Follow @only2dhir

Spring Boot Security Oauth2 Jwt Auth Example

By Dhiraj, 14 March, 2018 124K

In this article, we will be discussing about OAUTH2 implementation with spring boot security and JWT token and securing REST APIs.In my last article of Spring Boot Security OAUTH2 Example, we created a sample application for authentication and authorization using OAUTH2 with default token store but spring security OAUTH2 implementation also provides functionality to define custom token store.Here, we will be creating a sample spring security OAUTH2 application using JwtTokenStore.Using JwtTokenStore as token provider allows us to customize the token generated with TokenEnhancer to add additional claims. For a role based OAUTH2 implementation, you can visit this article.

Most of the configurations in this application are very similar to my previous article of spring security OAUTH2 implementation and hence we may avoid some common codes and configuration that we built in our last application. Let us start with a brief introduction of JWT and then we will dive into creating our authorization server, resource server and later we will discuss about adding custom claims in the token.If you don't want to use OAUTH2 and simply want to create an authentication process using JWT token, then you can visit my previous article of Using JWT with Spring Boot Security With Angular.

Json Web Token

JSON Web Token (JWT) is an open standard (RFC 7519) that defines a compact and self-contained way for securely transmitting information between parties as a JSON object.a stateless authentication mechanism as the user state is never saved in server memory.A JWT token consists of 3 parts seperated with a dot(.) i.e. Header.payload.signature

Header has 2 parts type of token and hashing algorithm used.The JSON structure comprising these two keys are Base64Encoded.

Recommended

- Spring Boot Oauth2 Angular
- Jwt Role Based Authorization
- Angular Jwt Authentication
- Spring Boot Jwt Auth
- Spring Boot Security Hibernate Login Example
- Spring Jms Activemq Integration Example
- Spring Boot Security Oauth2 Example
- React Js Jwt Authentication Example

Payload contains the claims.Primarily, there are three types of claims: reserved, public, and private claims. Reserved claims are predefined claims such as iss (issuer), exp (expiration time), sub (subject), aud (audience).In private claims, we can create some custom claims such as subject, role, and others.

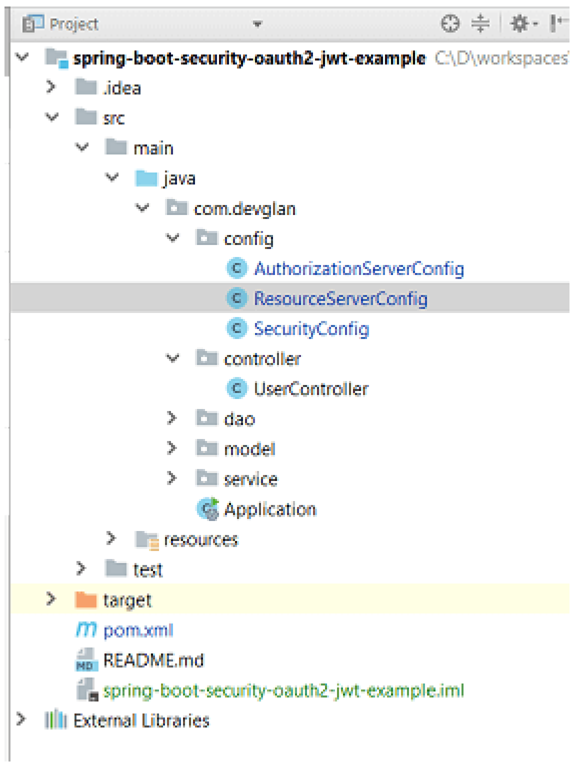
```
{
  "sub": "Alex123",
  "scopes": [
    {
      "authority": "ROLE_ADMIN"
    }
  ],
  "iss": "http://devglan.com",
  "iat": 1508607322,
  "exp": 1508625322
}
```

Signature ensures that the token is not changed on the way.For example if you want to use the HMAC SHA256 algorithm, the signature will be created in the following way:

```
HMACSHA256(
  base64UrlEncode(header) + "." +
  base64UrlEncode(payload),
  secret)
```

Following is a sample JWT token.Here is a full stack spring boot application with jwt authentication application to secure REST APIs using jwt token mechanism.

Project Structure



```
Project
spring-boot-security-oauth2-jwt-example C:\D\workspaces
    .idea
    src
        main
            java
                com.devglan
                    config
                        AuthorizationServerConfig
                        ResourceServerConfig
                        SecurityConfig
                    controller
                        UserController
                    dao
                    model
                    service
                    Application
            resources
        test
    target
    pom.xml
    README.md
    spring-boot-security-oauth2-jwt-example.iml
External Libraries
```

Maven Dependencies

Here, spring-security-jwt provides JwtTokenStore support.

```
<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-security</artifactId>
</dependency>
<dependency>
        <groupId>org.springframework.security.oauth</groupId>
        <artifactId>spring-security-oauth2</artifactId>
</dependency>
<dependency>
        <groupId>org.springframework.security</groupId>
        <artifactId>spring-security-jwt</artifactId>
</dependency>
```

Authorization Server Config

I hope you are familiar of OAUTH2 architecture and authorization server. I have explained about it in my last article of OAUTH2.The following configuration is very similar to our last configuration of Spring Boot Security OAUTH2 Example apart from the JwtAccessTokenConverter and TokenStore.Here, JwtAccessTokenConverter is the helper that translates between JWT encoded token values and OAuth authentication information. We have added our custom signature to make the JWT token more robust.Apart from JwtTokenStore, spring security also provides InMemoryTokenStore and JdbcTokenStore.

For a 3rd party authorization server such as Google, you can visit this - Spring Boot OAuth2 with Google.

`ClientDetailsServiceConfigurer` : a configurer that defines the client details service. Client details can be initialized, or you can just refer to an existing store.

`AuthorizationServerSecurityConfigurer` : defines the security constraints on the token endpoint.

`AuthorizationServerEndpointsConfigurer` : defines the authorization and token endpoints and the token services.

ClientDetailsServiceConfigurer can be used to define an in-memory or JDBC implementation of the client details service.In this example, we are using an in-memory implementation.

AuthorizationServerConfig.java

```
        static final String CLIEN_ID = "devglan-client";
        static final String CLIENT_SECRET = "devglan-secret";
        static final String GRANT_TYPE_PASSWORD = "password";
        static final String AUTHORIZATION_CODE = "authorization_code";
    static final String REFRESH_TOKEN = "refresh_token";
    static final String IMPLICIT = "implicit";
        static final String SCOPE_READ = "read";
        static final String SCOPE_WRITE = "write";
    static final String TRUST = "trust";
        static final int ACCESS_TOKEN_VALIDITY_SECONDS = 1*60*60;
    static final int FREFRESH_TOKEN_VALIDITY_SECONDS = 6*60*60;

        @Autowired
        private AuthenticationManager authenticationManager;

        @Bean
        public JwtAccessTokenConverter accessTokenConverter() {
        JwtAccessTokenConverter converter = new JwtAccessTokenConverter();
        converter.setSigningKey("as466gf");
        return converter;
        }

    @Bean
    public TokenStore tokenStore() {
        return new JwtTokenStore(accessTokenConverter());
    }

        @Override
        public void configure(ClientDetailsServiceConfigurer configurer) throws Except
ion {

                configurer
                                .inMemory()
                                .withClient(CLIEN_ID)
                                .secret(CLIENT_SECRET)
                                .authorizedGrantTypes(GRANT_TYPE_PASSWORD, AUTHORIZATI
ON_CODE, REFRESH_TOKEN, IMPLICIT )
                                .scopes(SCOPE_READ, SCOPE_WRITE, TRUST)
                                .accessTokenValiditySeconds(ACCESS_TOKEN_VALIDITY_SECO
NDS).
                                refreshTokenValiditySeconds(FREFRESH_TOKEN_VALIDITY_SE
CONDS);
        }

        @Override
        public void configure(AuthorizationServerEndpointsConfigurer endpoints) throws
Exception {
                endpoints.tokenStore(tokenStore())
                                .authenticationManager(authenticationManager)
                .accessTokenConverter(accessTokenConverter());
        }
}
```

Resource Server Config

Resource in our context is the REST API which we have exposed for the crud operation.To access these resources, client must be authenticated.In real-time scenarios, whenever an user tries to access these resources, the user will be asked to provide his authenticity and once the user is authorized then he will be allowed to access these protected resources.

`resourceId:` the id for the resource (optional, but recommended and will be validated by the auth server if present).

JwtAccessTokenConverter implementation in resource server too.

Here, we have configured that /users is a protected resource and it requires an ADMIN role for the access.

For an integration with Google along with a custom login, you can visit this article - Spring Security OAuth2 Google Registration

```
@Configuration
@EnableResourceServer
public class ResourceServerConfig extends ResourceServerConfigurerAdapter {

        private static final String RESOURCE_ID = "resource_id";

        @Override
        public void configure(ResourceServerSecurityConfigurer resources) {
                resources.resourceId(RESOURCE_ID).stateless(false);
        }

        @Override
        public void configure(HttpSecurity http) throws Exception {
        http.
                anonymous().disable()
                .authorizeRequests()
                .antMatchers("/users/**").access("hasRole('ADMIN')")
                .and().exceptionHandling().accessDeniedHandler(new OAuth2AccessDeniedHandler());
        }

}
```

REST APIs

Now let us expose some protected REST resource using spring controller.

```
@RestController
@RequestMapping("/users")
public class UserController {

    @Autowired
    private UserService userService;

    @RequestMapping(value="/user", method = RequestMethod.GET)
    public List listUser(){
        return userService.findAll();
    }

    @RequestMapping(value = "/user", method = RequestMethod.POST)
    public User create(@RequestBody User user){
        return userService.save(user);
    }

    @RequestMapping(value = "/user/{id}", method = RequestMethod.DELETE)
    public String delete(@PathVariable(value = "id") Long id){
        userService.delete(id);
        return "success";
    }

}
```

Following is the userservice implementation to validate user.

```
        @Autowired
        private UserDao userDao;

        public UserDetails loadUserByUsername(String userId) throws UsernameNotFoundException {
                User user = userDao.findByUsername(userId);
                if(user == null){
                        throw new UsernameNotFoundException("Invalid username or password.");
                }
                return new org.springframework.security.core.userdetails.User(user.getUsername(), user.getPassword(), getAuthority());
        }

        private List getAuthority() {
                return Arrays.asList(new SimpleGrantedAuthority("ROLE_ADMIN"));
        }

        public List findAll() {
                List list = new ArrayList<>();
                userDao.findAll().iterator().forEachRemaining(list::add);
                return list;
        }
}
```

Above userservice is configured in SecurityConfig.java as below. You can use this Online Bcrypt Calculator to genertae Bcrypt password.

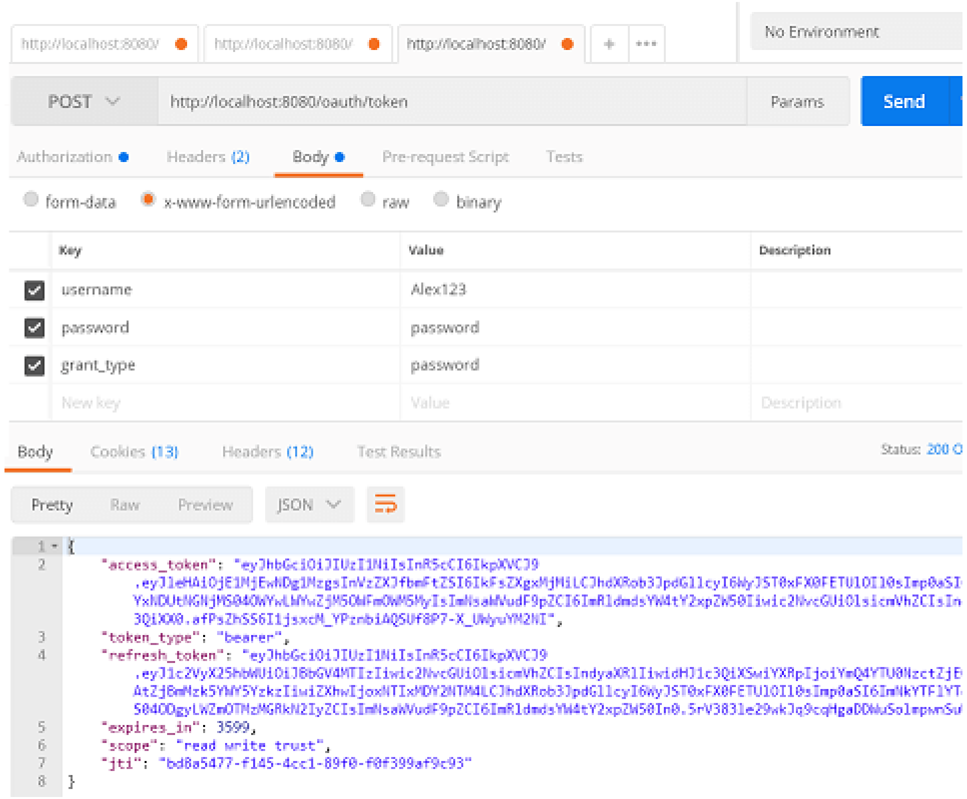
```
public class SecurityConfig extends WebSecurityConfigurerAdapter {

    @Resource(name = "userService")
    private UserDetailsService userDetailsService;

    @Override
    @Bean
    public AuthenticationManager authenticationManagerBean() throws Exception {
        return super.authenticationManagerBean();
    }

    @Autowired
    public void globalUserDetails(AuthenticationManagerBuilder auth) throws Exception {
        auth.userDetailsService(userDetailsService)
                .passwordEncoder(encoder());
    }

    @Override
    protected void configure(HttpSecurity http) throws Exception {
        http
                .csrf().disable()
                .anonymous().disable()
                .authorizeRequests()
                .antMatchers("/api-docs/**").permitAll();
    }

    @Bean
    public BCryptPasswordEncoder encoder(){
        return new BCryptPasswordEncoder();
    }

    @Bean
    public FilterRegistrationBean corsFilter() {
        UrlBasedCorsConfigurationSource source = new UrlBasedCorsConfigurationSource();
        CorsConfiguration config = new CorsConfiguration();
        config.setAllowCredentials(true);
        config.addAllowedOrigin("*");
        config.addAllowedHeader("*");
        config.addAllowedMethod("*");
        source.registerCorsConfiguration("/**", config);
        FilterRegistrationBean bean = new FilterRegistrationBean(new CorsFilter(source));
        bean.setOrder(0);
        return bean;
    }
}
```

Create User Script

```
INSERT INTO User (id, username, password, salary, age) VALUES (1, 'Alex123', '$2a$04$I9Q2sDc4QGGg5WNTLmsz0.fvGv3OjoZyj81PrSFyGOqMphqfS2qKu', 3456, 33);
INSERT INTO User (id, username, password, salary, age) VALUES (2, 'Tom234', '$2a$04$PCIX2hYrve38M7eOcqAbCO9UqjYg7gfFNpKsinAxh99nms9e.8HwK', 7823, 23);
INSERT INTO User (id, username, password, salary, age) VALUES (3, 'Adam', '$2a$04$I9Q2sDc4QGGg5WNTLmsz0.fvGv3OjoZyj81PrSFyGOqMphqfS2qKu', 4234, 45);
```

Testing OAUTH2 JWT Application

as `devglan-client` and `devglan-secret`. This will result access_token, token_type, refresh_token, expiry etc.



POST http://localhost:8080/oauth/token

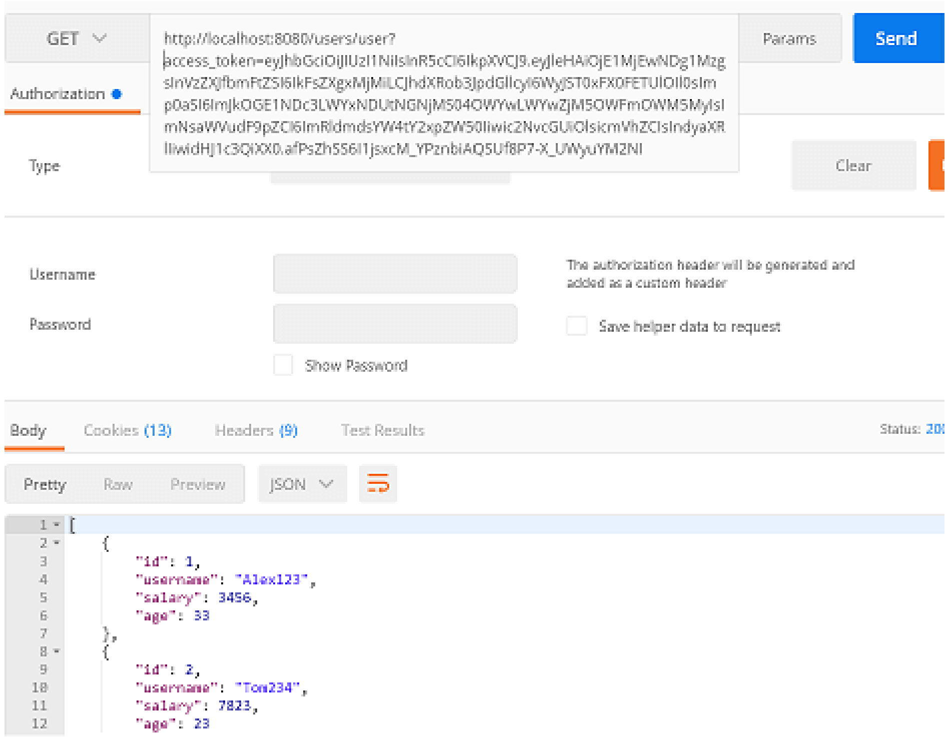
Body: x-www-form-urlencoded

Key	Value	Description
☑ username	Alex123	
☑ password	password	
☑ grant_type	password	

Status: 200

```
{
    "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9
        .eyJleHAiOjE1MjEwNDg1MzgsInVzZXJfbmFtZSI6IkFsZXgxMjMiLCJhdXRob3JpdGllcyI6WyJST0xFX0FETU1OIl0sImp0aSI
        YxNDUtNGNjMS04OWYwLWYwZjM5OWFmOWM5MyIsImNsaWVudF9pZCI6ImRldmdsYW4tY2xpZW50Iiwic2NvcGUiOlsicmVhZCIsIn
        3QiXX0.afPsZhSS6I1jsxcM_YPznbiAQ5Uf8P7-X_UWyuYM2NI",
    "token_type": "bearer",
    "refresh_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9
        .eyJ1c2VyX25hbWUiOiJBbGV4MTIzIiwic2NvcGUiOlsicmVhZCIsIndyaXRlIiwidHJ1c3QiXSwiYXRpIjoiYmQ4YTU0NzctZjE
        AtZjBmMzk5YWY5YzkzIiwiZXhwIjoxNTIxMDY2NTM4LCJhdXRob3JpdGllcyI6WyJST0xFX0FETU1OIl0sImp0aSI6ImNkYTF1YT
        504ODgyLWZmOTMzMGRkN2IyZCIsImNsaWVudF9pZCI6ImRldmdsYW4tY2xpZW50In0.5rV383le29wkJq9cqHgaDDWuSolmpwnSu
    "expires_in": 3599,
    "scope": "read write trust",
    "jti": "bd8a5477-f145-4cc1-89f0-f0f399af9c93"
}
```

Now, we can use the same token to access protected resources.



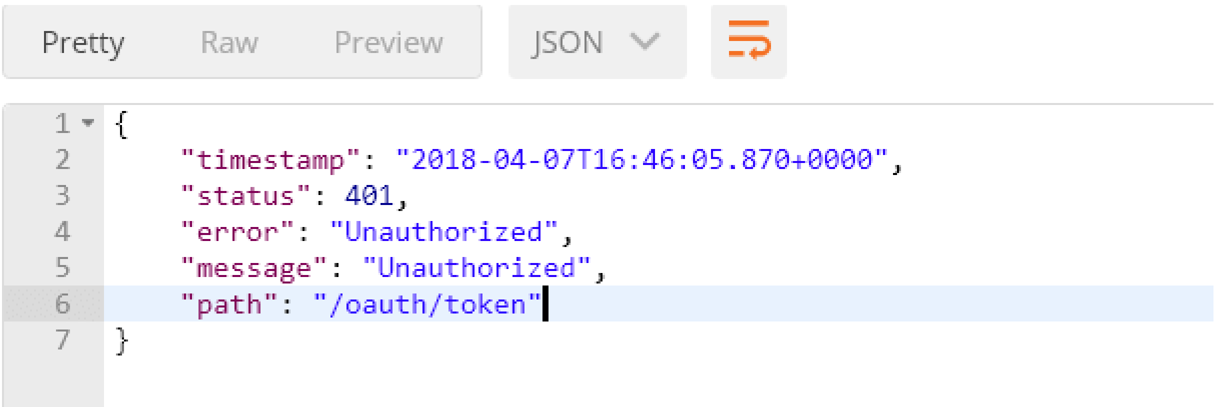
GET http://localhost:8080/users/user?access_token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE1MjEwNDg1MzgsInVzZXJfbmFtZSI6IkFsZXgxMjMiLCJhdXRob3JpdGllcyI6WyJST0xFX0FETUlOIl0sImp0aSI6ImJkOGE1NDc3LWYxNDUtNGNjMS04OWYwLWYwZjM5OWFmOWM5MyIsImNsaWVudF9pZCI6ImRldmdsYW4tY2xpZW50Iiwic2NvcGUiOlsicmVhZCIsIndyaXRlIiwidHJ1c3QiXX0.afPsZhSS6I1jsxcM_YPznbiAQ5Uf8P7-X_UWyuYM2NI

Status: 20

```
[
    {
        "id": 1,
        "username": "Alex123",
        "salary": 3456,
        "age": 33
    },
    {
        "id": 2,
        "username": "Tom234",
        "salary": 7823,
        "age": 23
```

Spring Boot 2 OAUTH2

While running this application with above configurations in Spring Boot 2, you will find below error.



```
{
    "timestamp": "2018-04-07T16:46:05.870+0000",
    "status": 401,
    "error": "Unauthorized",
    "message": "Unauthorized",
    "path": "/oauth/token"
}
```

Following are the changes in `pom.xml` to make this example work with spring boot 2.

```
<parent>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>2.0.0.RELEASE</version>
</parent>
...
<dependency>
        <groupId>org.springframework.security.oauth</groupId>
        <artifactId>spring-security-oauth2</artifactId>
        <version>2.0.10.RELEASE</version>
</dependency>
```

For Spring Boot 2 you need to Bcrypt CLIENT_SECRET,so in AuthorizationServerConfig.java change line 17 into:

```
static final String CLIENT_SECRET = "$2a$04$e/c1/RfsWuThaWFCrcCuJeoyvwCV0URN/6Pn9ZFlrtIWaU/vj/BfG";
```

Conclusion

Here, we discussed about using JWT as a token provider for spring boot security OAUTH2 implementation. In the next article we will be discussing about consuming this token in an angular 5 application.You can download the source from here.

If You Appreciate This, You Can Consider:

Donate

Like us at: or follow us at

Share this article on social media or with your teammates.

About The Author

A technology savvy professional with an exceptional capacity to analyze, solve problems and multi-task. Technical expertise in highly scalable distributed systems, self-healing systems, and service-oriented architecture. Technical Skills: Java/J2EE, Spring, Hibernate, Reactive Programming, Microservices, Hystrix, Rest APIs, Java 8, Kafka, Kibana, Elasticsearch, etc.

Further Reading on Spring Security

3. Angular Jwt Authentication

4. Spring Boot Jwt Auth

5. Spring Boot Security Hibernate Login Example

6. Spring Jms Activemq Integration Example

7. Spring Boot Security Oauth2 Example

8. React Js Jwt Authentication Example

ALSO ON **DEVGLAN**

Spring Boot Angular 8 Example

3 years ago • 11 comments

In this article, we will develop a full stack app using Spring Boot and …

Spring Boot MongoDB CRUD Example

3 years ago • 2 comments

In this tutorial, we will integrate MongoDB with a spring boot application …

Spring Boot React JS CRUD Example

3 years ago • 3 comments

This tutorial is about creating a full-stack app using Spring Boot and …

51 Comments **Devglan** **Disqus' Privacy Policy**

♡ **Favorite** 1 Tweet Share S

Join the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

9 ⌃ | ⌄ • Reply • Share ›

Dhiraj Ray ➦ Leandro Reis • 3 years ago
Source code download link in the conclusion section at the end
⌃ | ⌄ 1 • Reply • Share ›

Tharindu • 3 years ago
Hi, i want some advice from you. how to separate resource server and authorization se
so that i can use many resource servers and all of them will be authenticated by single
authorization server. where will be the code changed of this? thank you.
1 ⌃ | ⌄ • Reply • Share ›

Tharindu • 3 years ago • edited
Hi. your code is working fine. thank you for sharing knowledge. i have a minor questior
JWT. when i authenticate this will respond with access and refresh tokens fine. and i ca
to authorize. but the problem is once i re-authenticate myself, the previous the refresh
is still valid. so there can be many tokens for me, is it a common behavior of JWT or ar
something?
1 ⌃ | ⌄ • Reply • Share ›

Dhiraj Ray ➦ Tharindu • 3 years ago
Well this is a common behaviour but you must handle these scenarios in your a
invalidate previous user token once user logs in again or logout.
1 ⌃ | ⌄ • Reply • Share ›

Tharindu ➦ Dhiraj Ray • 3 years ago
Thank you.
1 ⌃ | ⌄ • Reply • Share ›

Kets Sap • 4 years ago
I am trying to generate the token from another ui project through javascript request. Bu
getting following error:
OPTIONS http://localhost:8080/oauth/token 401 ()
Failed to load http://localhost:8080/oauth/token: Response for preflight does not have I
status.

Can you please tell me what I need to change?
1 ⌃ | ⌄ • Reply • Share ›

feelposter mail ➦ Kets Sap • 4 years ago
.antMatchers(HttpMethod.OPTIONS).permitAll()
⌃ | ⌄ • Reply • Share ›

Sonali • a year ago
Hi ,
I got the access token. but get request is not working http://localhost:8080/Users/user?
access_token=eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJleHAiOjE2MjE3NjY0OTQ
⌃ | ⌄ • Reply • Share ›

Hanumanth jalsa • 2 years ago
What should I type in password and grant type in the postman??
⌃ | ⌄ • Reply • Share ›

man ticha • 2 years ago
how can i includes other properties or fields in my access token like Age or Salary
⌃ | ⌄ • Reply • Share ›

mbeddedsoft • 3 years ago
Hello, Thank you for the article. Do you have an example or can you tell me how I can
separate ResourceServer and AuthorizationServer? I know I need to identify my auth s
the resource server but there is something else I'm sure I need to do to all the Resourc
authenticate and authorize API requests. Can you help?
⌃ | ⌄ • Reply • Share ›

Pradyumna vinjamuri • 3 years ago • edited
Hi Dhiraj i tried several spring boot security examples and after some time i created no
spring boot application but i am unable to hit the rest end points i don't know why but ta

Dhiraj Ray Mod ➦ Pradyumna vinjamuri • 3 years ago

can u share yor code in github. I can take a look into if u r missing somethiong

^ | ˅ • Reply • Share ›

Pradyumna vinjamuri ➦ Dhiraj Ray • 3 years ago

Got it by removing security jar file dhiraj
Thanks for the reply

^ | ˅ • Reply • Share ›

HAN • 3 years ago

how can i place access_token in header instead of in parameter ?

^ | ˅ • Reply • Share ›

Dhiraj Ray Mod ➦ HAN • 3 years ago

Can you try with Authorization as the key

^ | ˅ • Reply • Share ›

vishnu dixit • 4 years ago

i want to create token only once having some validity (6 months) and share it with the c
token should be store in my(token provider) database. After the expires token, token p
only renew the token.you can use any other security feature of java

^ | ˅ • Reply • Share ›

paul • 4 years ago

Hi I got an error

When i run the spring booot application wiht authorization header as username : devgl

and passowrd: devglan-secret

body-> username:"Alex123"

password->"password"

grant_type->"password"

Im getting this error

{
"error": "invalid_grant",
"error_description": "Bad credentials"
}

The password in the database for the user is
"$2a$04$I9Q2sDc4QGGg5WNTLmsz0.fvGv3OjoZyj81PrSFyGOqMphqfS2qKu"

so i changed the password in the body part to the above

and again the same error

WHat should i do to make this application work

^ | ˅ • Reply • Share ›

Dhiraj Ray ➦ paul • 4 years ago

It should work in the manner in the shown in example. Please recheck your pas
else generate a new one with this tool - https://www.devglan.com/onl.... Also, if
spring boot 2, then check this section - Spring Boot 2 OAUTH2 because things
changed since spring boot 2.0 release.

^ | ˅ • Reply • Share ›

paul ➦ Dhiraj Ray • 3 years ago • edited

There is a post method to save the user which recieves the User Model
parameter.
I tried the post request. It is giving me error.
How can I save a user with username and password and other details li
and age etc.

Without the tool how can i generate the password using java

^ | ˅ • Reply • Share ›

"devglan-client" and "devglan-secret" to the server? Would this be in the header? Thar

^ | ⌄ • Reply • Share ›

Dhiraj Ray ➦ Eric Huang • 3 years ago

https://www.devglan.com/spr...

^ | ⌄ • Reply • Share ›

Dhiraj Ray ➦ Eric Huang • 4 years ago

Hello Eric,
That article is not posted yet. Actually, "devglan-client" and "devglan-secret" sh
passes in the header as basic authentication.

^ | ⌄ • Reply • Share ›

Eric Huang ➦ Dhiraj Ray • 4 years ago

Thanks for the quick response... this is my angular code but I am getting
400

```
private static readonly CLIENT_ID = "security-client";
private static readonly SECRET = "security-secret";

oAuthAuthenticated(credentials){

const headers = new HttpHeaders(
{
'Authorization' : 'Basic ' + btoa(AppService.CLIENT_ID + ':' + AppServic
'Content-type' : 'application/x-www-form-urlencoded; charset=utf-8'
}
)

this.httpClient.post("/oauth/token", credentials, {headers: headers})
.pipe(
catchError(this.handleError.bind(this))
)
.subscribe(response => {
console.log(response);
});

}
```

Am I setting up my header correctly? thanks

^ | ⌄ • Reply • Share ›

Aristio Dwi Nusa • 4 years ago

how to create new class i cant access requestmapping .
for example : localhost:8080/users/admin
this response for server is :

```
{
"timestamp": 1538731961789,
"status": 500,
"error": "Internal Server Error",
"exception": "java.lang.NullPointerException",
"message": "No message available",
"path": "/users/admin"
}
```

i duplicate file from user class to admin class.

^ | ⌄ • Reply • Share ›

Dhiraj Ray ➦ Aristio Dwi Nusa • 4 years ago

It's a NullPointer. Can you please check your logs

^ | ⌄ • Reply • Share ›

Mordor1989 • 4 years ago

How can I use a claims? Because I have a small problem. When I try this:
"SecurityContextHolder.getContext().authentication.principal as CurrentUser" CurrentU
customUserDetails class
I do not have information from token. only login.

^ | ⌄ • Reply • Share ›

Mordor1989 ➦ Mordor1989 • 4 years ago

Mordor1989 • 4 years ago

After restarting the AWS machine, all tokens are inactive, is the normal behavior?

Reply • Share ›

Dhiraj Ray ➦ Mordor1989 • 4 years ago

No this should not be a normal behaviour as we are manually ceating JWT toke custom expiry time and it is not related to a state

1 • Reply • Share ›

namviet • 4 years ago • edited

```
{
"timestamp": "2018-06-25T08:55:49.150+0000",
"status": 401,
"error": "Unauthorized",
"message": "Unauthorized",
"path": "/oauth/token"
}
```

do not work. do you have idea why

Reply • Share ›

Madhurima ➦ namviet • a year ago

Same error,what is the fix

Reply • Share ›

HARSHAL THAKRE • 4 years ago

Getting the following error

Caused by: org.springframework.beans.factory.BeanCreationException: Error creating name 'accessTokenConverter' defined in class path resource

Reply • Share ›

abdessalem kchaou • 4 years ago

Nice article.

I thinks it's better to store the source code in Github Repo. So that, we can try it immec

Thanks

Reply • Share ›

Dhiraj Ray Mod ➦ abdessalem kchaou • 4 years ago

Check the conclusion section.Its there

Reply • Share ›

abdessalem kchaou ➦ Dhiraj Ray • 4 years ago

Both : http://localhost:8080/signup and http://localhost:8080/token dont \
could you help me please

Reply • Share ›

abdessalem kchaou ➦ abdessalem kchaou • 4 years ago

could you send me an email to discuss about this project ?

I have some questions

Reply • Share ›

Vesko Vujovic ➦ Dhiraj Ray • 4 years ago

"status": 401,
"error": "Unauthorized",
"message": "Unauthorized",
"path": "/oauth/token"

I always have this error. do you have idea why

^ | ˅ • Reply • Share ›

Dhiraj Ray ➦ Vesko Vujovic • 4 years ago

Check if u ve correctly executed insert statements. If yes then de
userdetailserviceimpl n check if correct user is fetched from db

^ | ˅ • Reply • Share ›

Vesko Vujovic ➦ Dhiraj Ray • 4 years ago • edited

Yes everything is ok. It looks like userDetailsService is never call

Do you have any advice how to debug SPring security?

All the time i get 401, and i've done everything like on your tutoria

^ | ˅ • Reply • Share ›

Dhiraj Ray ➦ Vesko Vujovic • 4 years ago

What is the spring boot version. Is it 2.0?

^ | ˅ • Reply • Share ›

Vesko Vujovic ➦ Dhiraj Ray • 4 years ago

Yes 2.0

^ | ˅ • Reply • Share ›

Dhiraj Ray ➦ Vesko Vujovic • 4 years ago

Above code sample is not working for Spring boot 2.If u debug
userdetailserviceimple, the usrname is coming as devglan-client
Alex123. N we don't have entries in DB for devglan-client. Hence
I am not sure why this is happening in spring boot 2. Need to exp

^ | ˅ • Reply • Share ›

Vishal Raut ➦ Dhiraj Ray • 4 years ago

You'll need to encode the CLIENT_SECRET in the
AuthorizationServerConfig to make it work with Spring Boot 2, us
same BCryptPasswordEncoder used in SecurityConfig.

.secret(new BCryptPasswordEncoder().encode(CLIENT_SECRE

^ | ˅ • Reply • Share ›

Dhiraj Ray ➦ Vishal Raut • 4 years ago

Yes and I have updated the article under the section - Spring Boo
OAUTH2

^ | ˅ • Reply • Share ›

Teja MVSR ➦ Dhiraj Ray • a year ago

I have used devglan-client for client secret instead of bcrypt pass
spring boot 2.4.2 and its working fine.

^ | ˅ • Reply • Share ›

HARSHAL THAKRE ➦ Dhiraj Ray • 4 years ago

kindly share the link

^ | ˅ • Reply • Share ›

Dhiraj Ray Mod ➦ HARSHAL THAKRE • 4 years ago

http://www.devglan.com/spri...

^ | ˅ • Reply • Share ›

Load more comments

<devglan.com/>
DEVELOPING DEVELOPERS

Devglan is one stop platform for all programming tutorials and courses.

Contact Us

Dhiraj
dhiraj@devglan.com