Spring Security ile JWT Entegrasyonu

DevTech Community Etkinliği
Batuhan Düzgün - Mart 2021

Yol Haritası

1- Oturum Yönetimi

2- Spring Security Mimarisi

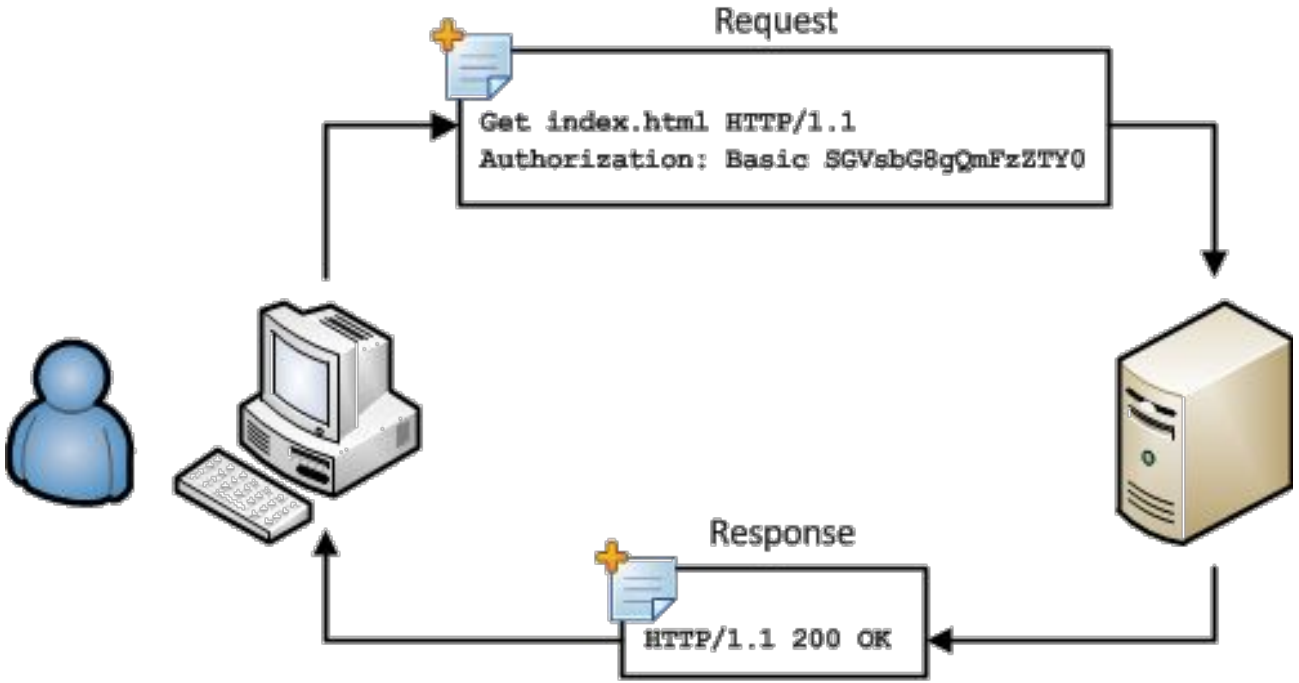
3- Json Web Token (JWT)

4- IDOR Atağına Önlem

5- Stateless REST API Tasarımı

1- Oturum Yönetimi

Oturum Yönetimi



Oturum Yönetimi

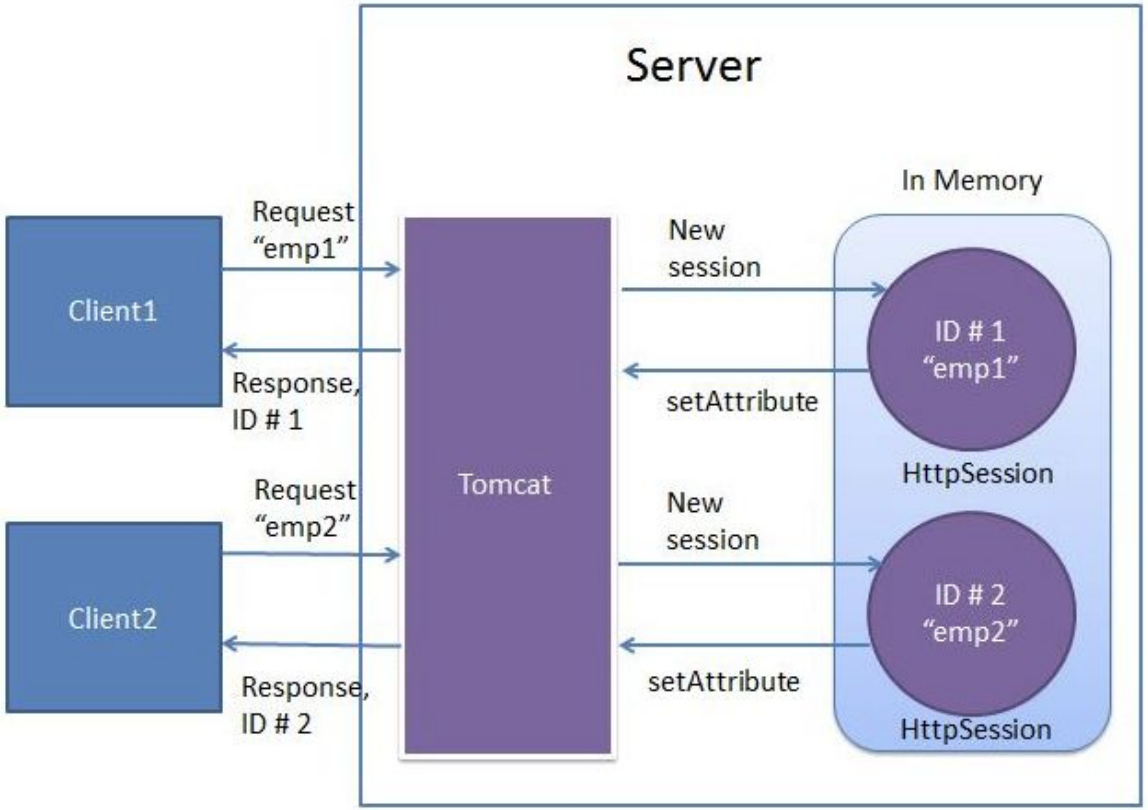
- Web uygulamaları çoğunlukla HTTP uygulama protokolü üzerinden haberleşmektedir.

- Client-Server Mimarisi

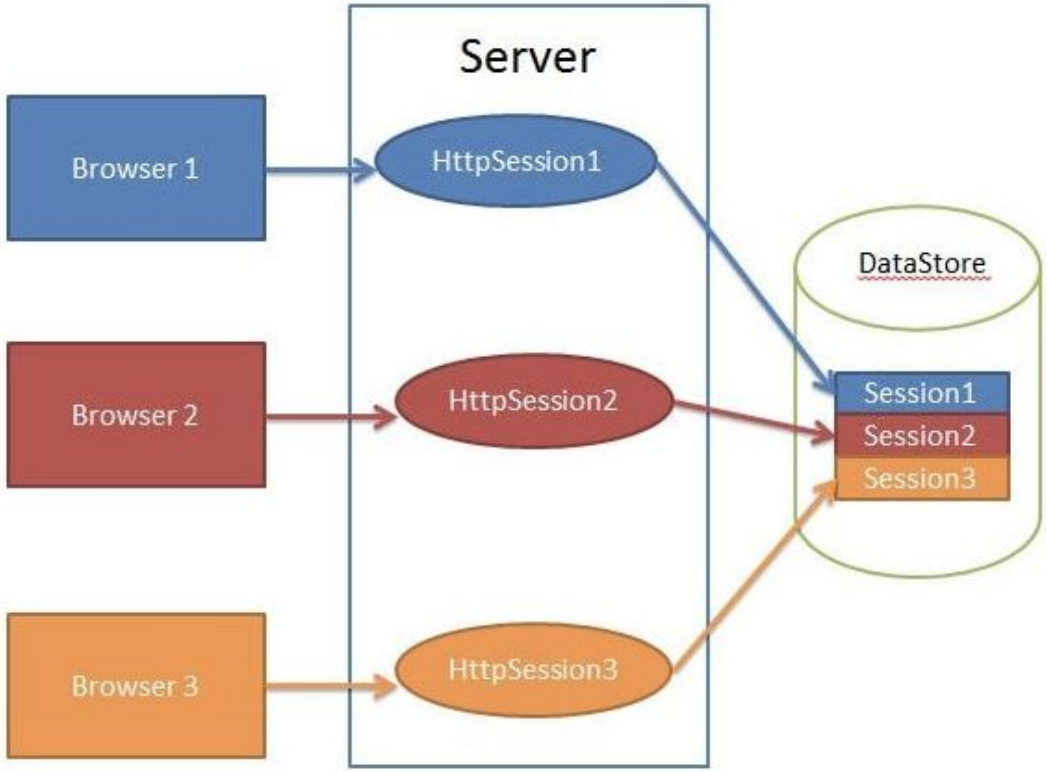
- HTTP Stateless bir protokol!

- Her HTTP isteği ve cevabı birbirinden bağımsız.

Oturum Yönetimi



Oturum Yönetimi



Oturum Yönetimi

Oturum Tasarımı

- REST API projelerinde stateless oturum yaklaşımı tercih edilir.

- Web uygulamalarında ise oturum oluşturulup kullanılması tercih edilir.

Oturum Yönetimi

Oturum İletimi

- HTTP Header üzerinden yapılabilir.

- Cookie'ler üzerinden yapılabilir.

- Güvenlik açısından Localstorage'da **saklamamak** lazım.

Oturum Yönetimi

Oturum Güvenliği (Cookie-Based)

- **httpOnly**: eğer true değere sahip olursa tarayıcı tarafında JS ile cookie'ler erişilemez.

- **secure**: eğer true ise cookie'ler ancak HTTPS bağlantısı üzerinden aktarılabilir.

Oturum Yönetimi

Spring Security ile Oturum Yönetimi

always: eğer hali hazırda oturum yoksa her zaman yenisini oluşturur.

ifRequired: sadece gerektiği durumda yeni oturum oluşturur.

never: kesinlikle yeni bir oturum oluşturma var olanı kullanır.

stateless: yeni oturum oluşturulmaz, var olan da kullanılmaz.

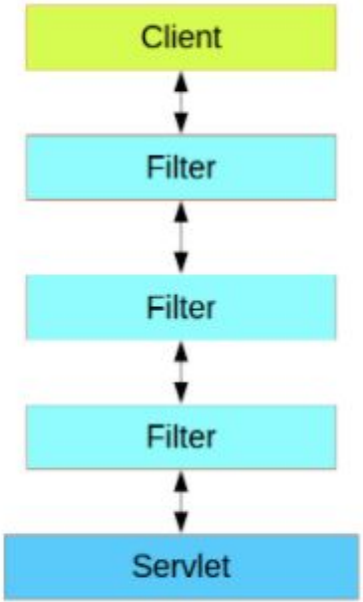
2- Spring Security Mimarisi

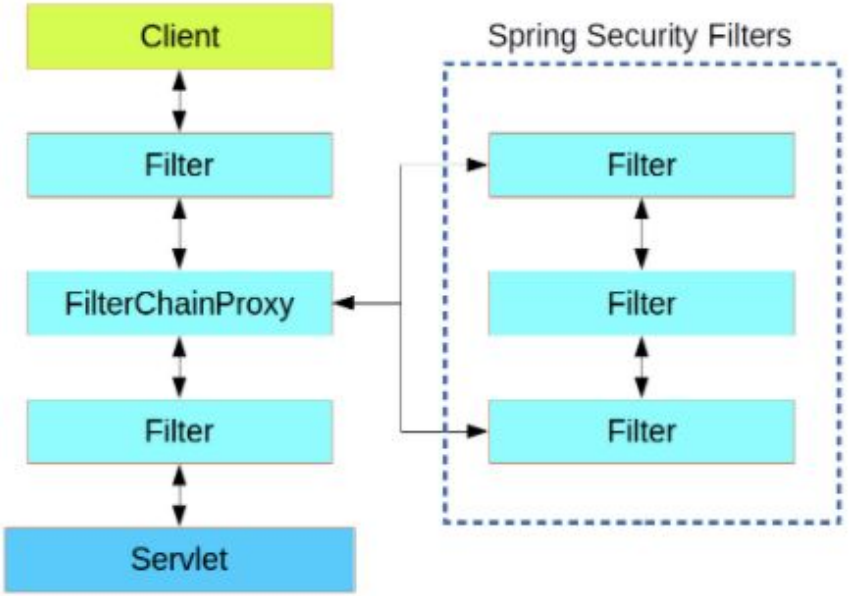
Spring Security Mimarisi

- Authentication
- Authorization

2.1- Spring Security Filtre Sistemi

Spring Security Mimarisi



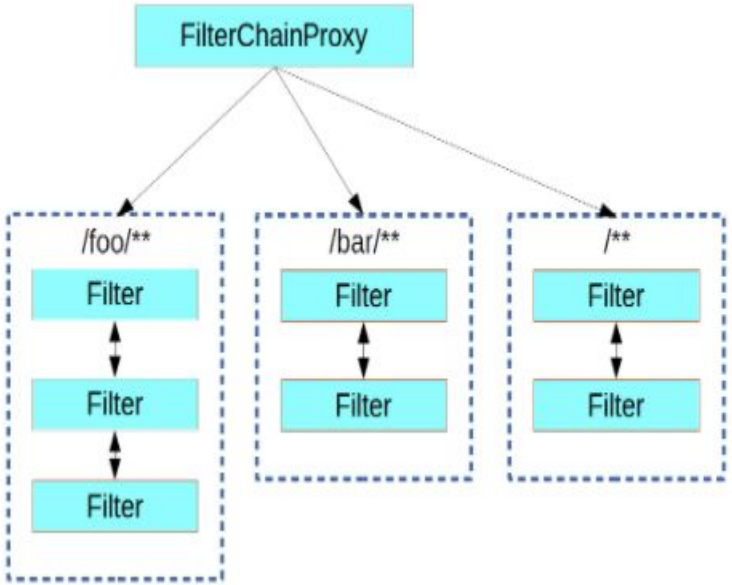


Spring Security Mimarisi

- Spring Security, "**FilterChainProxy**" isminde filtre sınıfı ile web uygulamasında yer alır.

- Bu filtre Spring @Bean olarak **ApplicationContext** içinde yer alır.

- Otomatik olarak tüm HTTP istekleri bu filtreden geçer.

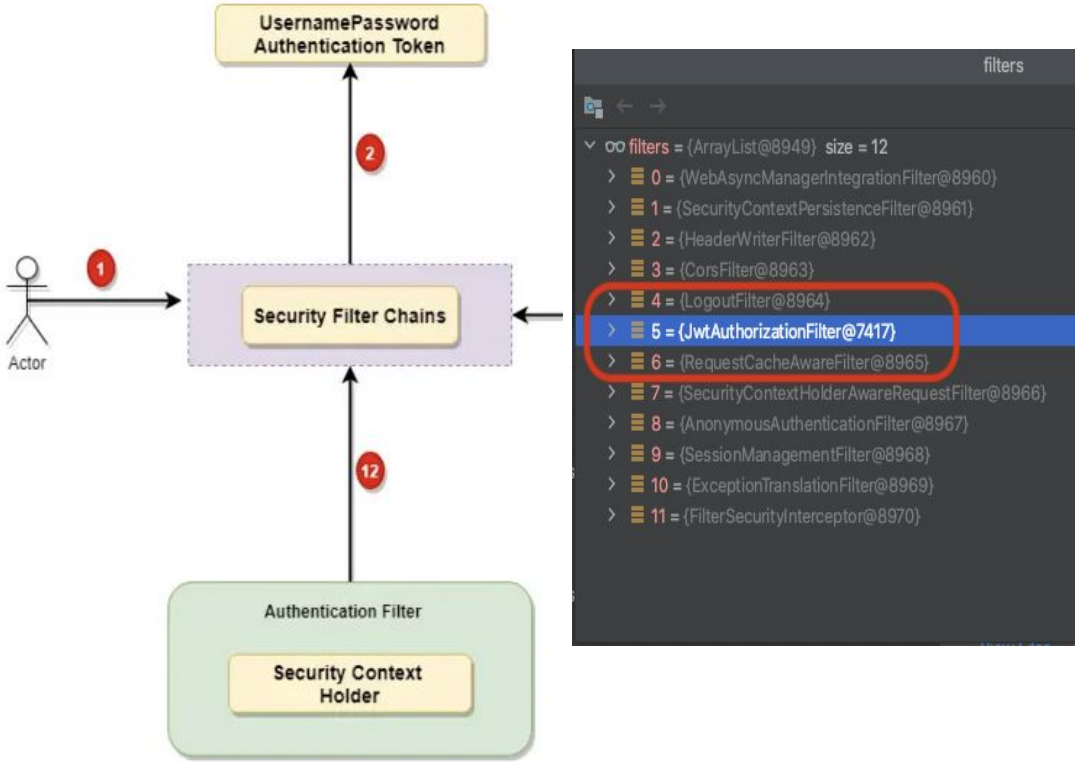


Spring Security Mimarisi

- **FilterChainProxy** filtresine bağlı sıralı şekilde bir çok alt filtre bağlıdır.

- Biz de bu filtre zincirine yeni bir tane ekleyebiliriz.

- **JWT** filtresini de bu zincire dahil edeceğiz.



2.2- Spring Security Auth. Akışı

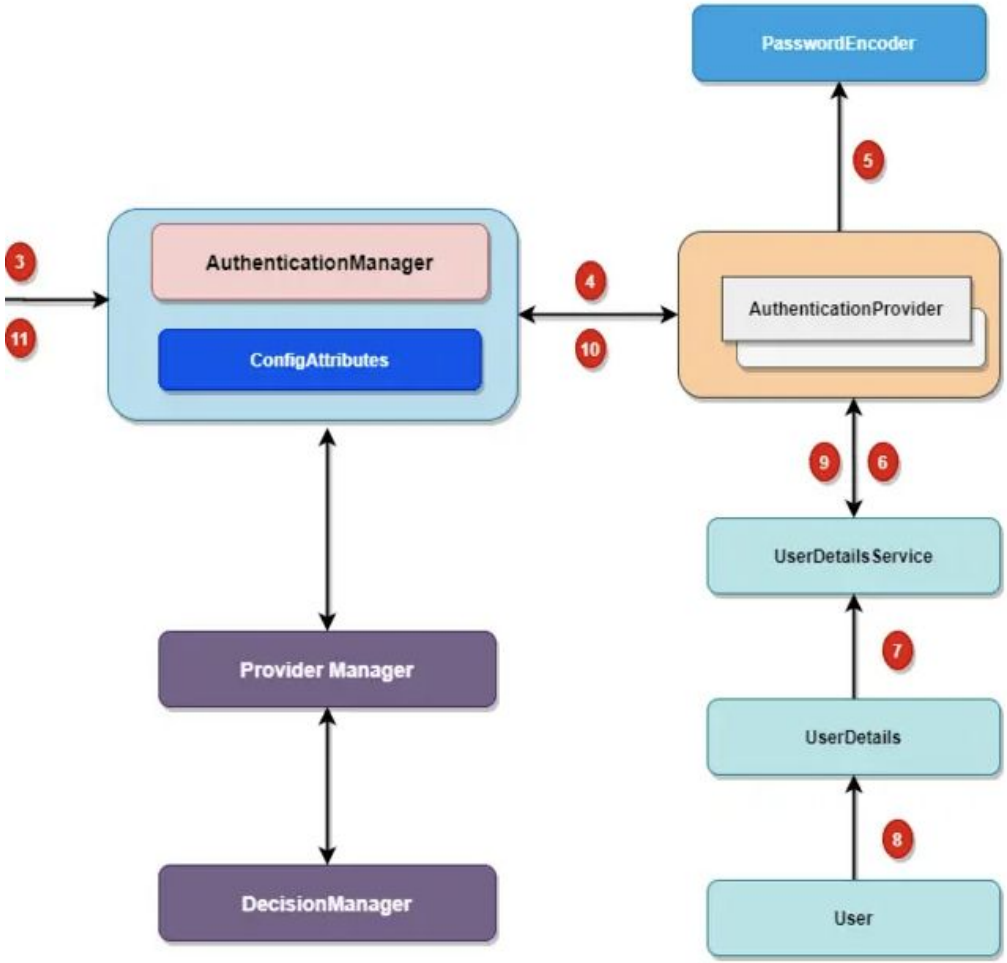
Spring Security Mimarisi

1- AuthenticationManager

2- AuthenticationProvider

3- UserDetailsService

4- SecurityContext Authentication



AuthenticationManager

- Spring Security tarafında authentication işleminin nasıl yapılacağını tanımlar.

- **AuthenticationManagerBuilder** ile JDBC, In-Memory, LDAP yöntemlerini entegre edilebilir.

- Birden fazla "**AuthenticationProvider**" içinde barındırabilir.

```
@Override
protected void configure(AuthenticationManagerBuilder auth) {

    UserAuthenticationProvider authenticationProvider = new UserAuthenticationProvider();
    authenticationProvider.setPasswordEncoder(NoOpPasswordEncoder.getInstance());
    authenticationProvider.setUserDetailsService(authenticationUserDetailsService);
    auth.authenticationProvider(authenticationProvider);
}
```

AuthenticationProvider

- Gelen istekteki bilgileri işleyip belirtilen authentication yönteminin çalıştırılmasını sağlar.

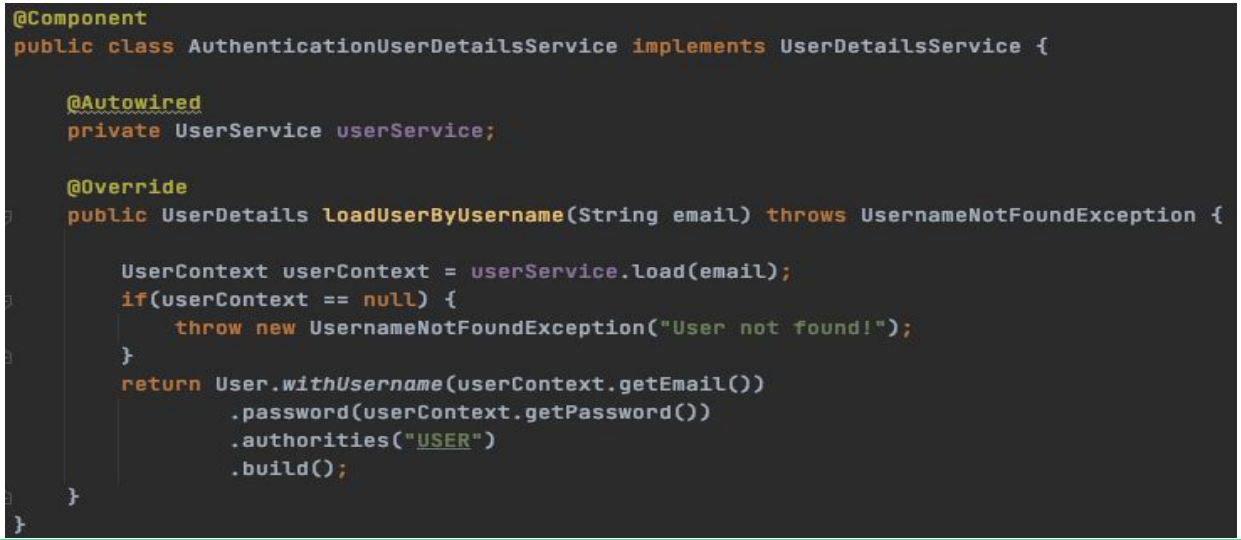
- **UserDetailsService**, **PasswordEncoder** gibi bağımlılıklarla çalışır.

```
public class UserAuthenticationProvider extends DaoAuthenticationProvider {

    @Override
    protected Authentication createSuccessAuthentication(Object principal, Authentication authentication, UserDetails user) {

        return new UsernamePasswordAuthenticationToken(principal, user.getPassword(), user.getAuthorities());
    }
}
```

UserDetailsService

- **AuthenticationProvider** tarafından kullanılır.

- Gelen bilgiye göre bir veri kaynağından sorgulama yapar. Örneğin kullanıcı veri tabanında kayıtlı mı?



```java
@Component
public class AuthenticationUserDetailsService implements UserDetailsService {

    @Autowired
    private UserService userService;

    @Override
    public UserDetails loadUserByUsername(String email) throws UsernameNotFoundException {

        UserContext userContext = userService.load(email);
        if(userContext == null) {
            throw new UsernameNotFoundException("User not found!");
        }
        return User.withUsername(userContext.getEmail())
                .password(userContext.getPassword())
                .authorities("USER")
                .build();
    }
}
```

SecurityContext Authentication

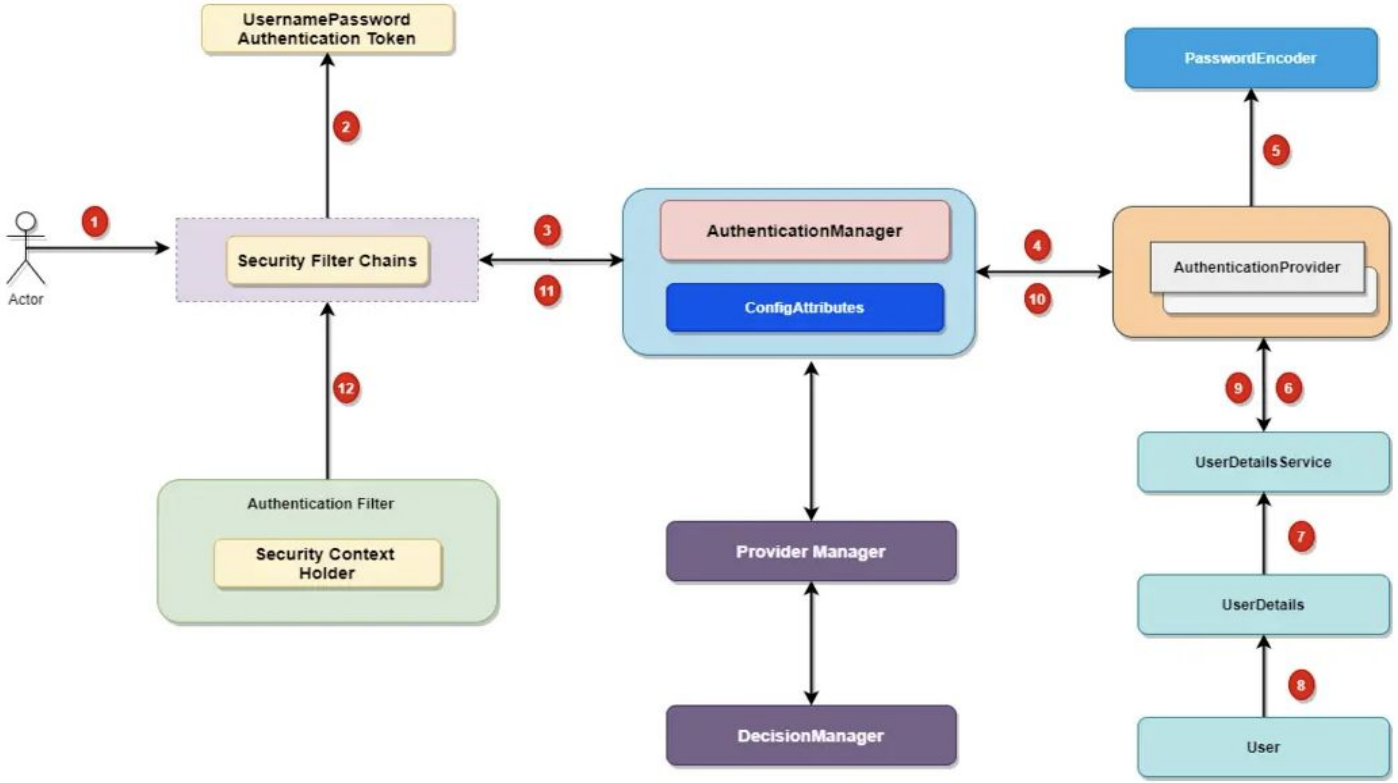
- Son olarak da başarılı bir giriş işlemine ait bilgiler **SecurityContext'te** tutulur.

- Bu bilgiler "**SecurityContextHolder**" vasıtasıyla saklanır.

```
String principal = tokenManager.getUsernameFromToken(jwtToken);
List<SimpleGrantedAuthority>  roles = tokenManager.getRolesFromToken(jwtToken);

UsernamePasswordAuthenticationToken usernamePasswordAuthenticationToken = new UsernamePasswordAuthenticationToken(
        principal,  credentials:  null, roles);

SecurityContextHolder.getContext().setAuthentication(usernamePasswordAuthenticationToken);
```

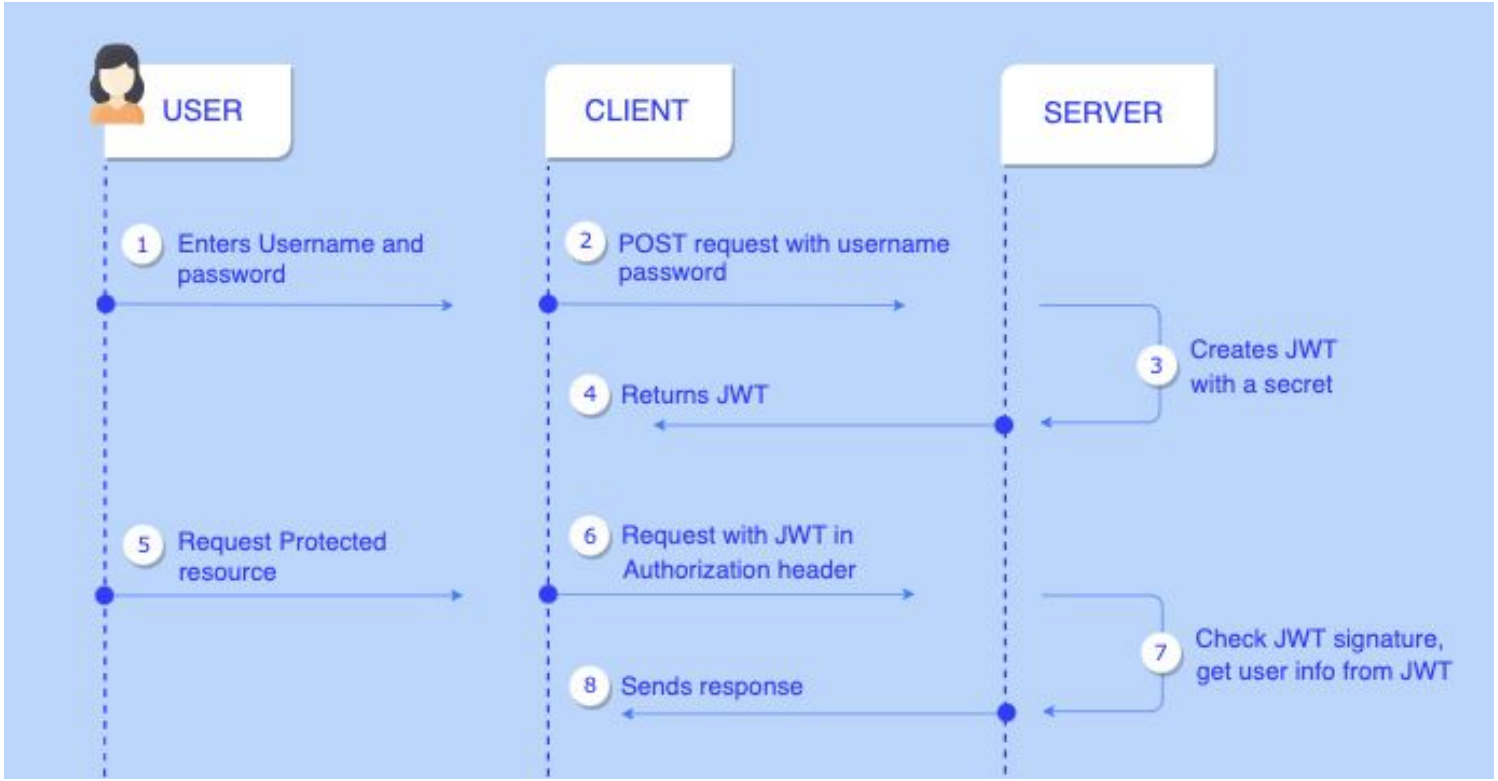
```
Authentication authenticate = authenticationManager
        .authenticate(
                new UsernamePasswordAuthenticationToken(email, password)
        );
```

Spring Security Mimarisi



3- Json Web Token

Json Web Token (JWT)

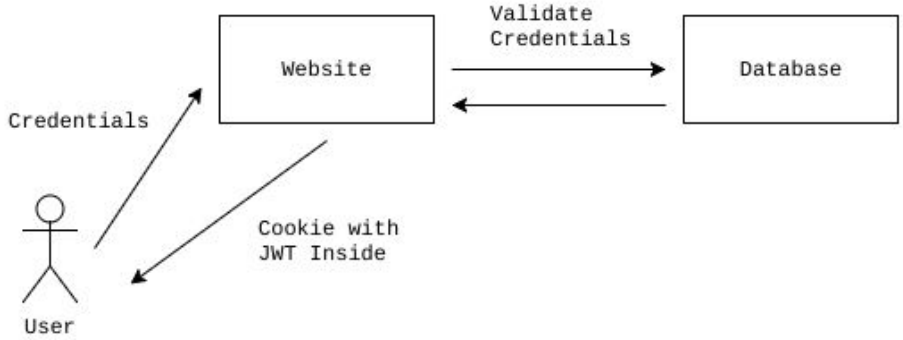


Json Web Token (JWT)

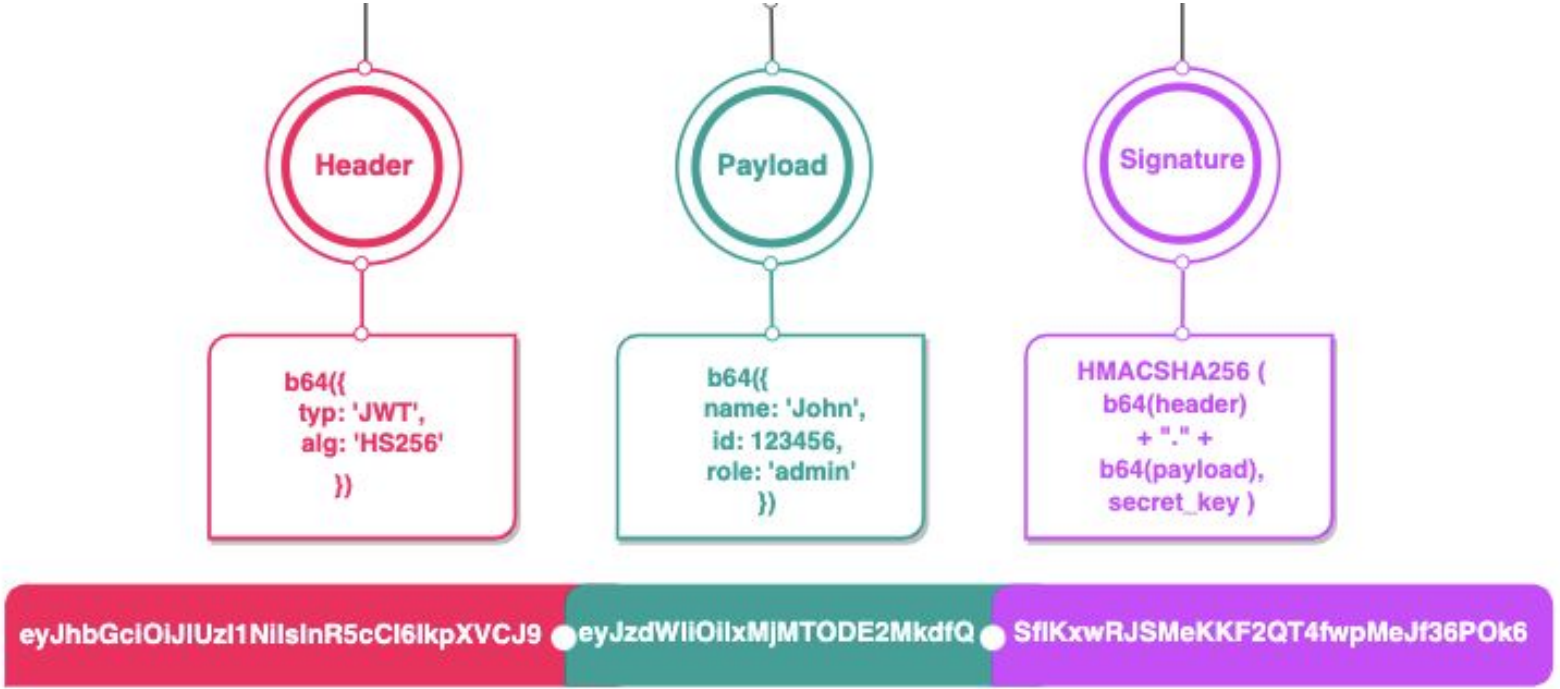
1- **Header**: Mesajı kriptografik olarak imzalayan algoritma tipini saklar.

2- **Payload**: Kullanıcı ile bilgileri saklar. (Claims)

3- **Signature**: Kimlik doğrulama içindir.



Json Web Token (JWT)



Json Web Token (JWT)

- Kimlik doğrulamak için basit JSON veri parçasıdır.

- Bearer Authentication formatında HTTP Header üzerinden sunucuya iletilebilir.

- JWT client tarafında saklanır.

- **Stateless** bir çalışma yöntemini sağlar.

- **JSESSIONID** ihtiyacı ortadan kalkar.

Json Web Token (JWT)

Avantajları

- Yatay ölçeklemeyi kolaylaştırır.

- Güvenli yöntemdir.

- Mobil client için esnek ve uygundur.

- Cookie bloklanan durumlar için alternatiftir.

4- IDOR Atağına Önlem

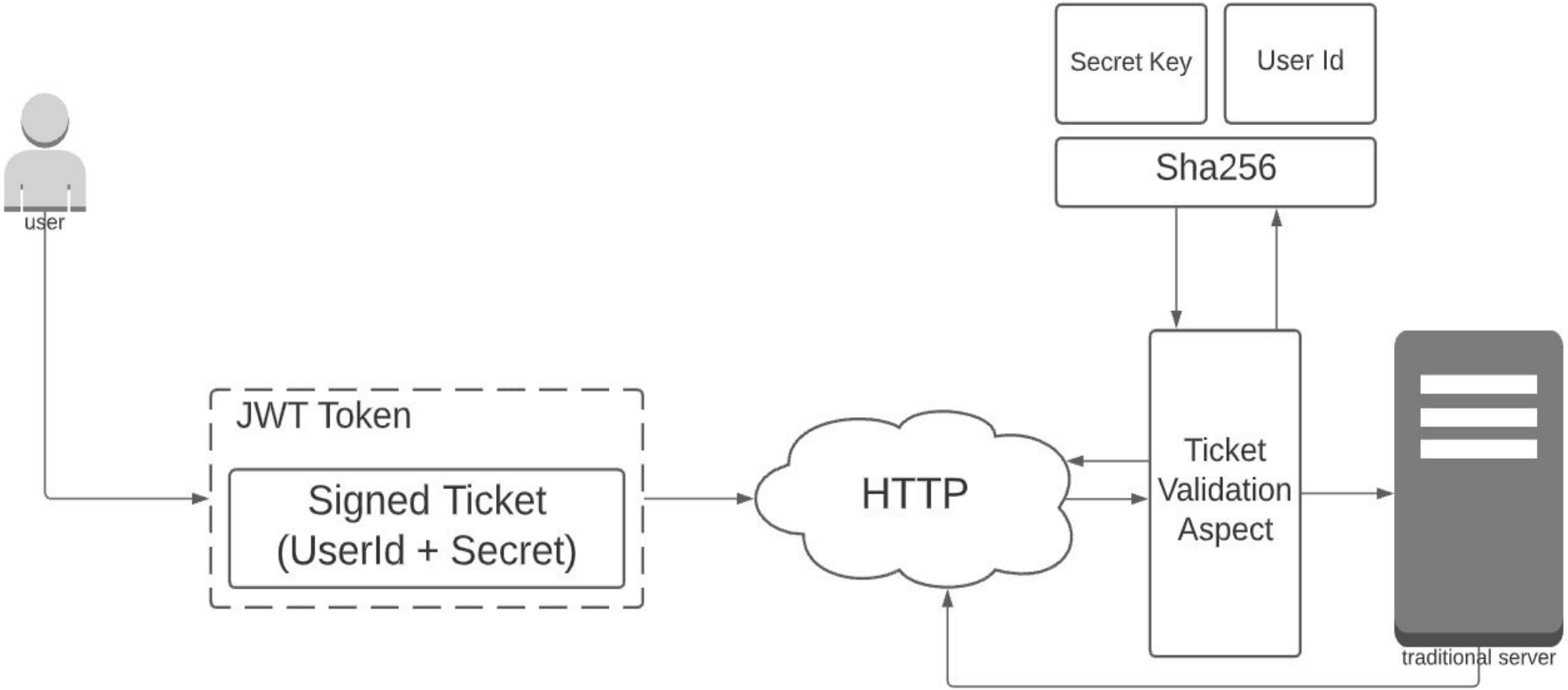
AOP + Kriptografi

Yapılacaklar

- Kullanıcıdan gelen ticketı kontrol eden aspect

- “@IdGuard” isimli custom etiket

AOP + Kriptografi

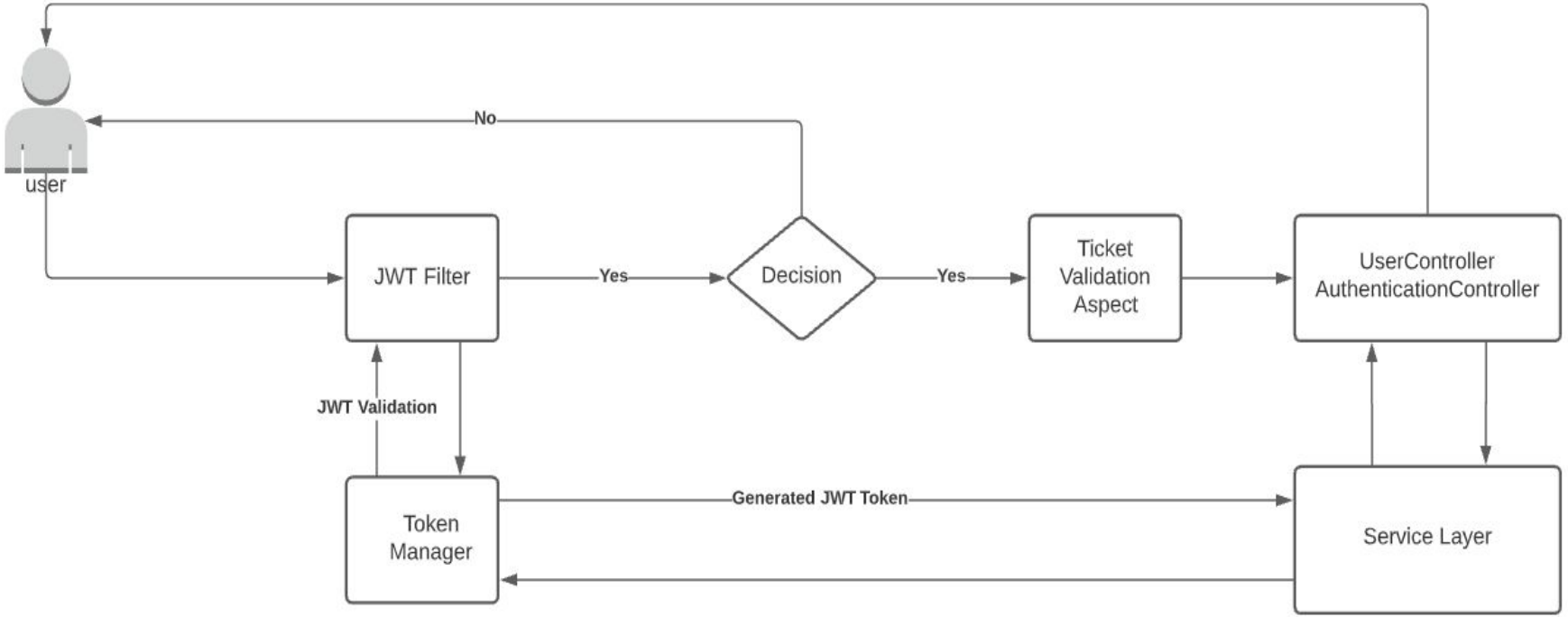


AOP + Kriptografi

```java
@Before("@annotation(idGuard)")
public void execute(JoinPoint joinPoint, IdGuard idGuard) {

    int argIndex = idGuard.parameterIndex();
    if(argIndex < 0){
        argIndex = 0;
    }

    Object[] args = joinPoint.getArgs();
    if(args == null || args.length == 0) {
        throw new RuntimeException("ACCESS ERROR FOR INVALID RESOURCE");
    }

    Object idParameterObj = args[argIndex];
    if(idParameterObj == null) {
        throw new RuntimeException("ACCESS ERROR FOR INVALID RESOURCE");
    }

    String expectedTicket = encryptionManager.encrypt(idParameterObj.toString());

    HttpServletRequest request = ((ServletRequestAttributes) RequestContextHolder.current
    String token = tokenManager.extractJwtFromRequest(request);
    if(!StringUtils.hasText(token)) {
        throw new RuntimeException("ACCESS ERROR FOR INVALID RESOURCE");
    }

    String realTicket = tokenManager.extractTicket(token);
    if(!(StringUtils.hasText(expectedTicket) && expectedTicket.contains(realTicket)) ) {
        throw new RuntimeException("ACCESS ERROR FOR INVALID RESOURCE");
    }
}
```

5- REST API Mimarisi

REST API Mimarisi



Proje URL

Project Source Code:

https://github.com/batux/spring-security-jwt-integration

Swagger URL:

http://localhost:9090/swagger-ui.html

Kaynakça

https://www.baeldung.com/spring-security-session

https://www.javainuse.com/spring/springboot_session

https://spring.io/guides/topicals/spring-security-architecture

https://www.javadevjournal.com/spring-security/spring-security-authentication/

https://www.javainuse.com/webseries/spring-security-jwt/chap4

https://metamug.com/article/security/jwt-java-tutorial-create-verify.html

https://developer.okta.com/blog/2017/08/17/why-jwts-suck-as-session-tokens

Teşekkürler!

/in/batuhanduzgun

@batux

batuhan.duzgun@windowslive.com
www.batuhanduzgun.com