

Add the dependency:

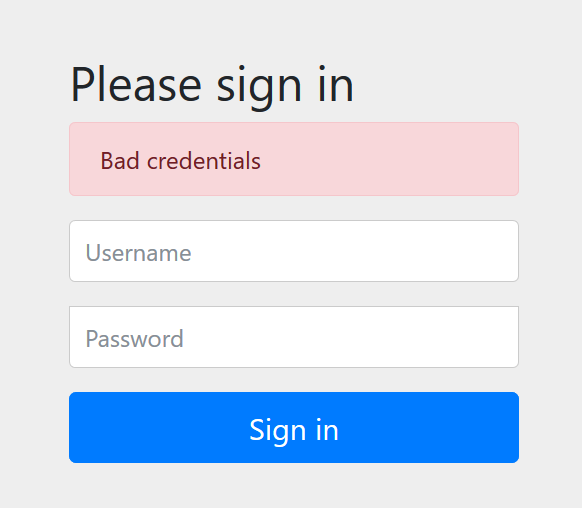
<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-security</artifactId>

</dependency>

Onec added the dependency we get login page:

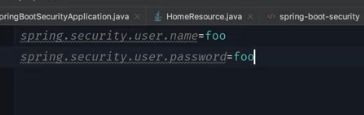


Default user : user

Password will be in console,

Using generated security password: a80204ce-af3d-4e38-b122-bb7ba7828196

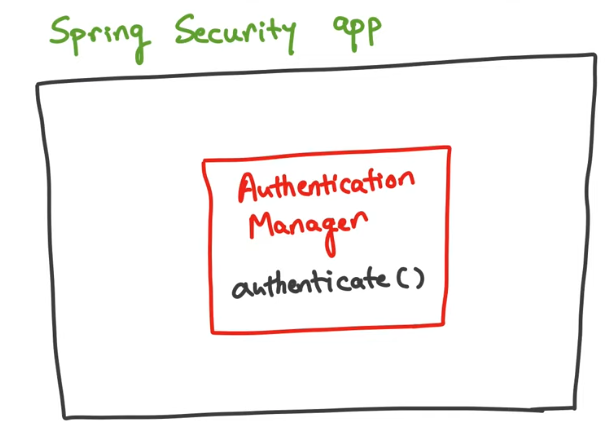
Creating the custom user:

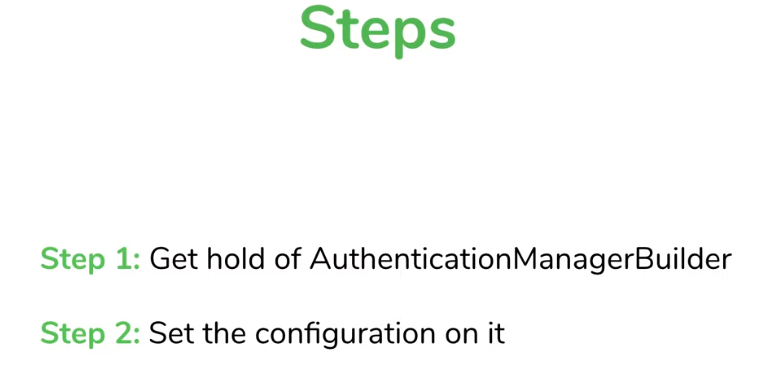


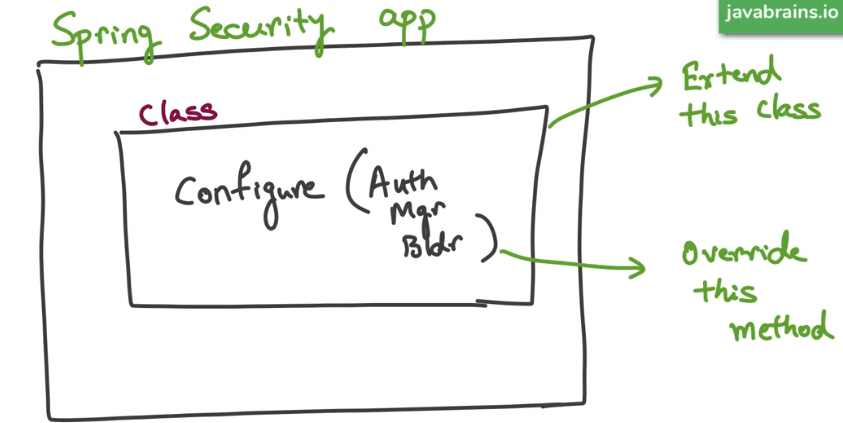
FILTER



Configuration the authentication in spring Security:







**import** org.springframework.context.annotation.Bean;

**import** org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;

**import** org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

**import** org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;

**import** org.springframework.security.crypto.password.~~NoOpPasswordEncoder~~;

**import** org.springframework.security.crypto.password.PasswordEncoder;

@EnableWebSecurity

**public** **class** SecurityConfiguration **extends** WebSecurityConfigurerAdapter {

@Override

**protected** **void** configure(AuthenticationManagerBuilder auth) **throws** Exception{

auth.inMemoryAuthentication()

.withUser("sachin")

.password("sachin")

.roles("USER")

.and() // for multiple users

.withUser("foo")

.password("foo")

.roles("ADMIN")

;

}

@Bean

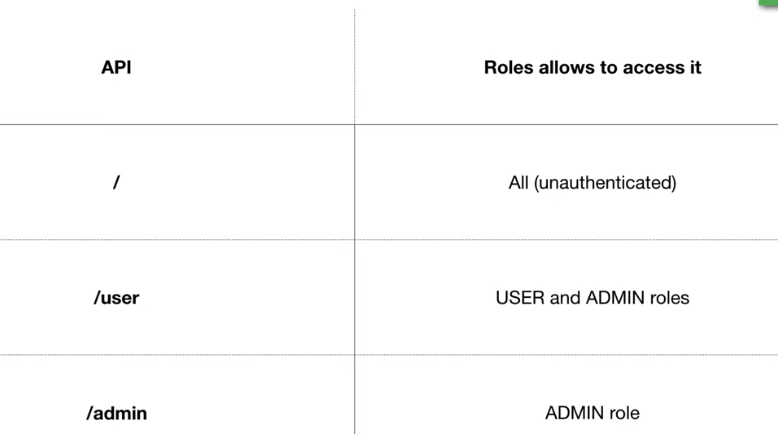
**public** PasswordEncoder getPasswordEncoder() {

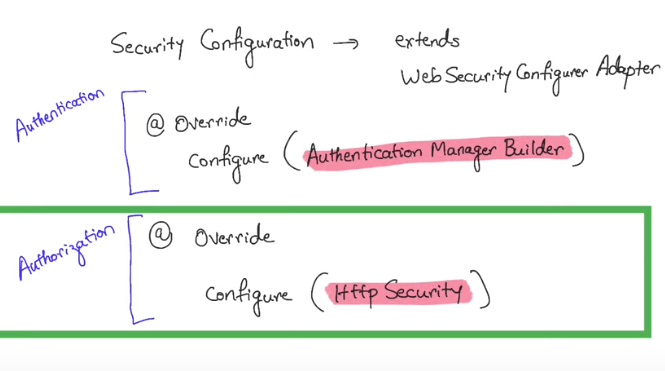
**return** ~~NoOpPasswordEncoder~~.~~getInstance~~();

}

}

# How to configure Spring Security Authorization





@Override

**protected** **void** configure(HttpSecurity http) **throws** Exception{

http.authorizeRequests()

.antMatchers("/admin").hasRole("ADMIN")

.antMatchers("/user").hasAnyRole("USER","ADMIN")

.antMatchers("/").permitAll()

.and().formLogin();

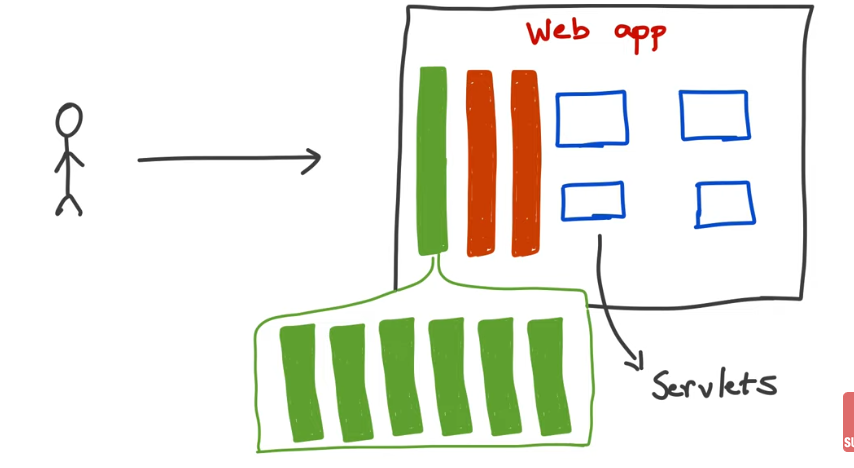
}

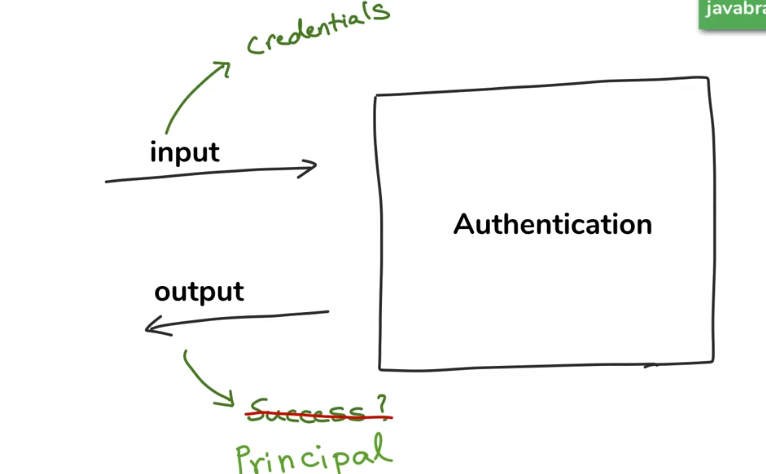
# How Spring Security Authentication works

Spring Security uses filter for API authentication.



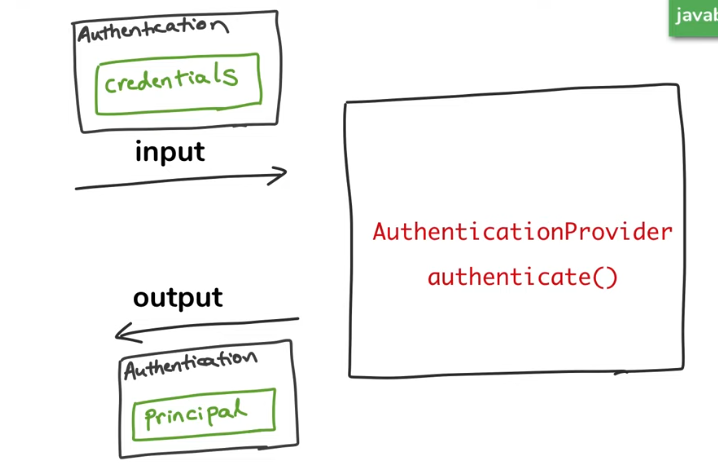
Filter: Is Constrict which filter the request.

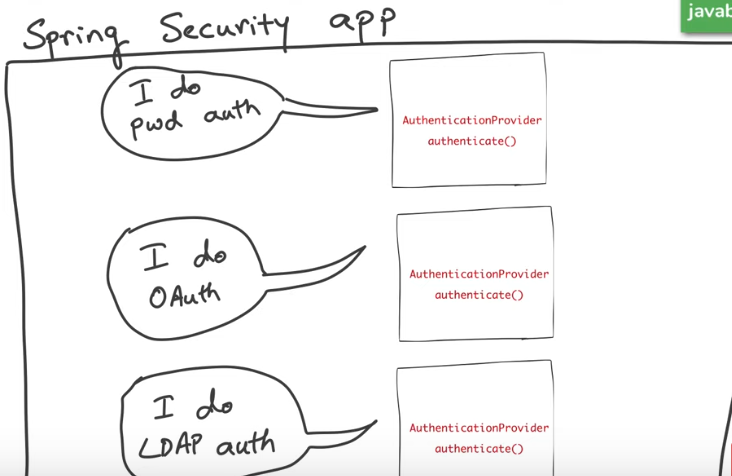


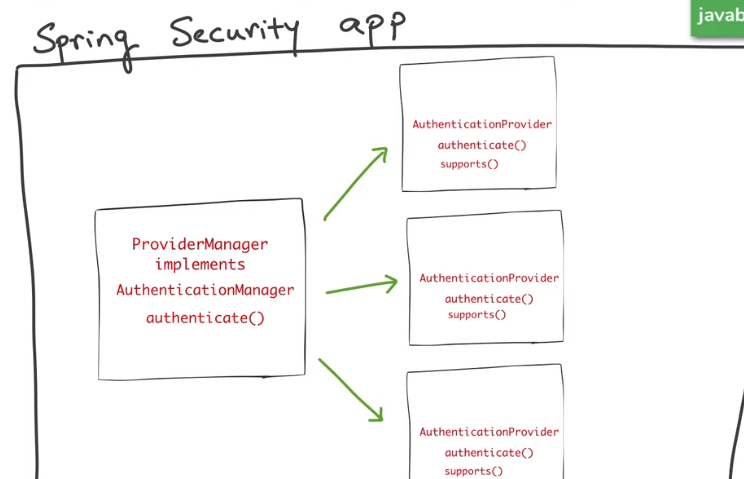


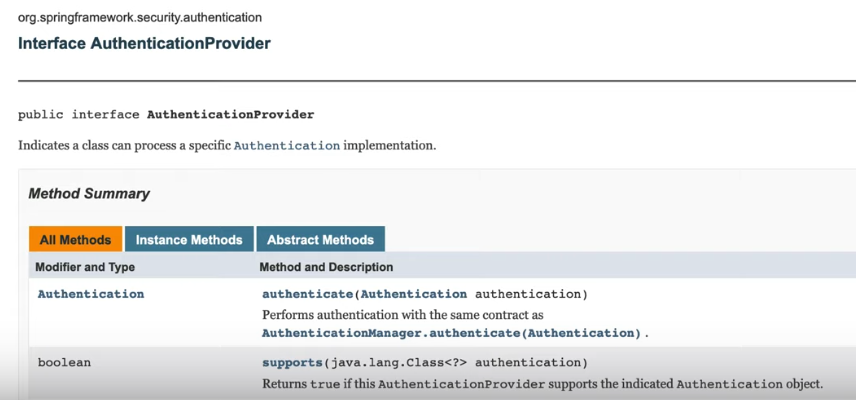
Returns principal.

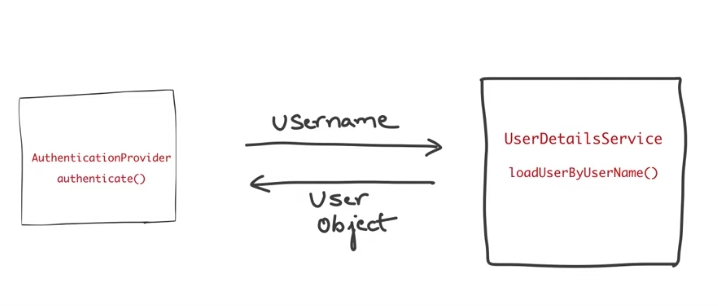
Authentication is done by AuthenticationProvider.

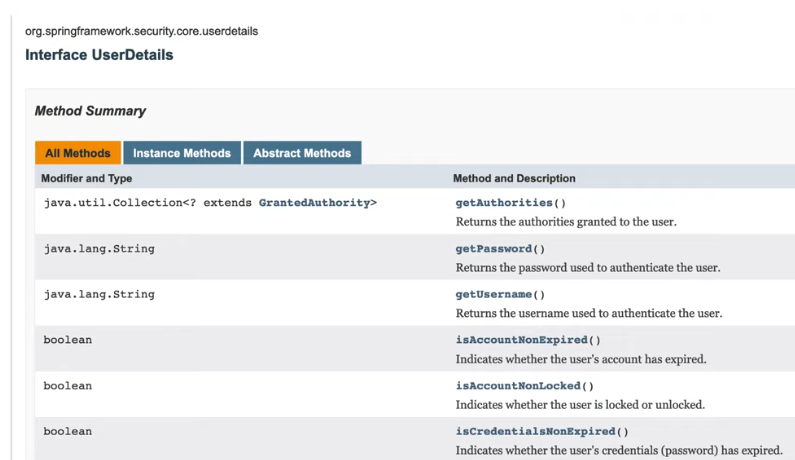


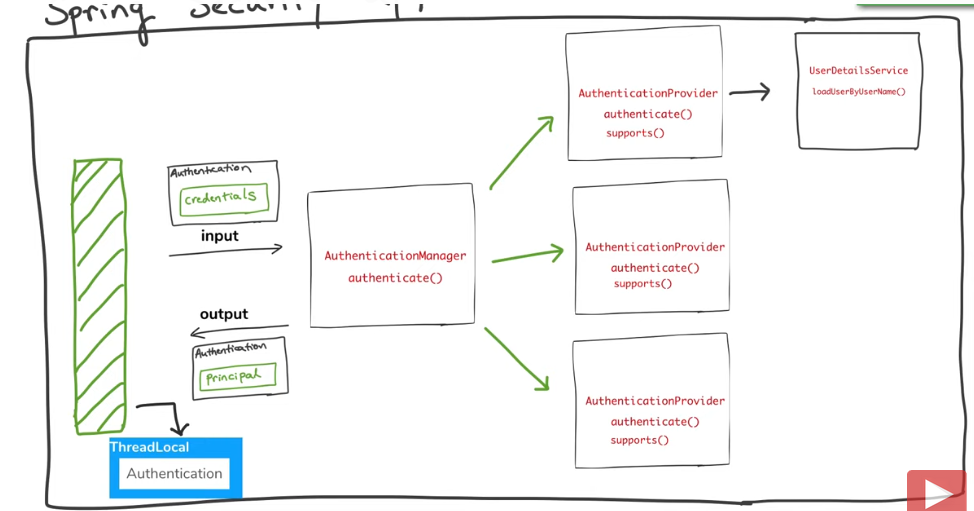




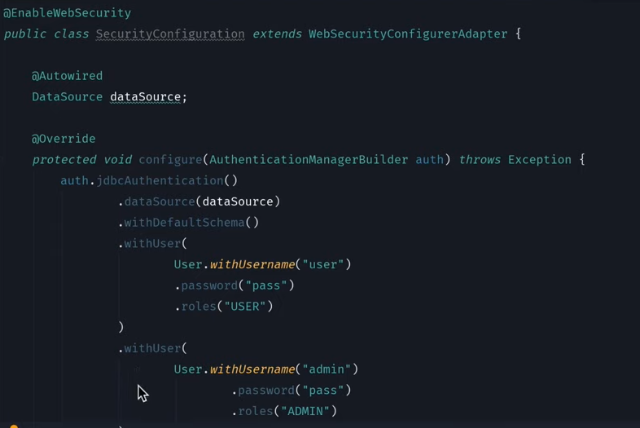








# How to setup JDBC authentication with Spring Security from scratch

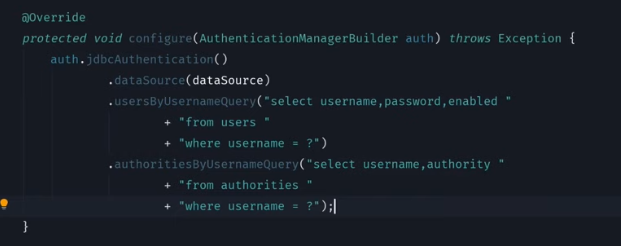




Default Schema :

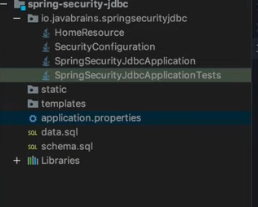
<https://docs.spring.io/spring-security/site/docs/current/reference/htmlsingle/#user-schema>

User defined schema mapping :

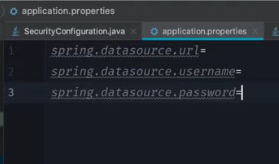


Schema.sql

Data.sql

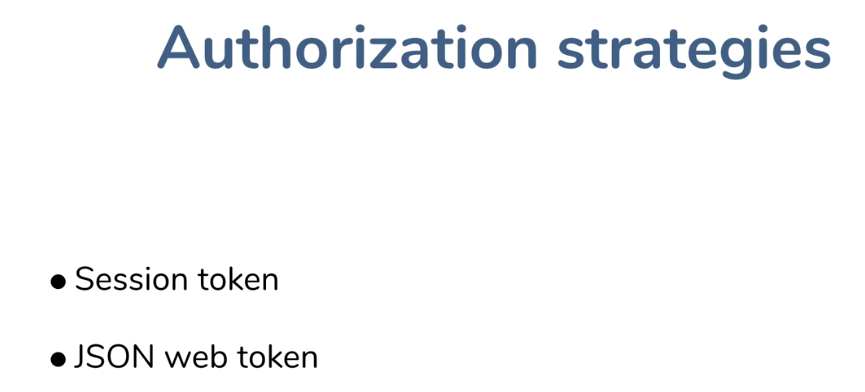


For External Data source configuration:



# What is JWT authorization really about

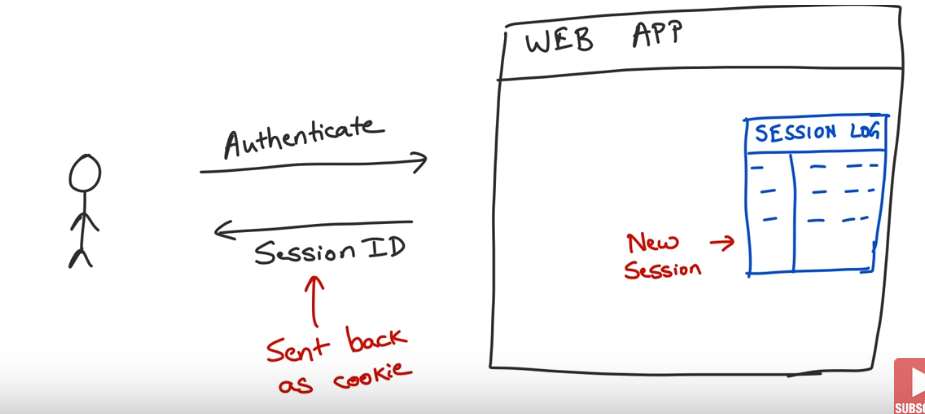
JWT : JAWT ( JSON Web Token) is used for authorization.



Request + IDENTITY

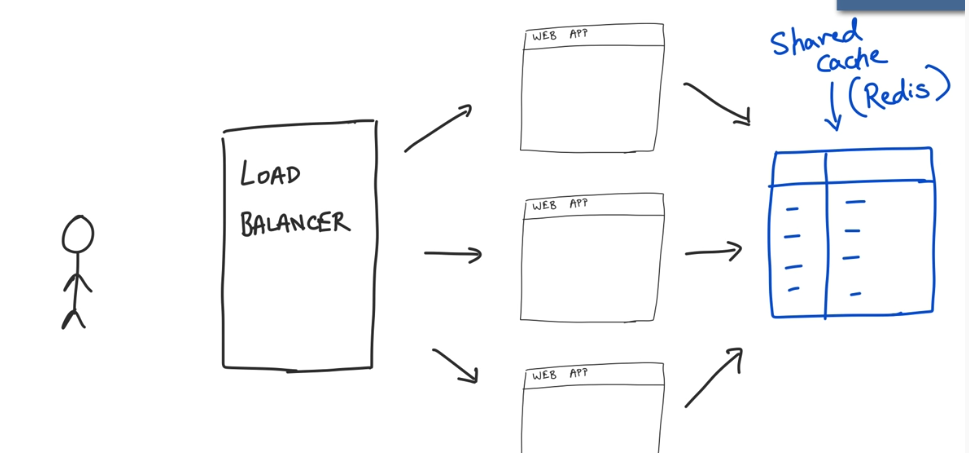
Session Token : SessionID + Cookies

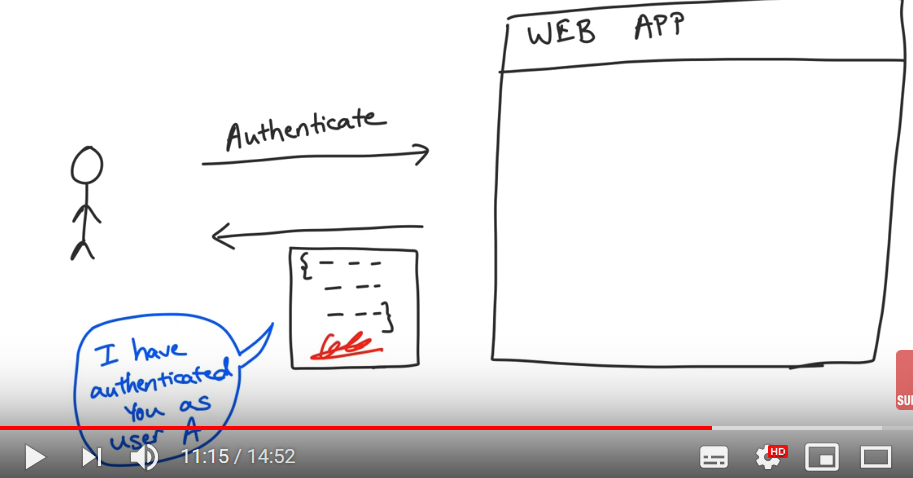
Session ID is stored in Cookies is most commonly used.



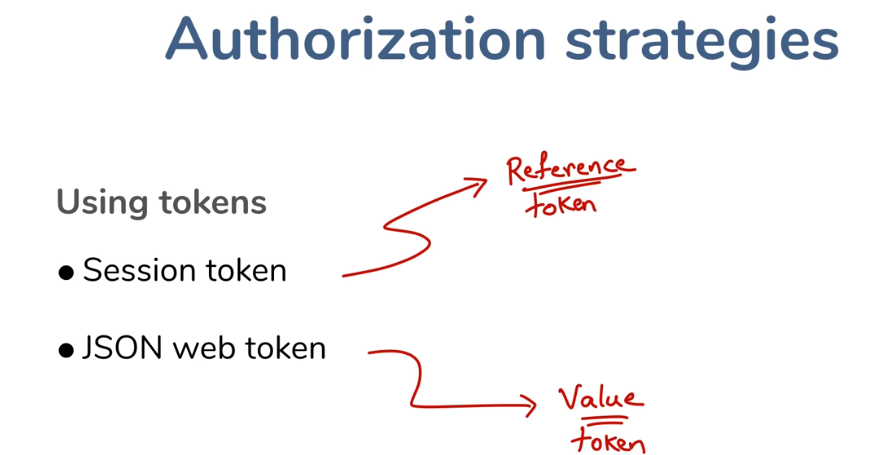
Problem :

* If the application is load Balanced the its difficult get session from server.
* So Redis shared cache came into picture. If the redis is down the whole session is down.
* So Sticky session pattern.





Signed token



# What is the structure of a JWT





<https://jwt.io/>

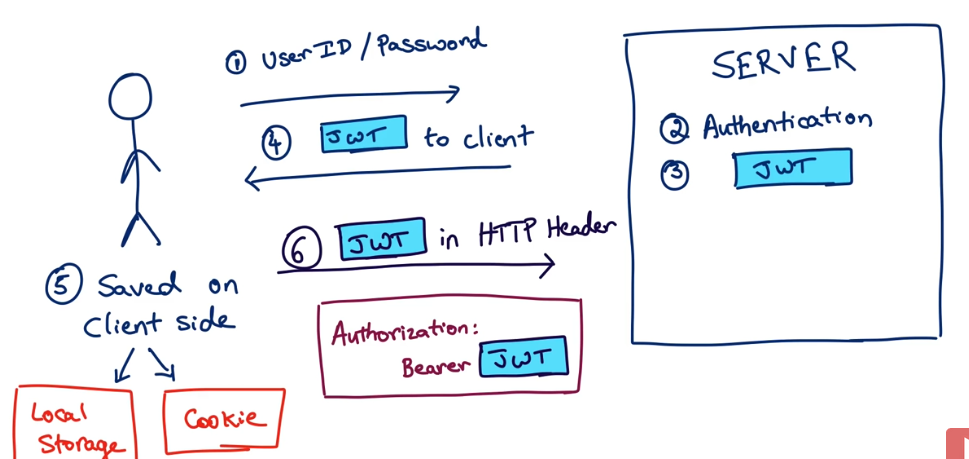
HMACSHA256(

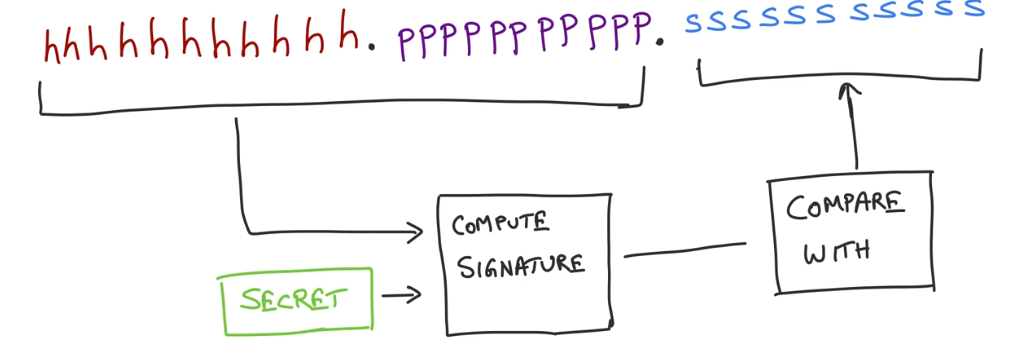
base64UrlEncode(header) + "." +

base64UrlEncode(payload),



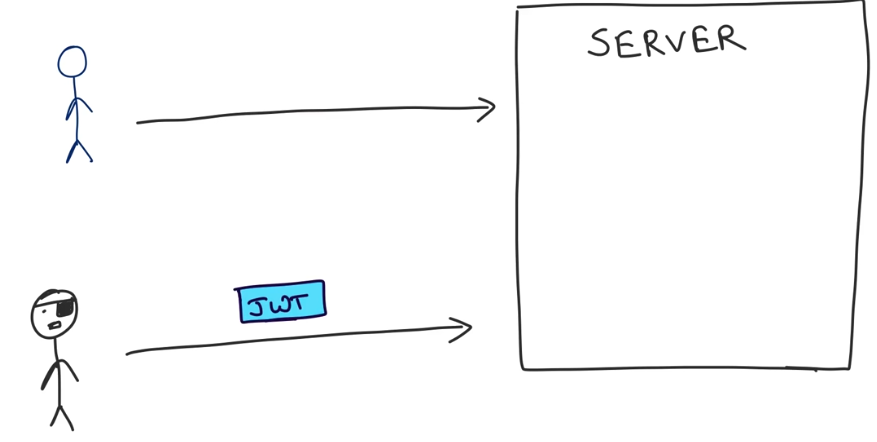
) secret base64 encoded





Problem with JWT:

* How secure is JWT if it is readable by anyone?
* What if someone steals my JWT and user it themselves?
* Server just verifies if JWT is correct. Doesn’t know who sent it.



Solution :

* Send JWT in https
* Use under congestion mechanize authentication and authorization (oAuth) .

How to disable the JWT?

Blacklisted JWT