1. What is spring security?

* Custom inbuilt authentication and access control framework
* Major advance is customizability

1. Core features of spring security?

* Extensive authentication and authorization
* Effective detection and prevention of attacks like cross-script forgery, session fixation and clickjacking
* Seamless integration with Servlet API
* Integration with Spring Web MVC features.

1. Authentication

* Biometric Authentication
* Token-based
* Multifactor

1. Authorization

* Verifying user permission

1. Spring Security OAuth 2.0?

* OAuth 2.0 is an authorization protocol that enables a client application to access protected resources through the authorized server.

1. What is the security filter chain in spring security?
2. What is JWT?
3. What are some security annotations that can involve spring expression language(SpEL)?

* @PreAuthorize
* @PostAuthorize
* @PreFilter
* @PreFilter
* @PostFilter
* @Secured

1. Important filter classes for Spring security?

* UserNamePasswordAuthenticationFilter
* BasicAuthenticationFilter
* SessionManagementFilter
* SecurityContextPersistenceFilter
* ExceptionTranslationFilter
* FilterSecurityInterceptor

1. What are access controls in spring security and how are they implemented?
2. What is the use of @Secured and @roleAllowed annotation in spring Security?
3. Define mutual authentication and provide an example of its usage.
4. Which servlet filter is responsible for intercepting incoming requests in the spring security application?

<https://javatechonline.com/spring-security-without-websecurityconfigureradapter/>

JWT

1. JSON Web token – Authorization Information
2. Structure of JWT ?

* Header
* Payload
* Signature

1. Set expiry date

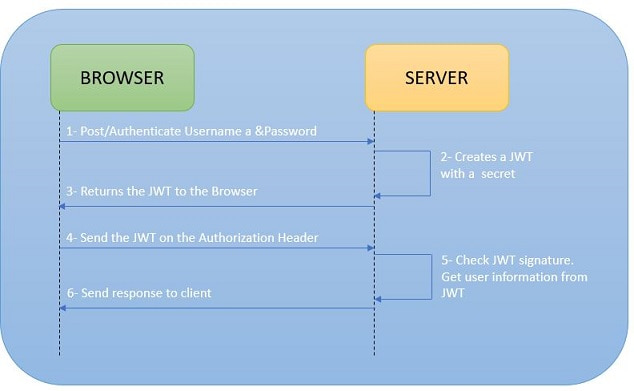
private String doGenerateToken(Map<String, Object> claims, String subject) {

return Jwts.builder().setClaims(claims).setSubject(subject).setIssuedAt(new Date(System.currentTimeMillis()))

.setExpiration(new Date(System.currentTimeMillis() + jwtExpirationInMs)).signWith(SignatureAlgorithm.HS512, secret).compact();

}

1. Workflow of JWT



1. Advantage of JWT ?

* Good performace
* Portable
* Cookies not allowed
* Digitally signed
* Setting expiry time.
* Plaintext

1. OAuth

* Authorization Protcol
* Authenticate using JWT doesnot allow to logout
* To make a proper logout use OAuth

1. JWS – stateless authentication

* Authorizing server doesn’t keep track of anything.
* No need to store the user information ins the session.

1. JWT has password ?

* Allows to regerenate the password.
* Has encoded user information.

1. AbstractSecurityInterceptor in spring Security ?

* Handles initial authorization of an incoming request .
* FilterSecurityInterceptor and MethodSecurityInterceptor are Implementation of AbstractSecurityInterceptor
* FilterSecurityInterceptor 🡪 default Filter.All authenticated user request will be authorized by FilterSecurityInterceptor
* MethodSecurityInterceptor 🡪 Implement method level security.Enables us to apply security at method level.

1. Spring boot Methos level security ?

* @PreAuthorize annotation used on controller method level security.

1. @PreAuthorise and @Secured

* Check the authorization before executing the method.
* @Secured - the roles cant be couples with AND /OR condition
* Spring expression language is not supported by @Secured annotation.

1. Difference between hasRole() and has Authority() ?

* hasRole() defines the Role (for Example: "Employee" or "Visitor"), while hasAuthority() defines the Rights (for Example: One Employee can only use the Main Door, but another one can also use the Backdoor

1. Spring Security’s @PreAuthorize and HttpSecurity ?

* HttpSecurity function rejects the requests in web request Filter
* HttpSecurity is associated with URL endpoints
* @PreAuthorize ,occurs later , before the controller method is executed.
* @PreAuthorize is associated with controller methods.
* @PreAuthorize uses SpEL 9Spring Expression Language)

1. @PreAuthorize annotation is enabled by @EnableglobalMethodSecurity(prePostEnabled = true)

@Component

@EnableWebSecurity

@EnableGlobalMethodSecurity(prePostEnabled = true, securedEnabled = true)

public class SecurityConfig extends WebSecurityConfigurerAdapter {

.......

.......

}

1. How to use Authorization based on OAuth 2.0 with PreAuthorize



Secure REST APi

1. Authentication and Authorization.

Authentication –

* + Identify the user from others
  + Using username and password
  + Example : Fingerprint ,security token ,security questions ,
  + SAML token received through SSO Login

Authorization :

* + Determine “What user is allowed to do”
  + Permission provided by system admin to access to the system

1. 4 ways to secure RESTFUL Services
   * BASIC Authentication
   * DIGEST Authentication
   * Client CERT Authentication
   * OAuth2 API keys
2. RESTful webservices security Implemetnation.

* Using SecurityContext
* Using Annotations

BASIC Authentication

* + Login using username and password
  + Encoded using Base64 Format
  + But not encrypted or hashed in anyway.
  + Any sniffer can read the data from the network.

DIGEST Authentication

* + Efficient password security as MD5, SHA, BCrypt ,Scrypt algorithms
  + Reset the password enach time when login is happened.

Client CERT Authentication

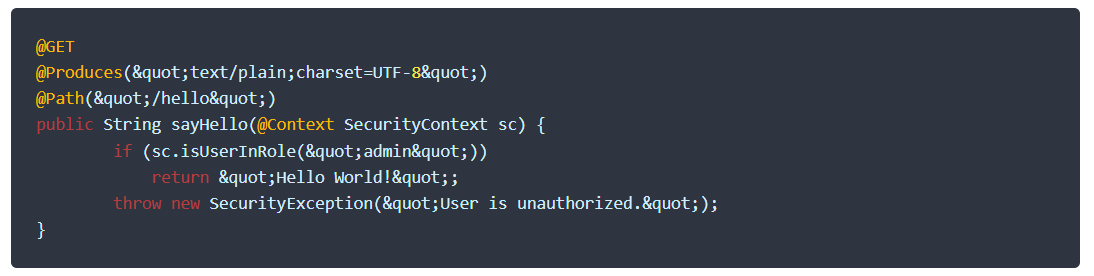
* + Certificate Authority
  + HTTPs communication procotol
  + Don’t have secure channel to prevent anyone from stealing the client;s identity.

OAuth2 API Keys

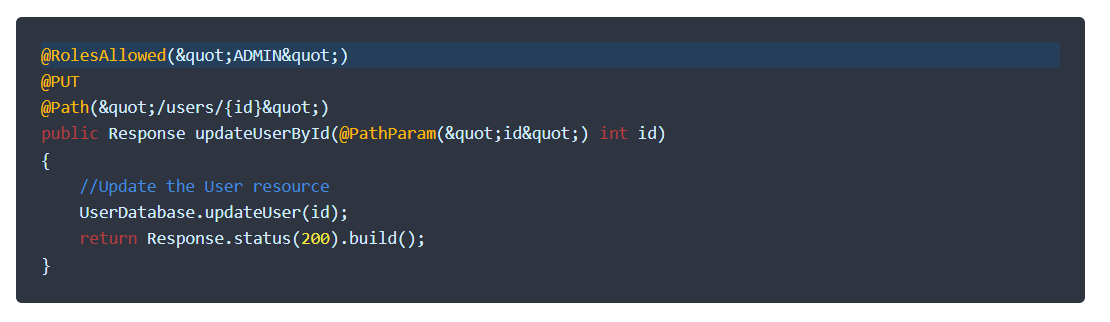
* + Site Authetication using API key and secret key belongs to the user.Provider will alow you to authenticate access to public information

Security Implementation

* Javax.ws.rs.core.SecurityContext interface provide access to security -related information for request and very similar to javax.servlet.http.HttpServletRequest



* JAR-RS annotation for method level authorization
  + Widely used in Enterprise Application and used to verify roles and responsibilities of an authenticated used for certain operation
    - @PermitAll
    - @DenyAll
    - @RolesAllowed



REST API Security Best Practices :

1. Use only HTTPS protocol , so that whole communication is always encrypted.
2. Never send auth credentials or API keys as query param. They may appear in URL and can be logged or tracked easily.
3. Use harder encryption levels
4. PUT,POST and DELETE request is protected from Cross Site Request Forgery,
5. Validate the input data ASAP in server method.Use ony primitive data as input parameter
6. Rely on frameworj provided validation features as tested by larger community.

https://medium.com/@harzaliolfa6/spring-security-part-1-a733a9e11170

Spring Security framework provides easy-to-use solution for java application.

Allows you to preconfigure or customize security features.

First objective is to protect http request processed by your application.

Once Spring security is insatalled it will go through

HTTP firewall , Proxy and filters

1. HTTP firewall it filters the communication flow and the information enters the application.

2. Manages access to the protected resource(Web application ) I takes care of classifying HTTP traffic and directs it to the appropriate servlet filters in the filter chain.

3. Filters ensures all the HTTP request are secure .

The collection of implemented servlet filters is called Spring security filter chain.

DelegatingFilterproxy class (implements the proxy design pattern) Spring security filter chain integrated into your application. Security filters then added to create security engine for application.

Create Spring boot project with below dependencies

* Spring Web – for web releated features
* Spring security – provide security level before even you initialize the application.
* OAuth 2 Client – to special security protocol on your protection form

Login into bank application to check the bank statement.

Access control has 2 steps

Authentication and Authorization

Authentication

Single factor authentication – identification based on one step authentication

Muti factor authentication -with finger prints

Browser handles authentication in different ways

Session based authentication

Server saves the session and sends acopy ( cookie)and also saved in users browser (user credentials +length of the seeion and session number ) and the server checks the validity. (stateful)

Token bbased authentication

Jwt : JavaScript Object Notation that encodes the information and transmits .only authorization server understands it .its more efficient than cookie.

Authorization is applied after authentication .

Authentication ensures that the user has the right information to prove that they are who they say they are.

Authorization ensures that the authenticated user only goes to the pages he is authorized to visit.

Role based access control is provided

Spring security provides default loginform to create login page .