Spring Security i Cross-origin resource sharing (*CORS*)

CORS

- *CORS* je mehanizam koji dozvoljava da se resursima sa jedne web stranice pristupi iz domena koji je različit domenu u kome je servirana pomenuta veb stranica
- Neki *cross-domain* zahtevi (pre svega *Ajax*) su podrazumevano zabranjeni zbog *same-origin security policy*
- *CORS* definiše način na koji server i browser mogu da komuniciraju tako da se bezbedno utvrdi da li propustiti *cross-origin* zahtev
- Pruža veću slobodu, jer je gotovo nemoguće imati zahteve koji se šalju iz istog domena, ali i veću sigurnost umesto da sve zahteve propustimo

CORS

- Ono što je važno jeste da svaki zahtev bude pre svega obrađen kroz CORS
  - *pre-flight request* – OPTIONS zahtev koji proverava da li se koristi CORS protokol i koje su metode dozvoljene
  - Ovakav OPTIONS zahtev se automatski šalje od strane pretraživača i programeri ne moraju sami da ih kreiraju

CORS

- Kako da definišemo koje zahteve propuštamo na server*?*
  - *CORS* je automatski omogućen od Spring verzije 4.2
  - *CORS* definicija
    - Upotrebom anotacije @CrossOrigin (lokalna definicija)
    - Kreiranjem klase koja implementira *WebMvcConfigurer* i redefinisanjem *addCorsMappings* metoeu (globalna definicija)
    - Definisanjem *CorsFilter* (definisanjem *CorsConfigurationSource bean-a;* svodi se na isto kao i *WebMvcConfigurer)*

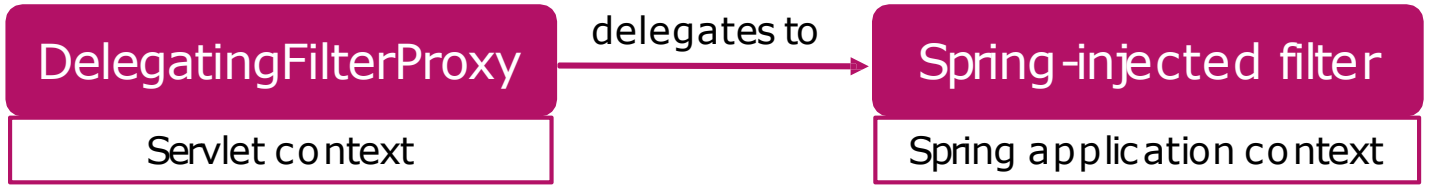
Moduli

- **Modul za konfiguraciju**
  - podrška za konfigurisanje bezbednosti u aplikaciji kroz XML ili Java anotacije
- **Jezgro** (*core*)
  - biblioteka sa ključnim funkcijama za bezbednost
- **Kriptografija**
  - podrška za kriptovanje šifara
- **Veb**
  - podrška za bezbednost veb aplikacije kroz filtere za procesiranje zahteva

- spring-security-core.jar
- spring-security-remoting.jar
- spring-security-web.jar
- spring-security-config.jar
- spring-security-ldap.jar
- spring-security-oauth2-core.jar
- spring-security-oauth2-client.jar
- spring-security-oauth2-jose.jar
- spring-security-acl.jar
- spring-security-cas.jar
- spring-security-openid.jar
- spring-security-test.jar

Filtriranje veb zahteva

- Spring sadrži generički filter `DelegatingFilterProxy` koji može da delegira posao specifičnom filteru



- Spring Security tu može da injektuje svoj specifičan filter `springSecurityFilterChain`
- Spring Boot automatski injektuje ovaj *bean* kroz anotaciju `@EnableWebSecurity`

Konfiguracija bezbednosti

- Spring Security podstiče bezbednost zasnovanu na komponentama. Pre se zasnivala na konfiguraciji sa WebSecurityConfigurerAdapter klasom i redefinisanjem odgovarajućih metoda.
- Bitne metode:
  - `filterChain(HttpSecurity http)`
    - definiše prava pristupa za zahteve ka određenim URL.
  - `webSecurityCustomizer()`
    - generalna bezbednost veb aplikacije
    - ignorisanje resursa, *firewall*, ...

Autentifikacija korisnika

- Za autentikaciju korisnika potrebno je da postoje uskladišteni podaci o korisnicima aplikacije i njihovim ulogama u sistemu
- Spring ima ugrađenu podršku za različite standardne pristupe skladištenju
  - *in-memory*
  - jdbc
  - LDAP
- Može se realizovati i nestandardna podrška za utvrđivanje identiteta korisnika

Reprezentacija korisnika

- Korisnik je predstavljen interfejsom `UserDetails`
- Za korisnika se minimalno evidentira:
  - **korisničko ime**
  - **lozinka**
  - **lista uloga** (uloga predstavljena interfejsom `GrantedAuthority`)
- Način pristupa je predstavljen interfejsom `UserDetailsService`

Nestandardna autentifikacija korisnika

- Interfejs `UserDetailsService` ima samo metodu

  `loadUserByUsername(String username)`

- Potrebno je implementirati interfejs i redefinisati ovu metodu na proizvoljan način
  - npr. korišćenjem JPA repozitorijuma za pristup bazi korisnika
- Pri konfiguraciji autentifikacije u metodi `filterChain` je potrebno navesti koja klasa je zadužena za autentifikaciju

Presretanje HTTP zahteva

- Spring security presreće HTTP zahteve posebnim filterom
- Može se definisati da li korisnik ima pravo da dobije odgovor na zahtev koji je poslat na određenu putanju
- Putanje se definišu u stilu koji koristi Ant alat za izgradnju softvera
  - Definisanje se vrši u metodi `filterChain(HttpSecurity )`
- Definiše se ko ima pravo da pristupi putanji:
  - ulogovani korisnici
  - svi korisnici
  - oni koji imaju određenu ulogu u sistemu
  - oni koji upućuju zahtev sa određene IP adrese
  - ...

Šifrovanje lozinki

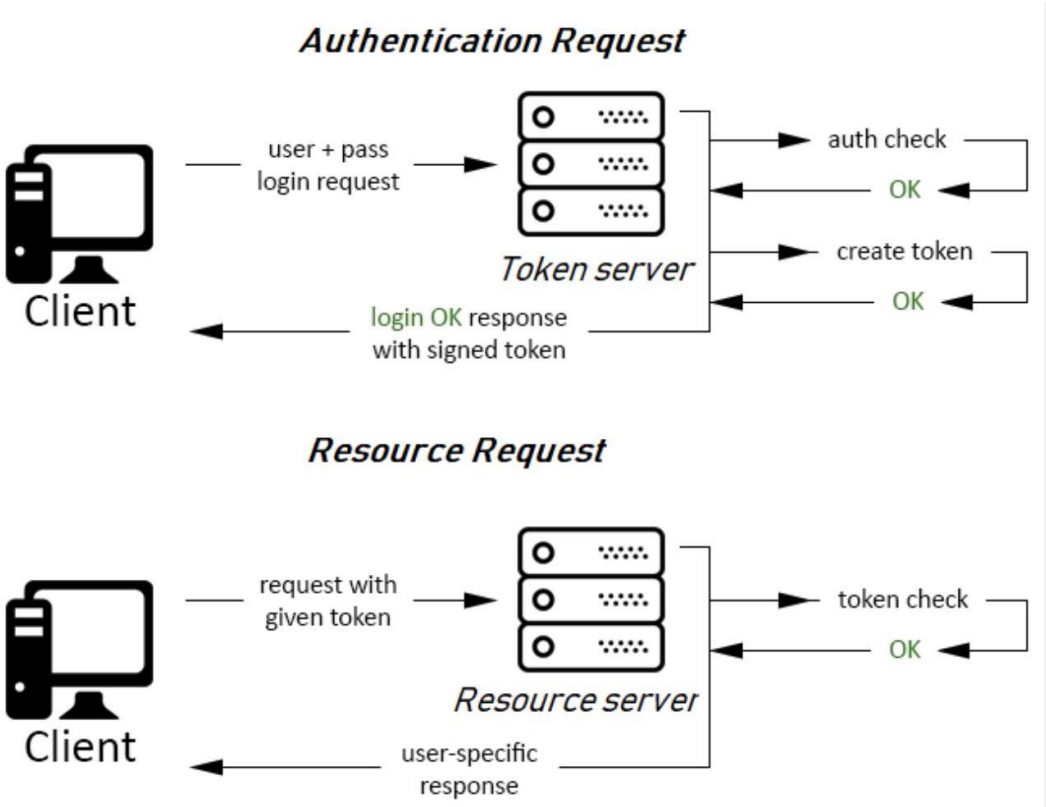
- Spring security ima ugrađenu podršku za skladištenje lozinki u šifrovanom obliku
- Dovoljno je instancirati odgovarajuću klasu za šifrovanje lozinki i primeniti je pri konfigurisanju načina identifikacije korisnika

```
@Bean
public DaoAuthenticationProvider authenticationProvider() {
    DaoAuthenticationProvider authProvider = new DaoAuthenticationProvider();
    authProvider.setUserDetailsService(userDetailsService());
    authProvider.setPasswordEncoder(passwordEncoder());

    return authProvider;
}
```

Autentifikacija bazirana na tokenima

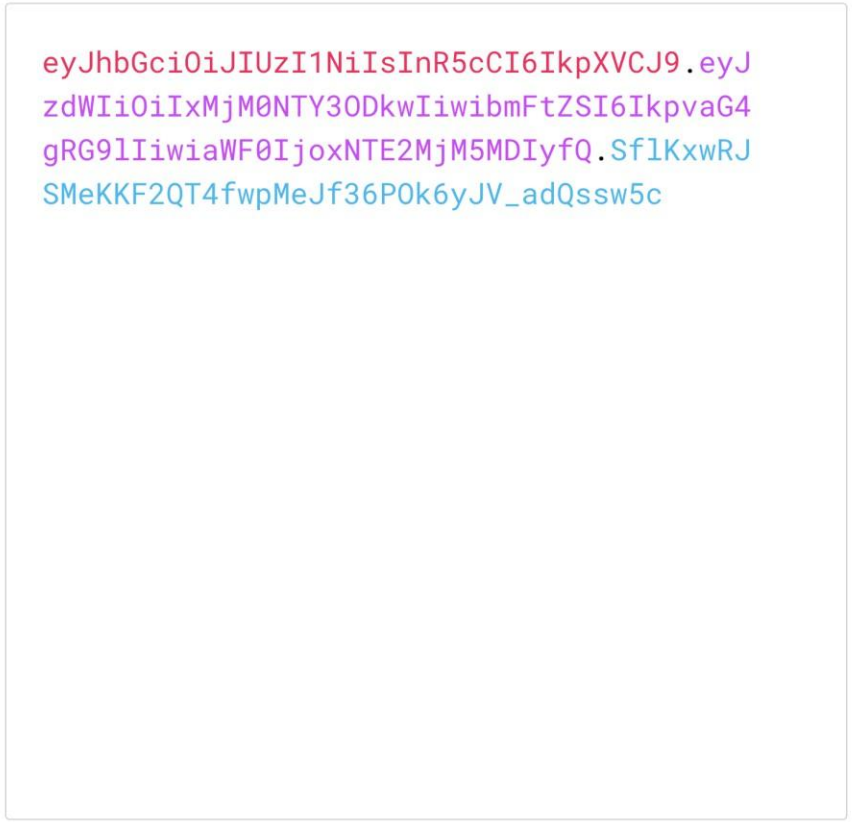
- HTTP protokol je *stateless* pa je potreban mehanizam identifikacije klijenta koji šalje višestruke zahteve
- Nakon uspešne autentifikacije, klijent dobija token koji je server generisao
- Sa svakim sledećim zahtevom, klijent šalje token kao informaciju o svom identitetu

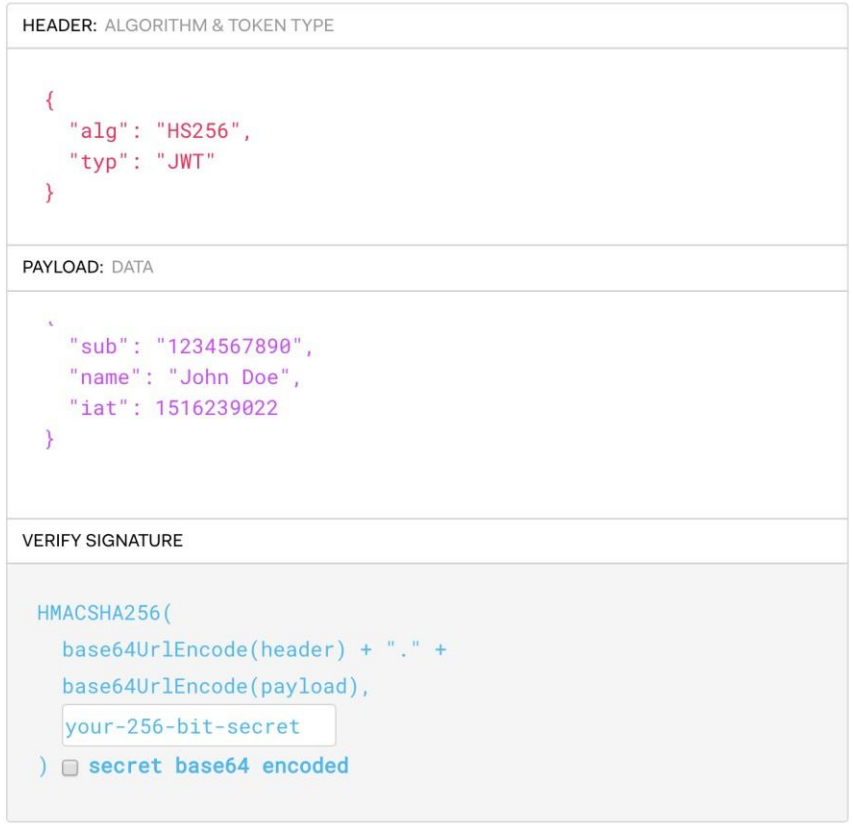


JSON veb token (JWT)

- Trenutno dominantan format u kojem se tokeni za autentikaciju reprezentuju
- https://jwt.io/
- To je string koji se sastoji iz tri dela:
  1. **zaglavlje** *(header)*
     - tip tokena i algoritam kriptovanja
  2. **glavni sadržaj** *(payload)*
     - podaci o korisniku na kojeg se token odnosi, rok trajanja tokena, ...
  3. **potpis** *(signature)*
     - string generisan kriptovanjem zaglavlja, sadržaja i serverove tajne reči

JSON veb token (JWT)





Realizacija JWT autentikacije kroz Spring Security

- Potrebno je nakon uspešnog logovanja generisati JWT token i poslati ga klijentu
- Svaki zahtev je potrebno presresti kako bi se proverilo da li sadrži validan JWT token
  - presretanje se vrši dodavanjem novog filtera
  - filter će postaviti odgovarajuću autentifikaciju ako je token validan
  - dalje Spring security dozvoljava pristup putanji zavisno od uspešnosti autentikfiacije i ranije konfigurisanih prava pristupa putanji