2 usages kdtTetz *

@RequiredArgsConstructor

public class UserService {

private final UserRepository userRepository;



사용자 username 을 통해 사용자를 찾는 메서드

```
@Service 2 usages kdtTetz *
```

```
@RequiredArgsConstructor
```

```
public class UserService {
```

```
    private final UserRepository userRepository;
```

```
    public User findByUsername(String username) { return userRepository.findByUsername(username);
```

```
    public boolean isPasswordValid(User user, String rawPassword) { 1 usage kdtTetz *
```

```
        return rawPassword.equals(user.getPassword());
```

```
}
```

DB 에 있는 사용자의 비밀번호와
입력한 비밀번호가 일치하는지 확인하는 메서드



UserContoroller

만들기



- ▼ org
 - ▼ example
 - > config
 - ▼ controller
 - > board
 - > book
 - > member
 - > post
 - > todo
 - ▼ user

© UserController

클라이언트의 요청과 그에 따른 처리를 담당하는
UserController 만들기!

필요 어노테이션 및 주입 추가하기!
이번에는 MVC 패턴으로 처리할 것이므로
@Controller 어노테이션!

/user 로 모든 주소를 받아서 분배할 예정이므로
RequestMapping 은 /user 로 배정



```
@Controller  👤 kdtTetz *  
@RequiredArgsConstructor  
@RequestMapping(🌐📄 "/user")  
public class UserController {  
    private final UserService userService;  
    private final String context = "/user";  
}
```




로그인 페이지 보여주기

```
@GetMapping(🌐"/login")  
public String loginPage() {  
    return context + "/login";  
}
```

/user/login 요청이 들어오면
/views/user/login.jsp 페이지 보여주기





login.jsp 페이지 만들기

JSP admin.jsp

JSP login.jsp

<https://github.com/xenosign/spring-code-repo/blob/main/jsp/login.jsp>



```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
    <title>Login</title>
</head>
<body>
<%@include file="../header2.jsp"%>
<h1>로그인</h1>
<form action="/user/login" method="post">
    아이디: <input type="text" name="username"><br>
    비밀번호: <input type="password" name="password"><br>
    <input type="submit" value="로그인"/>
</form>
</body>
</html>
```

로그인 요청은

POST 방식
/user/login 에 보냅니다



로그인 처리

컨트롤러

전달 한 파라미터를
@RequestParam 으로 받기

데이터 전송을 위한 Model
로그인 정보 저장을 위한 Session

```
@PostMapping("/login") new *
public String login(@RequestParam("username") String username, @RequestParam String password, Model model, HttpSession session) {
    User user = userService.findByUsername(username);

    if (user == null) {
        model.addAttribute(attributeName: "errMsg", attributeValue: "해당 id의 사용자가 없습니다");
        return context + "/login-failed";
    }

    if (!userService.isPasswordValid(user, password)) {
        model.addAttribute(attributeName: "errMsg", attributeValue: "비밀 번호가 틀립니다");
        return context + "/login-failed";
    }

    model.addAttribute(attributeName: "username", username);
    session.setAttribute(s: "loginUser", user);
    return context + "/login-success";
}
```



```
@PostMapping("/login") new *
public String login(@RequestParam("username") String username) {
    User user = userService.findByUsername(username);

    if (user == null) {
        model.addAttribute(attributeName: "errMsg", attributeValue: "해당 id의 사용자가 없습니다");
        return context + "/login-failed";
    }

    if (!userService.isPasswordValid(user, password)) {
        model.addAttribute(attributeName: "errMsg", attributeValue: "비밀 번호가 틀립니다");
        return context + "/login-failed";
    }

    model.addAttribute(attributeName: "username", username);
    session.setAttribute(s: "loginUser", user);
    return context + "/login-success";
}
```

해당 아이디를 가지는 사용자가 없는 경우
메시지를 모델에 담아 login-failed 페이지로 이동

비밀번호가 틀린 경우 메시지를 모델에 담아
login-failed 페이지로 이동

로그인에 성공한 경우
session 에 loginUser 속성을 만들고
user 정보를 넣어서 로그인 여부를 체크
+ model 에 username 을 담아서 전달!



```
@Component  👤 kdtTetz
public class AuthenticationInterceptor implements HandlerInterceptor {
    @Override  no usages  👤 kdtTetz
    public boolean preHandle(HttpServletRequest request, HttpServletResponse response,
        throws Exception {
        String requestUri = request.getRequestURI();
        System.out.println("요청 URI: " + requestUri);

        HttpSession session = request.getSession();
        if (session.getAttribute(s: "loginUser") == null) {
            System.out.println("로그인 안됨. 리다이렉트 중: /user/login");
            response.sendRedirect(s: "/user/login");
            return false;
        }
        return true;
    }
}
```

인터셉터 코드를 보시면 session 의
loginUser 속성의 값이 있는지를 체크 합니다!

로그인이 되면 해당 값에 객체가 들어가므로
요청이 해당 인터셉터를 지나서 원래의 컨트롤러에 도착



로그인 성공, 실패

페이지

<https://github.com/xenosign/spring-code-repo/blob/main/jsp/login-success.jsp>



```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
    <title>Login</title>
</head>
<body>
<%@include file="../header2.jsp"%>
    <h1>로그인 성공</h1>
    <h2>${username} 님 환영 합니다.</h2>
    <a href="/login/logout">로그아웃</a>
</body>
</html>
```

<https://github.com/xenosign/spring-code-repo/blob/main/jsp/login-failed.jsp>



```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
    <title>Login</title>
</head>
<body>
<%@include file="../header2.jsp"%>
    <h1>로그인 실패</h1>
    <h2>${errMsg}</h2>
    <a href="/user/login">로그인 페이지로 이동</a>
</body>
</html>
```

헤더 메뉴 수정





```
<%@ page contentType="text/html; charset=UTF-8" language="java" %>
```

```
<header>
```

```
    <h3>V1 MyBatis</h3>
```

```
    <a href="/">HOME</a>
```

```
    <a href="/post/v1/show">게시글 목록</a>
```

```
    <a href="/post/v1/404">404</a>
```

```
    <a href="/post/v1/error">error</a>
```

```
    <h3>V1 REST</h3>
```

```
    <a href="/">HOME</a>
```

```
    <a href="/post/v1/rest/show">게시글 목록</a>
```

```
    <h3>V1 JPA</h3>
```

```
    <a href="/">HOME</a>
```

```
    <a href="/post/v2/show">게시글 목록</a>
```

```
    <h3>회원 기능</h3>
```

```
    <a href="/user/register">회원가입</a>
```

```
    <a href="/user/login">로그인</a>
```

```
    <a href="/user/logout">로그아웃</a>
```

```
    <a href="/user/admin">admin</a>
```

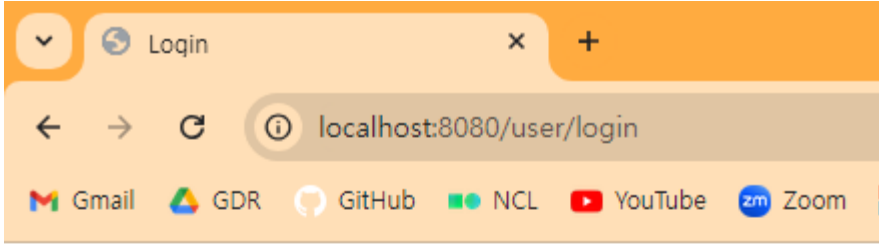
```
</header>
```

자유로운 테스트를 위해서 헤더에
각각의 기능 링크 추가



로그인

테스트



V1 MyBatis

[HOME](#) [게시글 목록](#) [404 error](#)

V1 REST

[HOME](#) [게시글 목록](#)

V1 JPA

[HOME](#) [게시글 목록](#)

회원 기능

[회원가입](#) [로그인](#) [로그아웃](#) [admin](#)

로그인

아이디:

비밀번호:

기존에 등록한 id, password 로
로그인을 테스트!!

로그인 성공

12 님 환영 합니다.

[로그아웃](#)

로그인 실패

비밀 번호가 틀립니다

[로그인 페이지로 이동](#)

로그인 실패

해당 id의 사용자가 없습니다

[로그인 페이지로 이동](#)





이제 로그인이 되었으므로
인터셉터를 지나서 기존의 요청에 도달이 가능합니다!

V1 MyBatis
[HOME 게시글 목록](#) 404 error

V1 REST
[HOME 게시글 목록](#)

V1 JPA
[HOME 게시글 목록](#)

회원 기능
[회원가입](#) [로그인](#) [로그아웃](#) [admin](#)

글 목록

제목 검색 / 내용 검색

ID	Title	Content	Actions
1	첫 번째 게시물	이것은 첫 번째 게시물의 내용입니다. a	<input type="button" value="수정"/> <input type="button" value="삭제"/>
2	두 번째 게시물	이것은 두 번째 게시물의 내용입니다. b	<input type="button" value="수정"/> <input type="button" value="삭제"/>
3	세 번째 게시물	이것은 세 번째 게시물의 내용입니다. c	<input type="button" value="수정"/> <input type="button" value="삭제"/>
4	네 번째 게시물	이것은 네 번째 게시물의 내용입니다. d	<input type="button" value="수정"/> <input type="button" value="삭제"/>

새글 작성하기



로그 아웃 처리!





```
@GetMapping(🌐"/logout")  👤 kdtTetz  
public String logout(HttpSession session) {  
    session.invalidate();  
    return "/user/logout";  
}
```

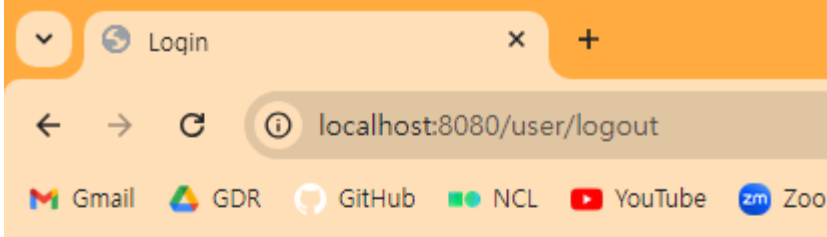
로그 아웃 요청이 들어오면
session 을 만료 처리합니다!

만료 처리 후 logout 페이지로 이동!

<https://github.com/xenosign/spring-code-repo/blob/main/jsp/logout.jsp>



```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
    <title>Login</title>
</head>
<body>
<%@include file="../header2.jsp"%>
<h1>로그 아웃</h1>
<h2>성공적으로 로그 아웃 되었습니다</h2>
<a href="/user/login">로그인 페이지로 이동</a>
</body>
</html>
```



V1 MyBatis

[HOME](#) [게시글 목록](#) [404 error](#)

V1 REST

[HOME](#) [게시글 목록](#)

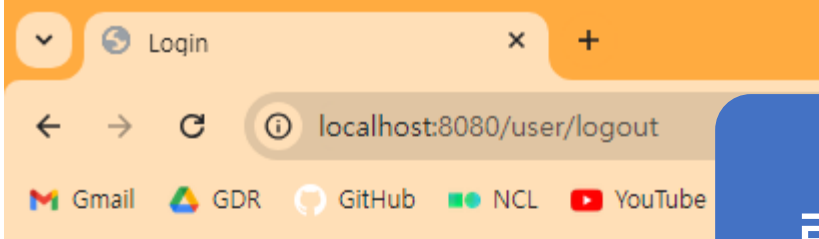
V1 JPA

[HOME](#) [게시글 목록](#)

회원 기능

[회원가입](#) [로그인](#) [로그아웃](#) [admin](#)

로그 아웃 링크를 클릭!



여기서 다시 다른
링크를 클릭 하면!?

V1 MyBatis

[HOME](#) [게시글 목록](#) [404 error](#)

V1 REST

[HOME](#) [게시글 목록](#)

V1 JPA

[HOME](#) [게시글 목록](#)

회원 기능

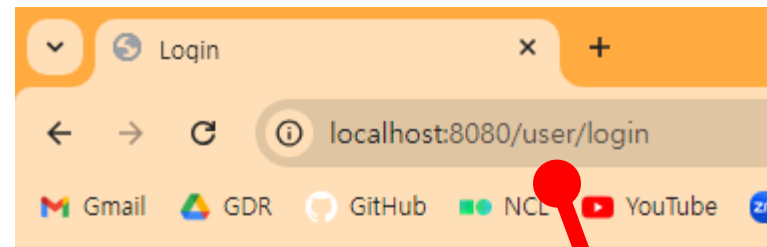
[회원가입](#) [로그인](#) [로그아웃](#) [admin](#)

로그 아웃

성공적으로 로그 아웃 되었습니다

[로그인 페이지로 이동](#)

성공 적으로
로그아웃 처리



V1 MyBatis

[HOME](#) [게시글 목록](#) [404 error](#)

V1 REST

[HOME](#) [게시글 목록](#)

V1 JPA

[HOME](#) [게시글 목록](#)

회원 기능

[회원가입](#) [로그인](#) [로그아웃](#) [admin](#)

로그인

아이디:
비밀번호:

이제는 로그아웃 되어서
다시 로그인 이 필요!



가짜 사회성 부족

여보세요?

아...네? ㅠㅠ...저는 잘 모르겠는데..(소심)



진짜 사회성 부족

“야 그거 가지고 그러냐? 그냥 장난이잖아.
두번 장난치면 뺨이라도 까겠네ㅋㅋㅋ
농담가지고 왜 정색하고 그래.”



아냐!
아직도 노력이
부족해.



진짜 사회성 부족





회원 가입 기능

구현!



회원 가입 페이지

보여주기

```
@GetMapping(🌐✓"/register")  👤 kdtTetz  
public String registerPage() {  
    return context + "/register";  
}
```

/user/register 요청이 들어오면
register.jsp 보여주기!



<https://github.com/xenosign/spring-code-repo/blob/main/jsp/register.jsp>



```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
    <title>Login</title>
</head>
<body>
<%@include file="../header2.jsp"%>
<h1>회원 가입</h1>
<form action="/user/register" method="post">
    아이디: <input type="text" name="username"><br>
    비밀번호: <input type="password" name="password"><br>
    <input type="submit" value="회원 가입"/>
</form>
</body>
</html>
```

회원 가입 요청은

POST 방식
/user/register 에 보냅니다



그란데 말입니다

우리는 총 몇 개의 레이어를 적용시켰나요?!?

Repository → Service → Controller

3개의 레이어를 구성 했습니다!

즉, 3개를 다 고쳐야 합니다

그란데 말입니다

MBC





회원 가입

Repository 처리



```
public User save(User user) {  
    em.persist(user);  
    return user;  
}
```

새로운 사용자를 등록하는 메서드



회원 가입

Service 처리



```
public void save(User user) {  👤 kdtTe  
    user.setRoles("ROLE_MEMBER");  
    userRepository.save(user);  
}
```

전달 받은 정보에
roles 컬럼에 값을 추가하여 저장!



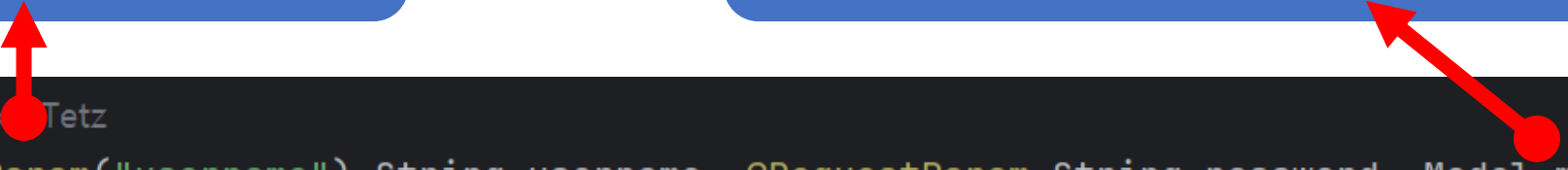
회원 가입

컨트롤러 처리

전달 한 파라미터를
@RequestParam 으로 받기

데이터 전송을 위한 Model

```
@PostMapping("/register")
public String register(@RequestParam("username") String username, @RequestParam String password, Model model) {
    if (username.isEmpty() || password.isEmpty()) {
        model.addAttribute("errMsg", "아이디 또는 비밀번호가 누락 되었습니다");
        return context + "/register-failed";
    }
}
```



@PostMapping(🌐"/register") 👤 kdtTetz

```
public String register(@RequestParam("username") String username, @RequestParam String password, Model model) {  
    if (username.isEmpty() || password.isEmpty()) {  
        model.addAttribute(attributeName: "errMsg", attributeValue: "아이디 또는 비밀번호가 누락 되었습니다");  
        return context + "/register-failed";  
    }  
  
    User user = userService.findByUsername(username);  
    if (user != null) {  
        model.addAttribute(attributeName: "errMsg", attributeValue: "동일한 ID 를 가지는 사용자가 존재합니다");  
        return context + "/register-failed";  
    }  
  
    User newUser = new User();  
    newUser.setUsername(username);  
    newUser.setPassword(password);  
    userService.save(newUser);  
  
    model.addAttribute(attributeName: "username", username);  
  
    return context + "/register-success";  
}
```

입력 값이 제대로 전달 안된 경우
가입 실패 페이지로 이동!

해당 username 을 가지는 회원이
존재하는 경우 가입 실패 페이지로 이동!

@PostMapping("/register") kdtTetz

```
public String register(@RequestParam("username") String username, @RequestParam String password, Model model) {  
    if (username.isEmpty() || password.isEmpty()) {  
        model.addAttribute("errMsg", "아이디 또는 비밀번호가 누락 되었습니다");  
        return context + "/register-failed";  
    }  
}
```

```
User user = userService.findByUsername(username);  
if (user != null) {  
    model.addAttribute("errMsg", "동일한 ID 를 가지는 사용자가 존재합니다");  
    return context + "/register-failed";  
}
```

```
User newUser = new User();  
newUser.setUsername(username);  
newUser.setPassword(password);  
userService.save(newUser);
```

```
model.addAttribute("username", username);
```

```
return context + "/register-success";
```

```
}
```

전달 받은 회원 정보를 userService 에
전달하여 회원 가입 진행

가입한 회원 정보를 Model 에 저장



회원 가입

성공, 실패 페이지

<https://github.com/xenosign/spring-code-repo/blob/main/jsp/register-success.jsp>



```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
    <title>Login</title>
</head>
<body>
<%@include file="../header2.jsp"%>
    <h1>회원 가입 성공</h1>
    <h2>${username} 님 환영 합니다.</h2>
    <a href="/user/login">로그인 페이지로 이동</a>
</body>
</html>
```

<https://github.com/xenosign/spring-code-repo/blob/main/jsp/register-failed.jsp>



```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
    <title>Login</title>
</head>
<body>
<%@include file="../header2.jsp"%>
    <h1>회원 가입 실패</h1>
    <h2>${errMsg}</h2>
    <a href="/user/register">로그인 페이지로 이동</a>
</body>
</html>
```



회원 가입

테스트



▼

Login

×

+

←

→

↻

localhost:8080/user/register

Gmail

GDR

GitHub

NCL

YouTube

V1 MyBatis

[HOME](#) [게시글 목록](#) [404 error](#)

V1 REST

[HOME](#) [게시글 목록](#)

V1 JPA

[HOME](#) [게시글 목록](#)

회원 기능

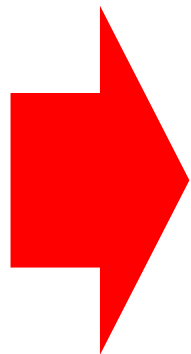
[회원가입](#) [로그인](#) [로그아웃](#) [admin](#)

회원 가입

아이디:

비밀번호:

회원 가입



▼

Login

×

+

←

→

↻

localhost:8080/user/register

Gmail

GDR

GitHub

NCL

YouTube

V1 MyBatis

[HOME](#) [게시글 목록](#) [404 error](#)

V1 REST

[HOME](#) [게시글 목록](#)

V1 JPA

[HOME](#) [게시글 목록](#)

회원 기능

[회원가입](#) [로그인](#) [로그아웃](#) [admin](#)

회원 가입 성공

33 님 환영 합니다.

[로그인 페이지로 이동](#)

회원 가입 실패

동일한 ID 를 가지는 사용자가 존재합니다

[로그인 페이지로 이동](#)

Result Grid				
	id	username	password	roles
▶	1	12	12	ROLE_MEMBER
	2	tetz	12	ROLE_MEMBER
	3	siwan	12	ROLE_MEMBER
	5	33	33	ROLE_MEMBER
⊙	NULL	NULL	NULL	NULL



Login

localhost:8080/user/login

GmailGDRGitHubNCLYouTube

V1 MyBatis

[HOME 게시글 목록 404 error](#)

V1 REST

[HOME 게시글 목록](#)

V1 JPA

[HOME 게시글 목록](#)

회원 기능

[회원가입](#) [로그인](#) [로그아웃](#) [admin](#)

로그인 성공

33 님 환영 합니다.

[로그아웃](#)





그런데 말입니다

DB 에 사용자 비번을 그대로 저장하는게 맞을까요?

Result Grid					Filter Rows:
	id	username	password	roles	
▶	1	12	12	ROLE_MEMBER	
	2	tetz	12	ROLE_MEMBER	
	3	siwan	12	ROLE_MEMBER	
	5	33	33	ROLE_MEMBER	
●	NULL	NULL	NULL	NULL	

그런데 말입니다



만약 해커한테 DB 해킹을 당한다면!?

그런데 여러분이 담당하고 있는 서비스가
쇼핑몰이라면!?

암호화 도입





build.gradle 에

암호화 라이브러리 추가

<https://github.com/xenosign/spring-code-repo/blob/main/gradle/crypto.gradle>



```
// 암호화
```

```
implementation 'org.springframework.security:spring-security-crypto'
```





UserConfig 에 암호화 라이브러리 Bean 등록



```
@Configuration  👤 kdtTetz *  
public class UserConfig implements WebMvcConfigurer {  
    @Bean  👤 kdtTetz  
    public BCryptPasswordEncoder passwordEncoder() {  
        return new BCryptPasswordEncoder();  
    }  
}
```

매번 암호화 도구를 생성해서 쓰면
낭비이므로 Bean 으로 등록하고
필요할 때 주입을 받아서 사용!



회원 가입 시에 암호화 기능 추가



그란데 말입니다

드디어 말입니다 레이어를 나눈 효과를 보게 됩니다!

사용자를 암호화해서 저장하는 부분은 어떤 부분만
고치면 될까요!?

해당 부분은 Service 파트에서
Repository 로 유저 정보를 보낼 때
비밀번호 부분만 암호화를 해서 보내면 됩니다!!

@Service 2 usages kdtTetz *

@RequiredArgsConstructor

public class UserService {

private final UserRepository userRepository;


private final BCryptPasswordEncoder passwordEncoder;

Bean 으로 등록된 암호화 도구를
주입받기





```
public void save(User user) {  👤 kdtTetz
    String encodedPassword = passwordEncoder.encode(user.getPassword());
    user.setPassword(encodedPassword);
    user.setRoles("ROLE_MEMBER");
    userRepository.save(user);
}
```



회원 정보를 저장할 때
전달 받은 비밀번호를 암호화 도구를 사용하여
암호화 한 뒤, 저장하는 형태로 변경!

```
public boolean isValid(User user, String rawPassword) {  
    return passwordEncoder.matches(rawPassword, user.getPassword());  
}
```



로그인을 처리할 때에도 이제
입력 받은 비밀번호를 암호화 한 다음
DB 에 저장된 암호화 된 비밀번호와 비교하여
로그인 처리를 해야합니다!

암호화 도구의 matches 함수는
첫번째 인자의 문자열을 자동으로 암호화 한 뒤
두번째 인자의 암호화 문자열과 비교하여
결과를 리턴 합니다!



암호화 된 회원 가입 테스트

▼

Login

×

+

←

→

↻

localhost:8080/user/register

GmailGDRGitHubNCLYouTubezm

V1 MyBatis

[HOME](#) [게시글 목록](#) [404 error](#)

V1 REST

[HOME](#) [게시글 목록](#)

V1 JPA

[HOME](#) [게시글 목록](#)

회원 기능

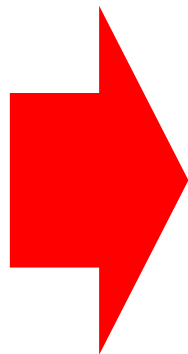
[회원가입](#) [로그인](#) [로그아웃](#) [admin](#)

회원 가입

아이디:

비밀번호:

회원 가입



▼

Login

×

+

←

→

↻

localhost:8080/user/register

GmailGDRGitHubNCLYouTube

V1 MyBatis

[HOME](#) [게시글 목록](#) [404 error](#)

V1 REST

[HOME](#) [게시글 목록](#)

V1 JPA

[HOME](#) [게시글 목록](#)

회원 기능

[회원가입](#) [로그인](#) [로그아웃](#) [admin](#)

회원 가입 성공


55 님 환영 합니다.


[로그인 페이지로 이동](#)






Result Grid







Filter Rows:

Edit:







Export/Import:

	id	username	password	roles
▶	1	12	12	ROLE_MEMBER
	2	tetz	12	ROLE_MEMBER
	2	siwan	12	ROLE_MEMBER
	5	33	33	ROLE_MEMBER
	6	55	\$2a\$10\$skUsfISNPsolN1uSYCFHA/O4NGvtQe7zJ...	ROLE_MEMBER
⊙	NULL	NULL	NULL	NULL

비밀번호가 암호화 되어서
저장 된 것을 확인 가능

▼

Login

×

+

←

→

↻

localhost:8080/user/login

Gmail

GDR

GitHub

NCL

YouTube

V1 MyBatis

[HOME](#) [게시글 목록](#) [404 error](#)

V1 REST

[HOME](#) [게시글 목록](#)

V1 JPA

[HOME](#) [게시글 목록](#)

회원 기능

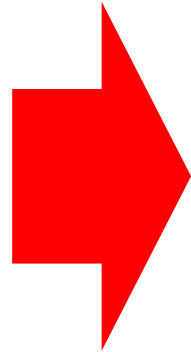
[회원가입](#) [로그인](#) [로그아웃](#) [admin](#)

로그인

아이디:

비밀번호:

로그인



▼

Login

×

+

←

→

↻

localhost:8080/user/login

Gmail

GDR

GitHub

NCL

YouTube

V1 MyBatis

[HOME](#) [게시글 목록](#) [404 error](#)

V1 REST

[HOME](#) [게시글 목록](#)

V1 JPA

[HOME](#) [게시글 목록](#)

회원 기능

[회원가입](#) [로그인](#) [로그아웃](#) [admin](#)

로그인 성공

55 님 환영 합니다.

[로그아웃](#)

▼

Login

×

+

←

→

↻

localhost:8080/user/login

Gmail

GDR

GitHub

NCL

YouTube

V1 MyBatis

[HOME](#) [게시글 목록](#) [404 error](#)

V1 REST

[HOME](#) [게시글 목록](#)

V1 JPA

[HOME](#) [게시글 목록](#)

회원 기능

[회원가입](#) [로그인](#) [로그아웃](#) [admin](#)

로그인 실패

비밀 번호가 틀립니다

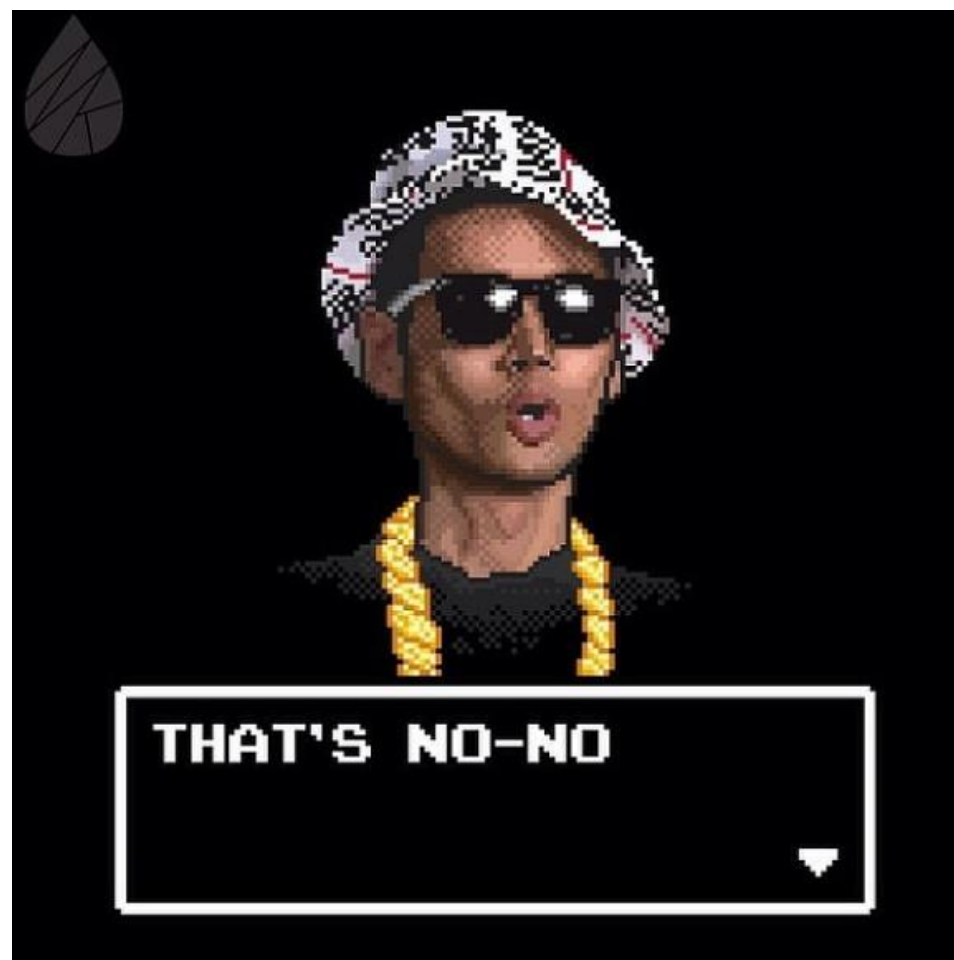
[로그인 페이지로 이동](#)

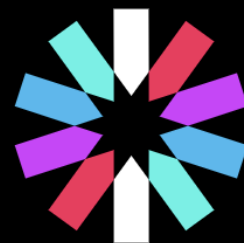


그란데 말입니다

그란데 말입니다!

이렇게 중요한 보안 관련 코드를
우리가 직접 짜는게 맞을까요!?





JWT

the face reader
lee jung jae, dipsy