인증 / 인가

양시영

AhnLab

웹서비스개발팀

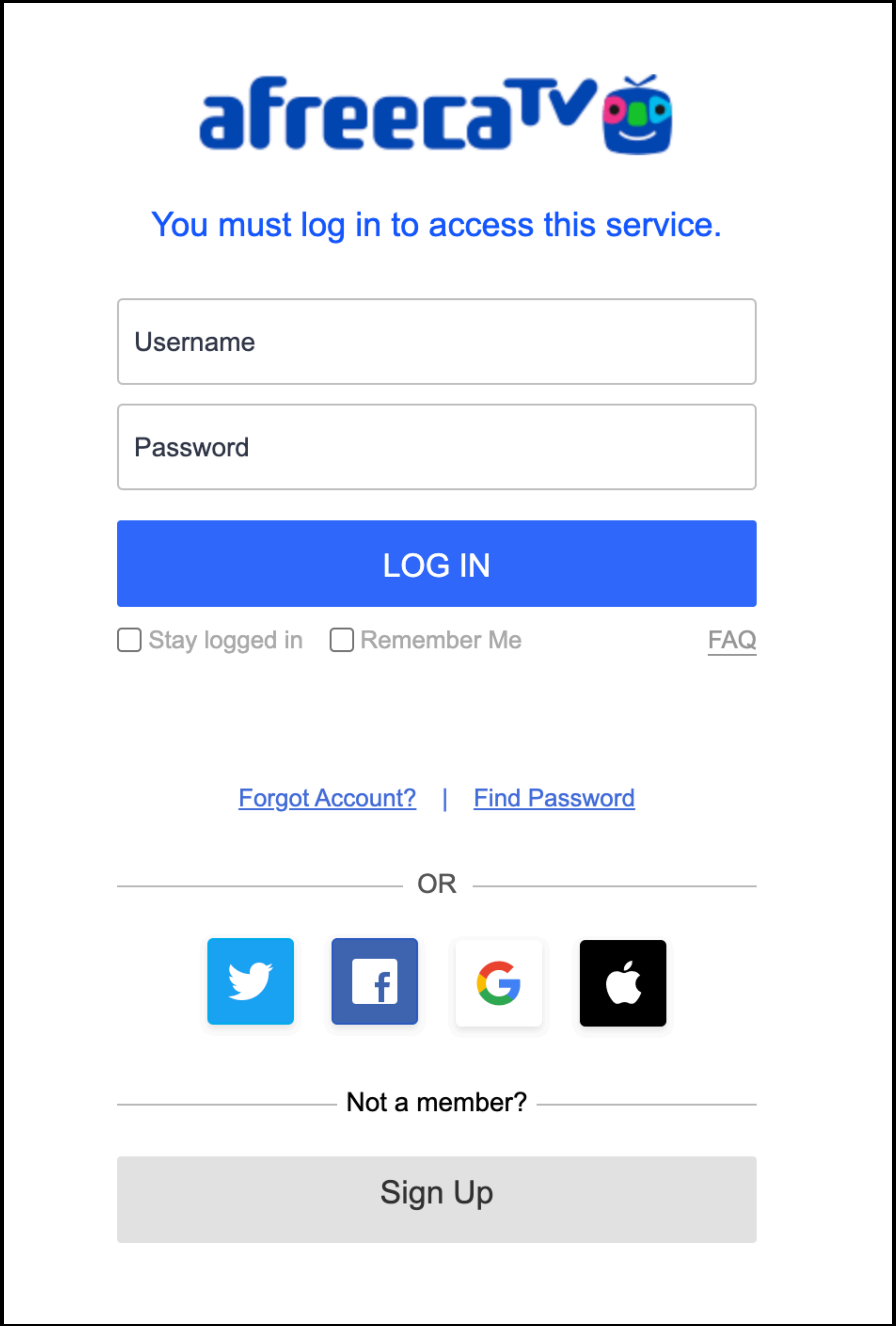
NBP NAVER BUSINESS PLATFORM

Security Dev
(정규직 광탈)

멀티플랫폼팀
멤버십

인증

너 누구세요?


You must log in to access this service.

Username

Password

LOG IN

☐ Stay logged in ☐ Remember Me FAQ

Forgot Account? | Find Password

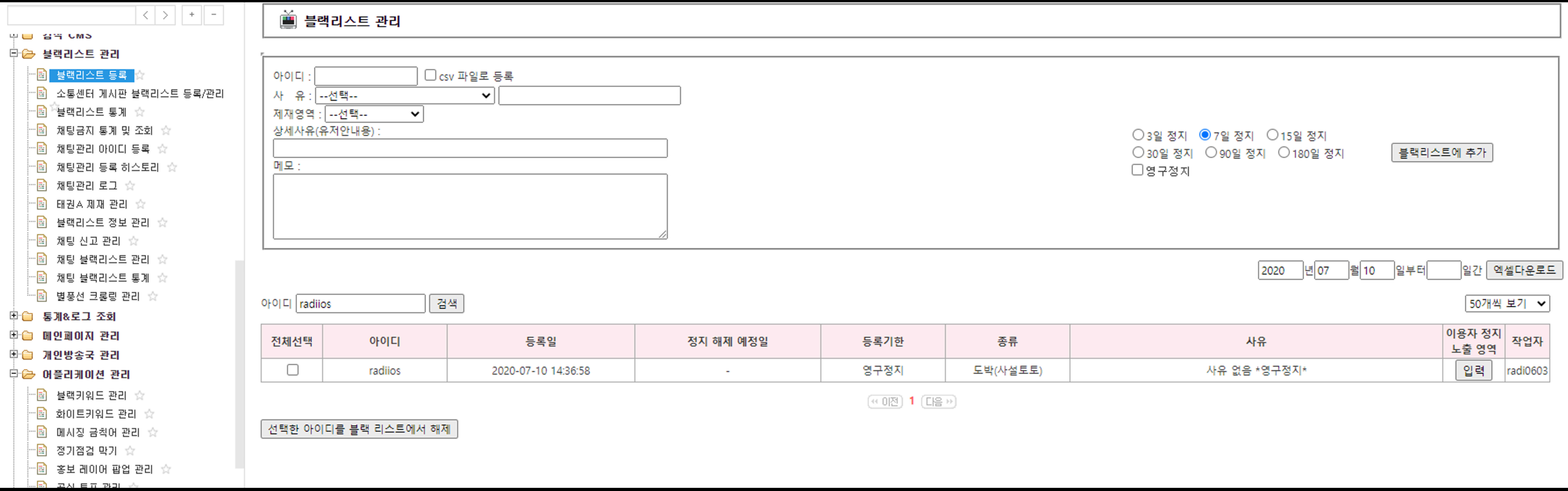
OR

Not a member?

Sign Up

인가

너는 어떤 권한이 있어요?



아이디 : csv 파일로 등록
사 유 : --선택--
제재영역 : --선택--
상세사유(유저안내용) :
메모 :

3일 정지 [✓ 7일 정지] 15일 정지
30일 정지 90일 정지 180일 정지
영구정지

블랙리스트에 추가

2020 년 07 월 10 일부터 일간 엑셀다운로드

아이디 radiios 검색

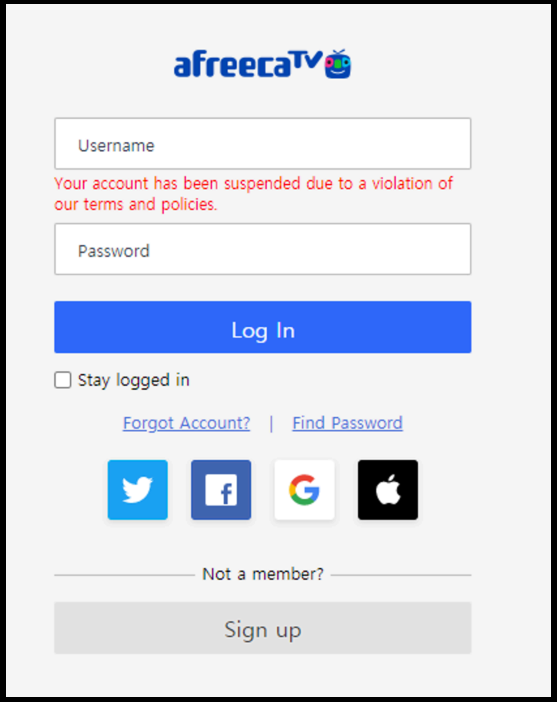
50개씩 보기

전체선택	아이디	등록일	정지 해제 예정일	등록기한	종류	사유	이용자 정지 노출 영역	작업자
☐	radiios	2020-07-10 14:36:58	-	영구정지	도박(사설토토)	사유 없음 *영구정지*	입력	radi0603

« 이전 1 다음 »

선택한 아이디를 블랙 리스트에서 해제

Admin(관리자)



Username

Your account has been suspended due to a violation of our terms and policies.

Password

Log In

☐ Stay logged in

Forgot Account? | Find Password

Not a member?

Sign up

User(사용자)

끝

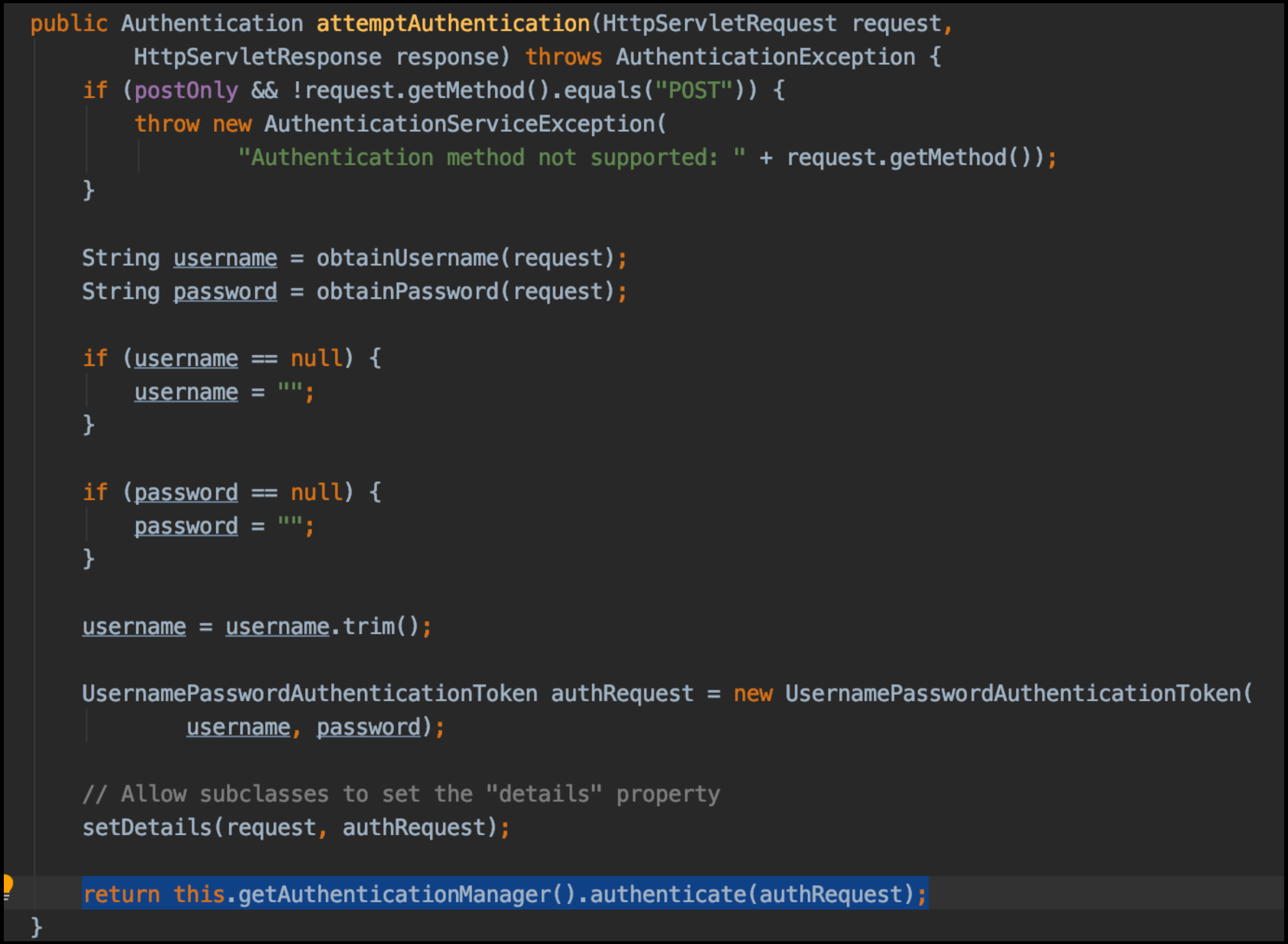
은 뻥

인증 / 인가와 관련된
Spring Security 기본 설정

1. WebAsyncManagerIntergrationFilter
2. SecurityContextPersistenceFilter
3. HeaderWriterFilter
4. CsrfFilter
5. LogoutFilter
6. UsernamePasswordAuthenticationFilter
7. DefaultLoginPageGeneratingFilter
8. DefaultLogoutPageGeneratingFilter
9. BasicAuthenticationFilter
10. RequestCacheAwareFtiler
11. SecurityContextHolderAwareReqeustFilter
12. AnonymouseAuthenticationFilter
13. SessionManagementFilter
14. ExeptionTranslationFilter
15. FilterSecurityInterceptor

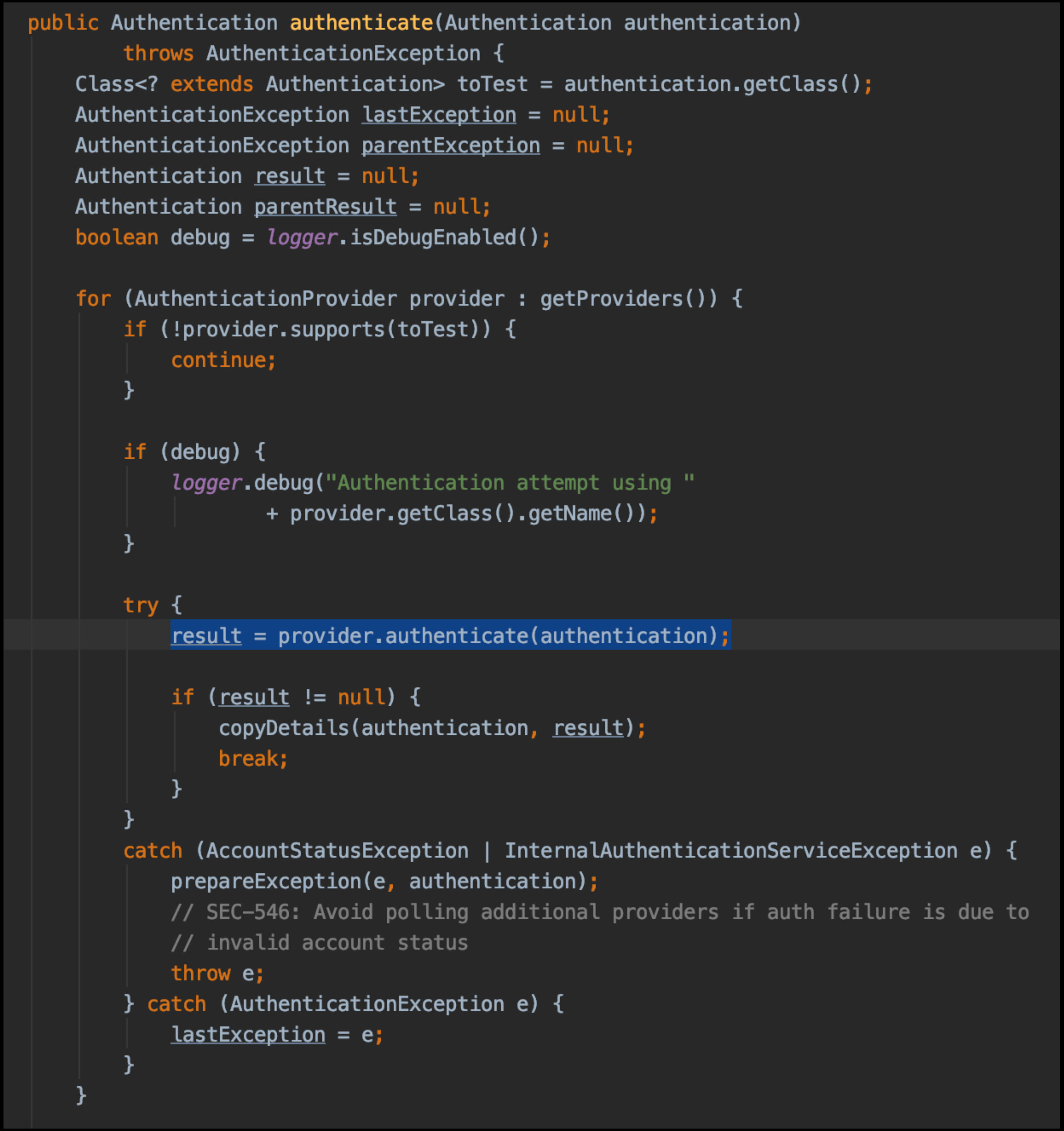
UsernamePasswordAuthenticationFilter

UsernamePasswordAuthenticationFilter



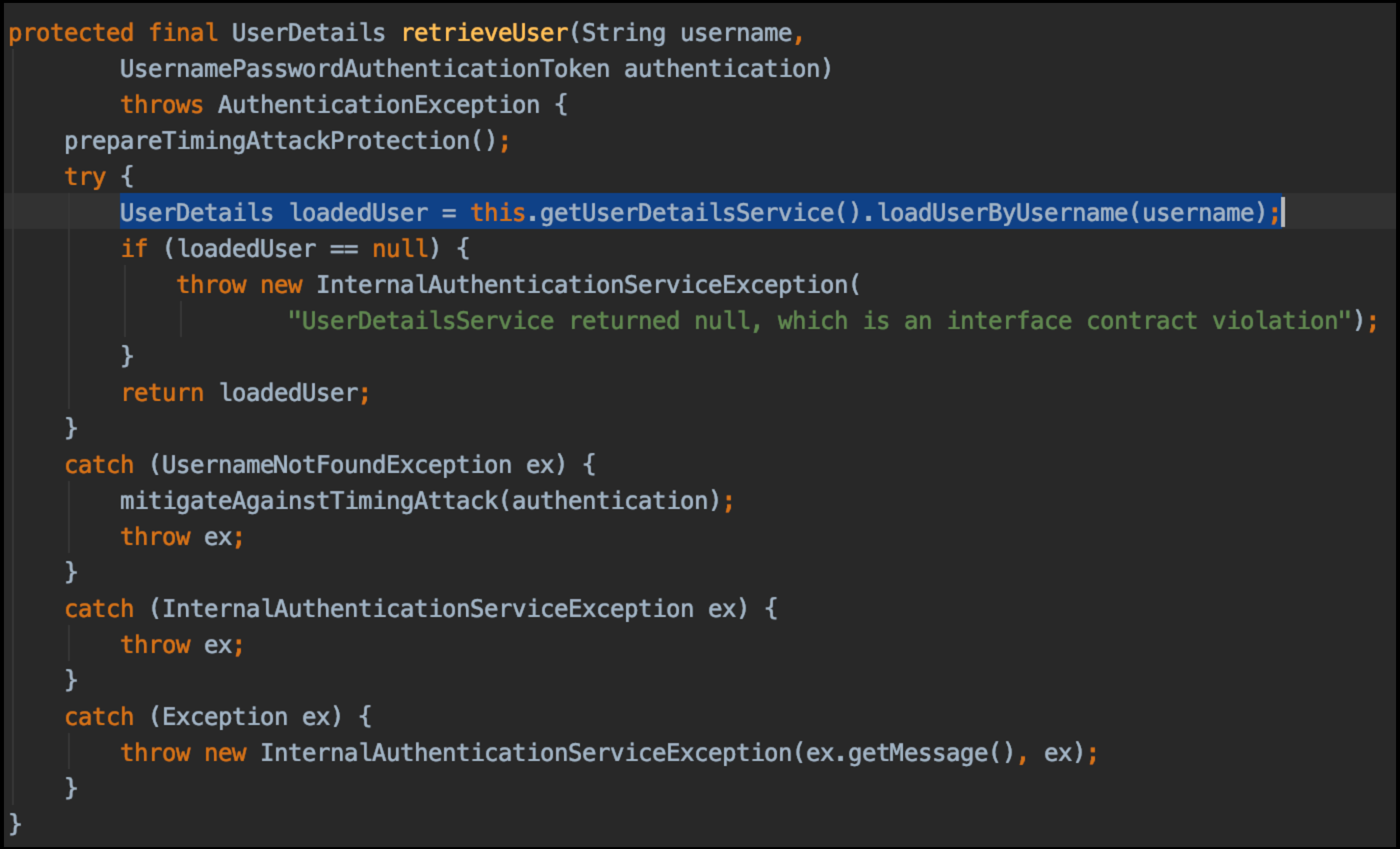
```java
public Authentication attemptAuthentication(HttpServletRequest request,
        HttpServletResponse response) throws AuthenticationException {
    if (postOnly && !request.getMethod().equals("POST")) {
        throw new AuthenticationServiceException(
                "Authentication method not supported: " + request.getMethod());
    }

    String username = obtainUsername(request);
    String password = obtainPassword(request);

    if (username == null) {
        username = "";
    }

    if (password == null) {
        password = "";
    }

    username = username.trim();

    UsernamePasswordAuthenticationToken authRequest = new UsernamePasswordAuthenticationToken(
            username, password);

    // Allow subclasses to set the "details" property
    setDetails(request, authRequest);

    return this.getAuthenticationManager().authenticate(authRequest);
}
```

UsernamePasswordAuthenticationFilter -> ProviderManager(implements AuthenticationManager)



```java
public Authentication authenticate(Authentication authentication)
        throws AuthenticationException {
    Class<? extends Authentication> toTest = authentication.getClass();
    AuthenticationException lastException = null;
    AuthenticationException parentException = null;
    Authentication result = null;
    Authentication parentResult = null;
    boolean debug = logger.isDebugEnabled();

    for (AuthenticationProvider provider : getProviders()) {
        if (!provider.supports(toTest)) {
            continue;
        }

        if (debug) {
            logger.debug("Authentication attempt using "
                    + provider.getClass().getName());
        }

        try {
            result = provider.authenticate(authentication);

            if (result != null) {
                copyDetails(authentication, result);
                break;
            }
        }
        catch (AccountStatusException | InternalAuthenticationServiceException e) {
            prepareException(e, authentication);
            // SEC-546: Avoid polling additional providers if auth failure is due to
            // invalid account status
            throw e;
        } catch (AuthenticationException e) {
            lastException = e;
        }
    }
```

UsernamePasswordAuthenticationFilter
-> ProviderManager(implements AuthenticationManager)
-> DaoAuthenticationProvider



```java
protected final UserDetails retrieveUser(String username,
        UsernamePasswordAuthenticationToken authentication)
        throws AuthenticationException {
    prepareTimingAttackProtection();
    try {
        UserDetails loadedUser = this.getUserDetailsService().loadUserByUsername(username);
        if (loadedUser == null) {
            throw new InternalAuthenticationServiceException(
                    "UserDetailsService returned null, which is an interface contract violation");
        }
        return loadedUser;
    }
    catch (UsernameNotFoundException ex) {
        mitigateAgainstTimingAttack(authentication);
        throw ex;
    }
    catch (InternalAuthenticationServiceException ex) {
        throw ex;
    }
    catch (Exception ex) {
        throw new InternalAuthenticationServiceException(ex.getMessage(), ex);
    }
}
```

인증은 내가 누구인지

인가는 내가 어떤 권한이 있는지

로그인!
서비스의 기초이자

서비스 보안의 핵심입니다.