SpringBoot Security - Part8 (OAuth Authentication)

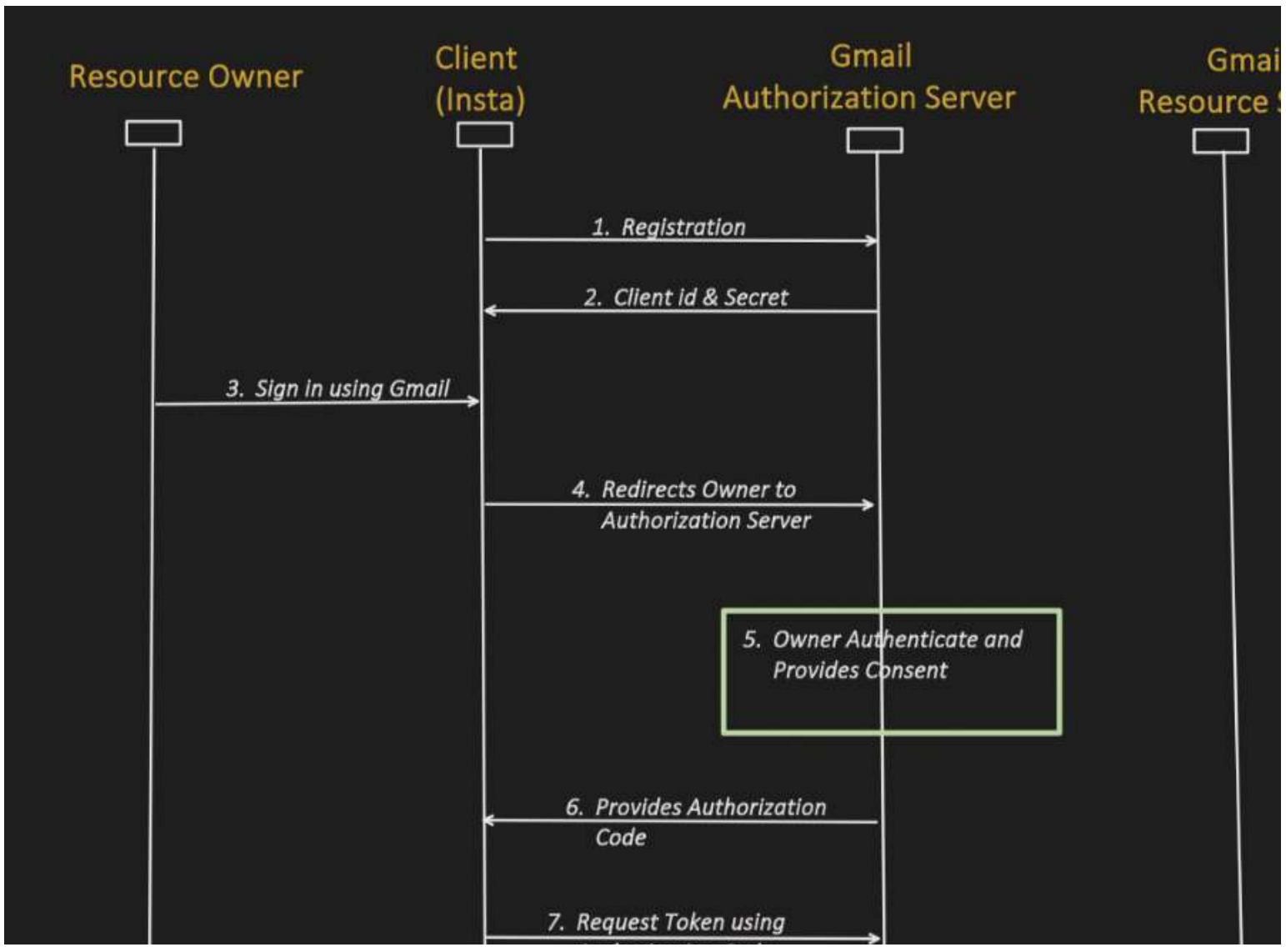
Sunday, 13 April 2025 10:18 AM

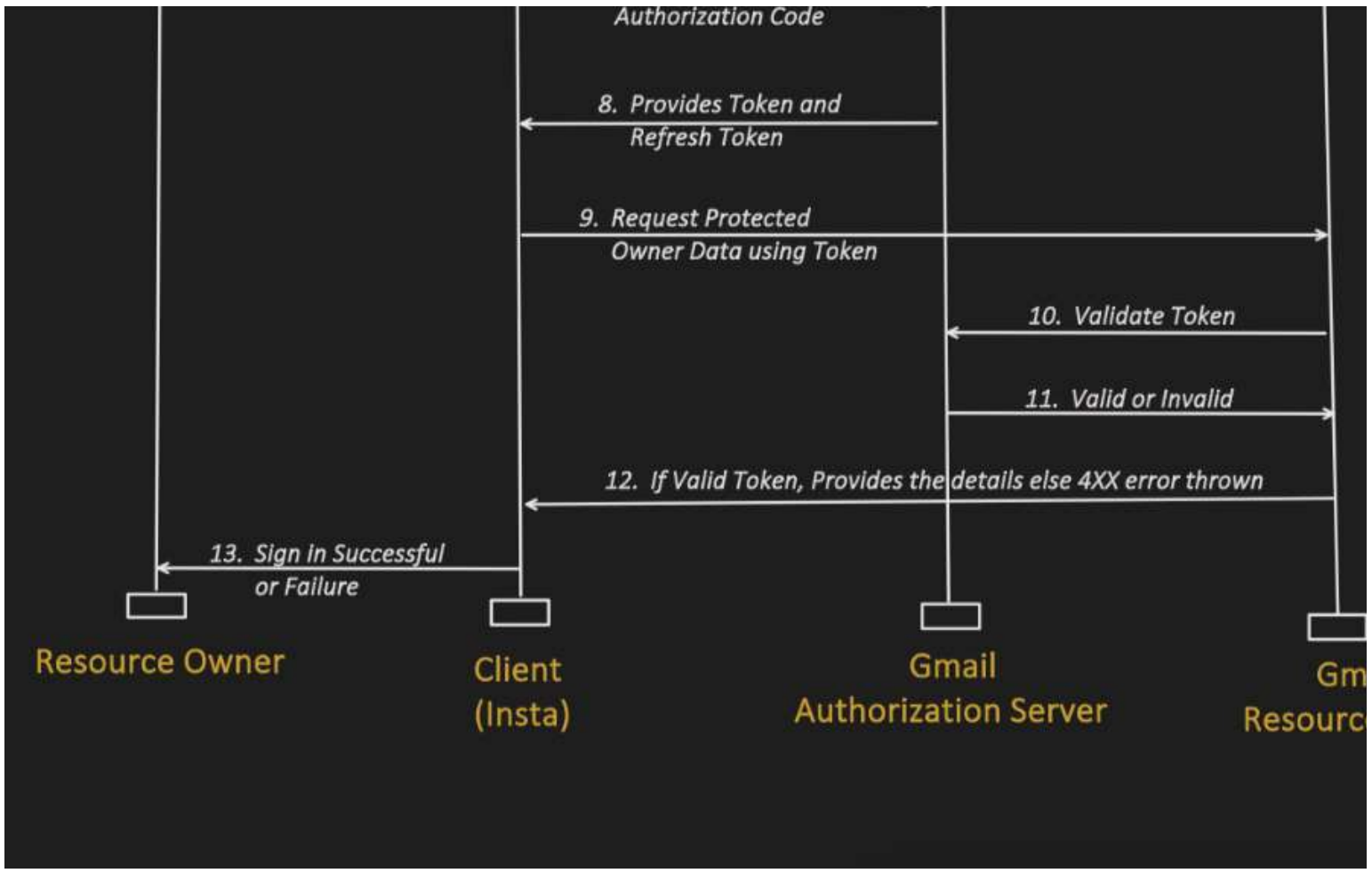
As explained in previous video:

- What is OAuth:
  It's an Open Authorization framework, enables secure third party access to user protecte data.

- And its different Grant Types like
  - *Authorization Code Grant*,
  - Implicit Grant,
  - Client Credentials Grant etc....

Quick Recap of "Authorization Code Grant" flow:





So, before we proceed with User Authentication implementation using OAuth framewor

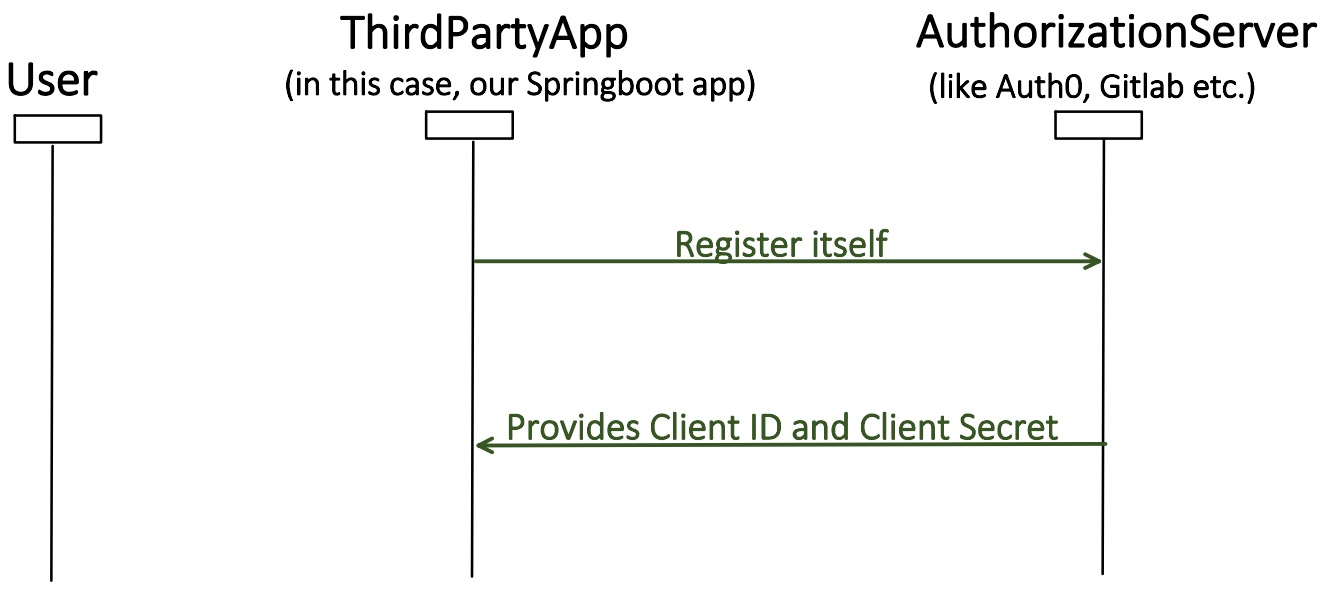
Lets understand difference between OAUTH and OIDC

S.No	Title	OAuth2	OIDC
1.	Full Form	Open Authorization	OpenID Connect
2.	Purpose Used for	Authorization • **Grant** secure third party access to **user protected data** like third party app showing my google calendar data	Authentication - Layer built on top of OAuth2 ar party app to **verify the identity** c Authorization serve
3.	Token generated	Access token (Opaque or JWT) - used by Third Party App, to call resource server API's to access user protected data. - Many times, these access token could be Opaque, means only Authorization server interpret and validate them.	ID_Token (JWT) + Access - ID_Token is a JWT token, whic Third party app, and can be used user. *(this JWT token contains minim required for Authentic* - Also has Access token, we car server API's to access more user r required.
4.	Scope	read/write: request access to read or write to resources. profile: request access to basic profile	openid (must) : it indicates that t requesting a ID TOKEN (to auther

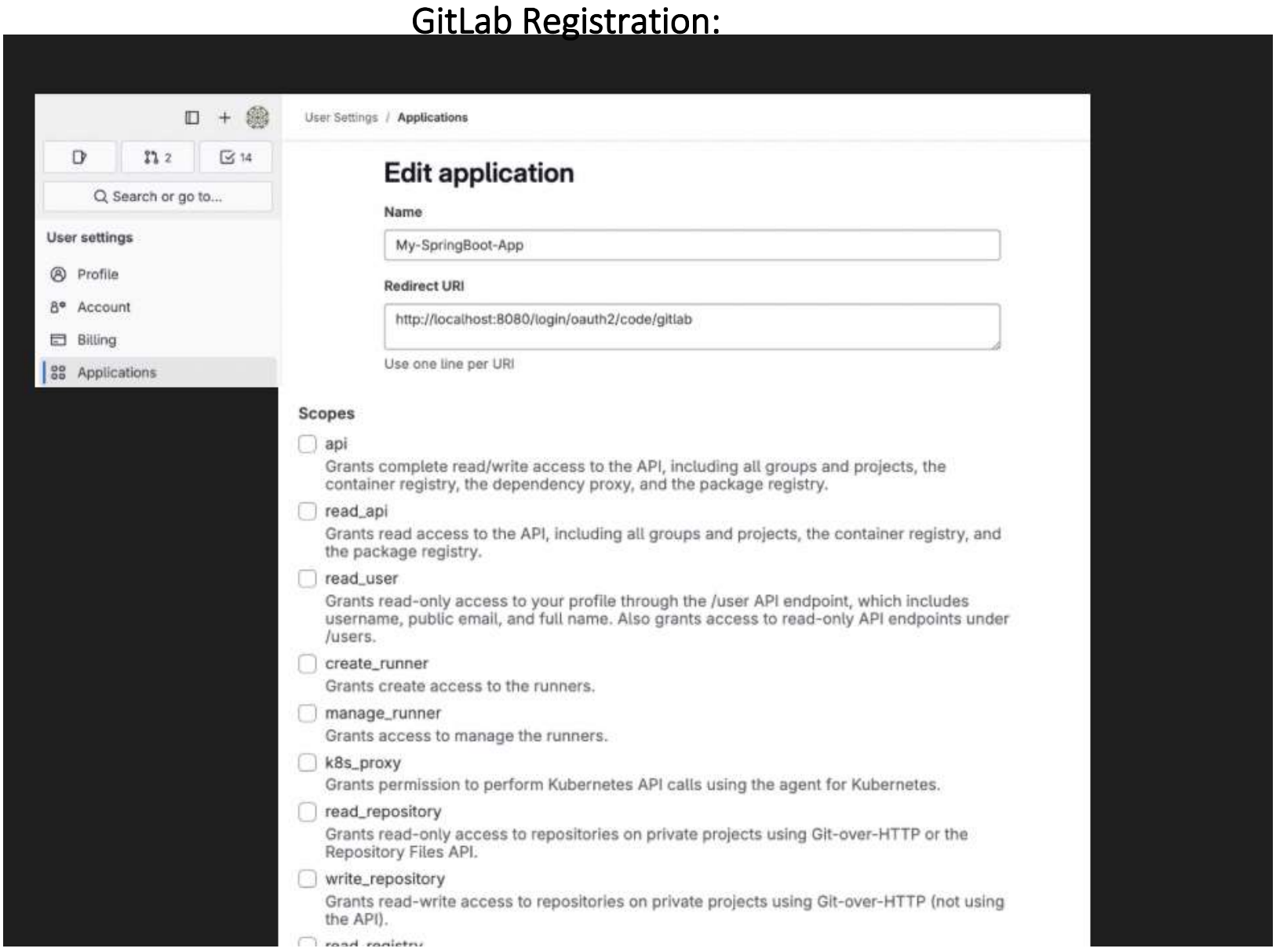
		info like name, profile picture etc. email: request access to user email address etc..	Now, this ID TOKEN (JWT) itself w minimal info which can be used user. But if required some additic use scope= openid, prof so this JWT now also has some pr too.

Step1:

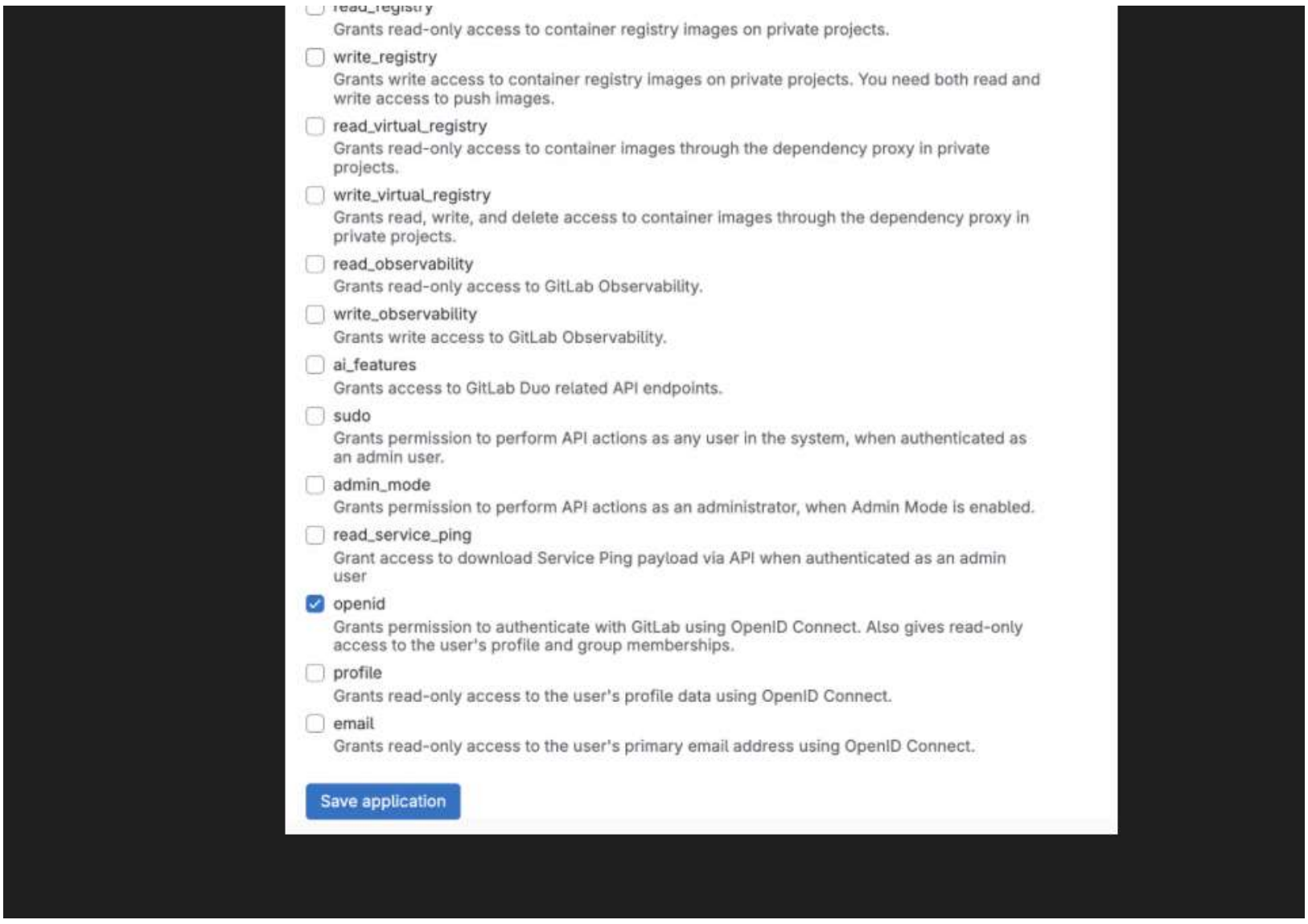
- Third party app registration to Authorization server.



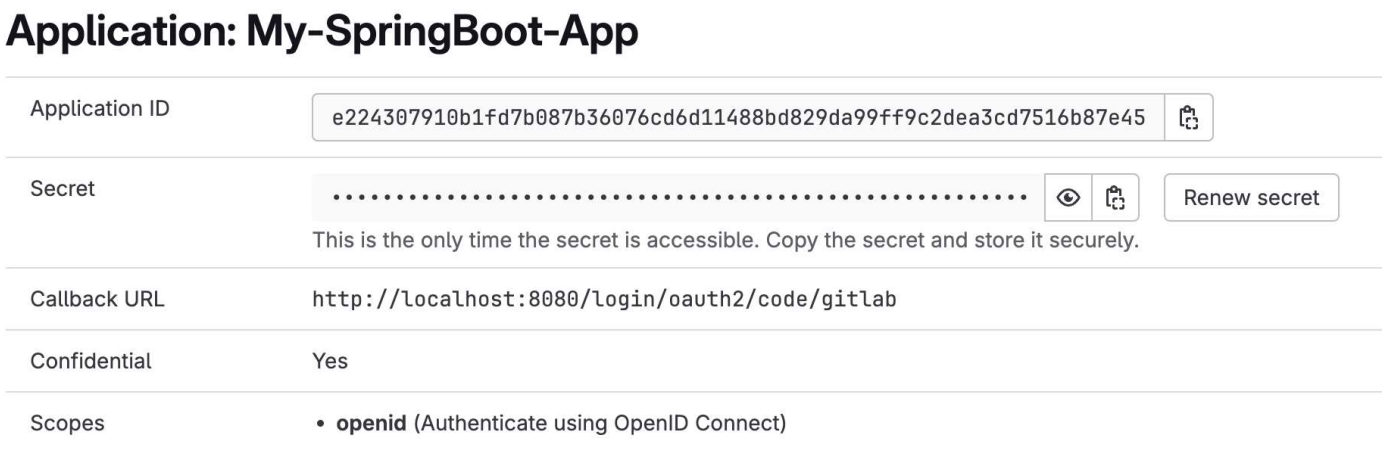
GitLab Registration:



Basic Ir



After registration, we get Client id and Secret



pom.xml

```
<dependency>
<groupId>org.springframework.boot</groupId>
<artifactId>spring-boot-starter-oauth2-
client</artifactId>
</dependency>
```

application.properties

```
#OAuth configurations
##Gitlab
spring.security.oauth2.client.registration.gitlab.client-id=e224307910b1fd7b087b36076cd6d11488bd829da99ff9c2de
```

```
spring.security.oauth2.client.registration.gitlab.client-secret=gloas-4d80d8c0b97e7807e36270769e5da47b2467af93
spring.security.oauth2.client.registration.gitlab.scope=openid
spring.security.oauth2.client.registration.gitlab.authorization-grant-type=authorization_code
spring.security.oauth2.client.registration.gitlab.redirect-uri=http://localhost:8080/login/oauth2/code/gitlab
spring.security.oauth2.client.provider.gitlab.authorization-uri=https://gitlab.com/oauth/authorize
spring.security.oauth2.client.provider.gitlab.token-uri=https://gitlab.com/oauth/token
spring.security.oauth2.client.provider.gitlab.issuer-uri=https://gitlab.com
spring.security.oauth2.client.provider.gitlab.jwk-set-uri=https://gitlab.com/oauth/discovery/keys

##Auth0
spring.security.oauth2.client.registration.auth0.client-id=pCDhGLi2bXTLlYb0PwPZGMVgFek6PiEx
spring.security.oauth2.client.registration.auth0.client-secret=crSWtyt6uBYJ_NJ1jbN8GX4mBULnYB622ejCnnzYxQ1JRyE
spring.security.oauth2.client.registration.auth0.scope=openid, profile
spring.security.oauth2.client.registration.auth0.authorization-grant-type=authorization_code
spring.security.oauth2.client.registration.auth0.redirect-uri=http://localhost:8080/login/oauth2/code/auth0
spring.security.oauth2.client.provider.auth0.authorization-uri=https://dev-g4t3m70i6jqdcl6i.us.auth0.com/authc
spring.security.oauth2.client.provider.auth0.token-uri=https://dev-g4t3m70i6jqdcl6i.us.auth0.com/oauth/token
spring.security.oauth2.client.provider.auth0.issuer-uri=https://dev-g4t3m70i6jqdcl6i.us.auth0.com/
spring.security.oauth2.client.provider.auth0.jwk-set-uri=https://dev-g4t3m70i6jqdcl6i.us.auth0.com/.well-known
```

SecurityConfig.java

```java
@Configuration
@EnableWebSecurity
public class SecurityConfig {

    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity http) throws
        http.authorizeHttpRequests(auth -> auth
                        .anyRequest().authenticated())
                .csrf(csrf -> csrf.disable())
                .oauth2Login(Customizer.withDefaults());
        return http.build();
    }
}
```

Basic Controller class, just for testing

```java
@RestController
public class UserDetailsController {

    @GetMapping("/")
    public String defaultHomePageMethod(){
        return "hello, you are logged in";
    }
    @GetMapping("/users")
    public String getUsersDetails(){
```

```
        return "fetched the details of successful
    }
}
```

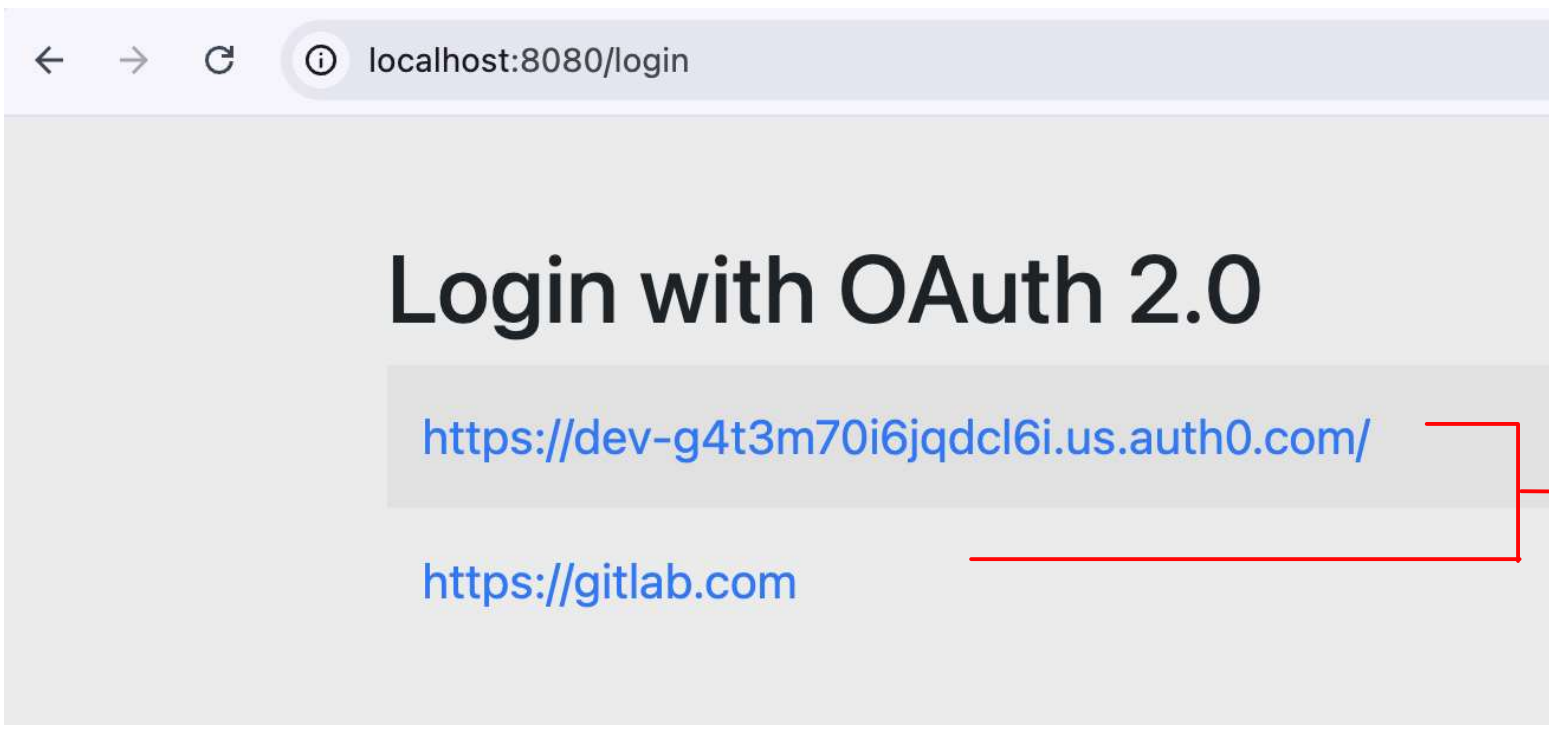
Let's try:

Notice one thing that, I have not created any User in our Springboot app.

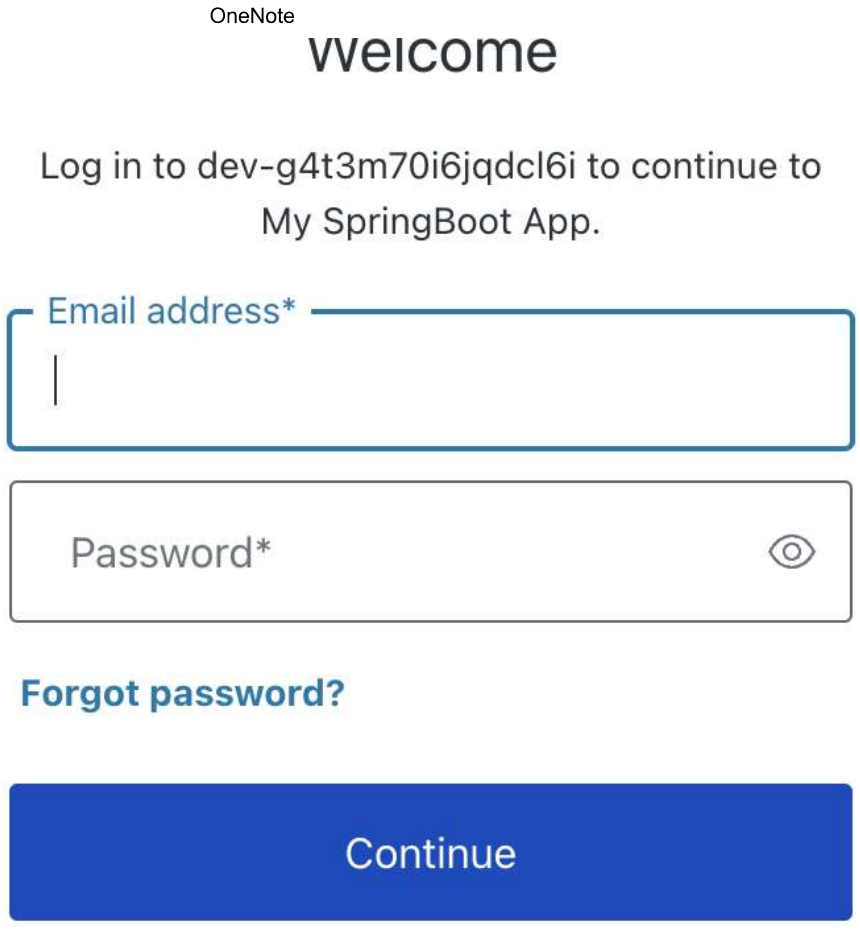
Started the application server :

```
SpringbootApplication
2025-04-13T20:00:23.955+05:30  INFO 28511 --- [          main] o.h.e.t.j.p.i.JtaPlatformInitiator       : HHH000489: No JTA platform availa
Hibernate:
    drop table if exists user_register cascade
Hibernate:
    create table user_register (
        id bigint generated by default as identity,
        password varchar(255) not null,
        role varchar(255),
        username varchar(255) not null unique,
        primary key (id)
    )
2025-04-13T20:00:23.963+05:30  INFO 28511 --- [          main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFact
2025-04-13T20:00:25.216+05:30  WARN 28511 --- [          main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enable
2025-04-13T20:00:25.264+05:30  INFO 28511 --- [          main] r$InitializeUserDetailsManagerConfigurer : Global AuthenticationManager conf
2025-04-13T20:00:25.515+05:30  INFO 28511 --- [          main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 8080 (http
2025-04-13T20:00:25.520+05:30  INFO 28511 --- [          main] c.c.l.SpringbootApplication              : Started SpringbootApplication in
2025-04-13T20:00:26.563+05:30  INFO 28511 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring DispatcherSer
2025-04-13T20:00:26.563+05:30  INFO 28511 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherS
2025-04-13T20:00:26.564+05:30  INFO 28511 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Completed initialization in 1 ms
```

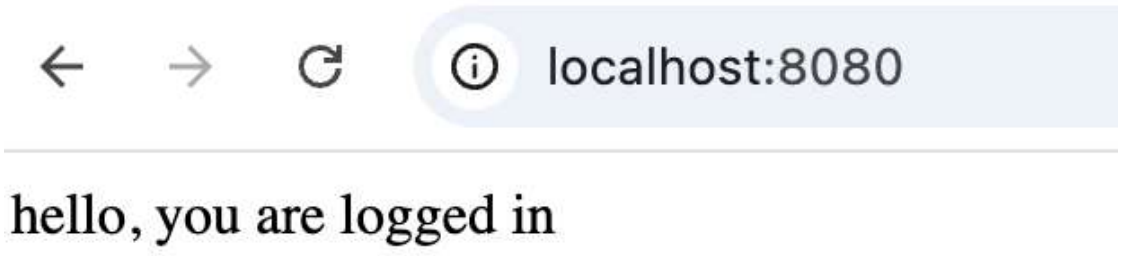
Typed the *localhost:8080* url, it takes us to /login endpoint



When I clicked the Auth0 authorization server, it redirects me to Auth0 login p



Once I provided the sign in at Auth0 and provided the consent, I am able to



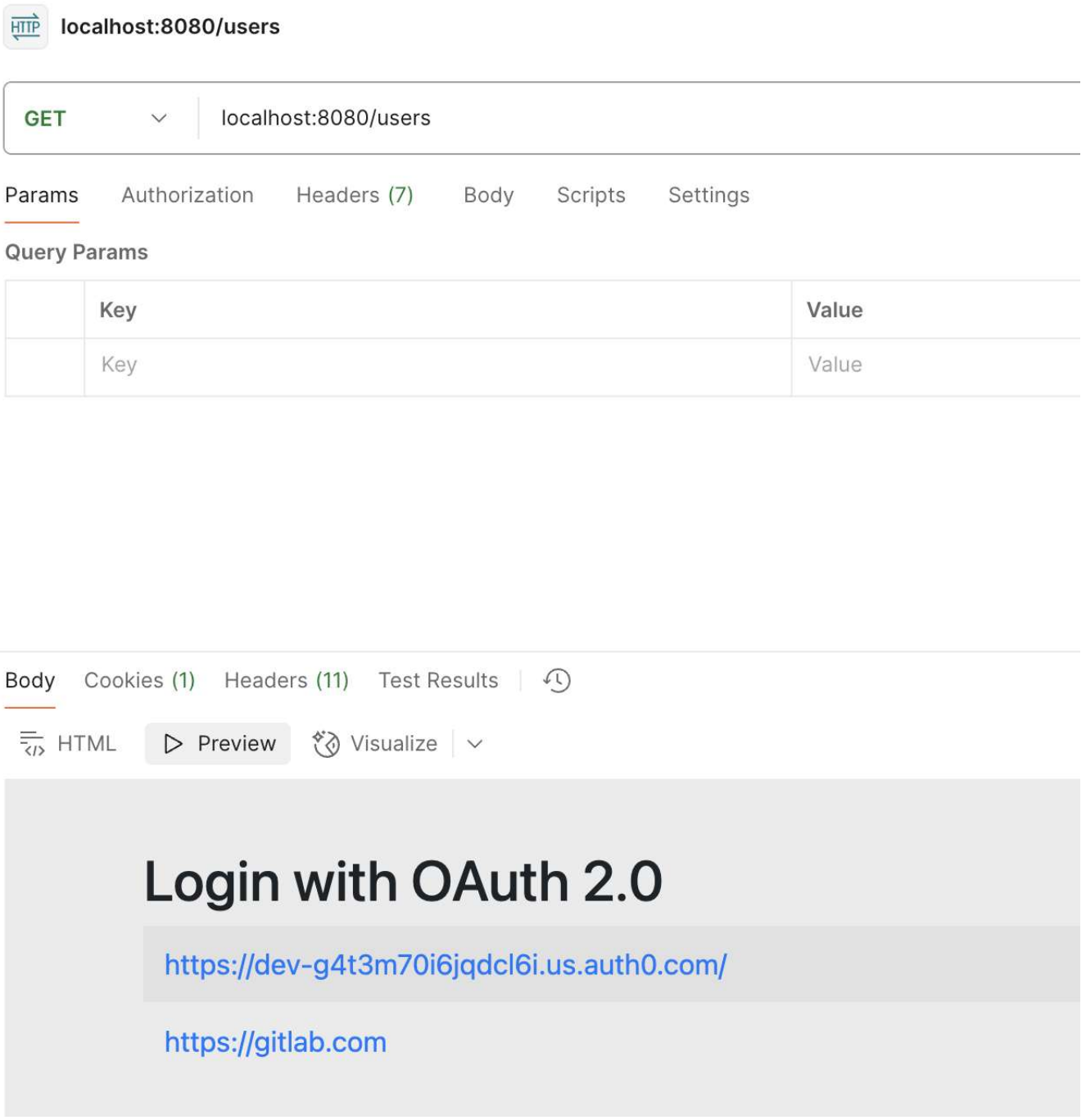
Now, if I try to access, any other API, I don't have to sign in again and access

What? With just pom.xml, application.properties and SecurityConfig.java
run the complete OAuth2 flow.

Answer is Yes, Springboot Security framework provides the compete functionality of OA don’t have to code anything.

But here is the twist:

When I tried to access the "/users" API through postman (not through browser), it takes



Because, Springboot assume that, Oauth2 login will be done on a browser, so by-default it

AuthorizationServer

ThirdPartyApp

User

(in this case, our Springboot app)

(like Auth0, Gitlab et

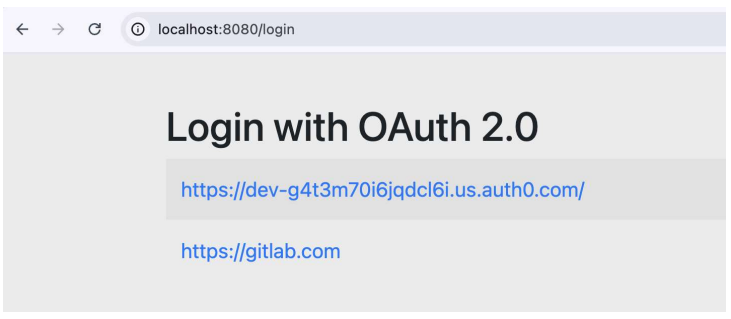
Register itself

Provides Client ID and Client Secret

localhost:8080/login

DefaultLoginPageGeneratingFilter.jav

```
if (this.oauth2LoginEnabled) {
    sb.append("<h2 class=\"form-signin-heading\">
    sb.append(createError(loginError, errorMsg));
    sb.append(createLogoutSuccess(logoutSuccess))
    sb.append("<table class=\"table table-striped
    for (Map.Entry<String, String> clientAuthenti
            .entrySet()) {
        sb.append(" <tr><td>");
        String url = clientAuthenticationUrlToCli
        sb.append("<a href=\"").append(contextPat
        String clientName = HtmlUtils.htmlEscape(
        sb.append(clientName);
        sb.append("</a>");
        sb.append("</td></tr>\n");
    }
    sb.append("</table>\n");
}
```



Showed the list of Authorization Server Supported

User click on one of the authorization server links, then /oauth2/authorization/{registration_id} will get invoked.
In this registration_id is either auth0 or gitlab, what we have configured in application.properties

localhost:8080 /oauth2/authorization /{registration_id}

```
#OAuth configurations
##Gitlab
spring.security.oauth2.client.registration.gitlab.client-id=e224
spring.security.oauth2.client.registration.gitlab.client-secret=
spring.security.oauth2.client.registration.gitlab.scope=openid
spring.security.oauth2.client.registration.gitlab.authorization-
spring.security.oauth2.client.registration.gitlab.redirect-uri=h
spring.security.oauth2.client.provider.gitlab.authorization-uri=
spring.security.oauth2.client.provider.gitlab.token-uri=https://
spring.security.oauth2.client.provider.gitlab.issuer-uri=https:/
spring.security.oauth2.client.provider.gitlab.jwk-set-uri=https:

##Auth0
spring.security.oauth2.client.registration.auth0.client-id=pCDhG
spring.security.oauth2.client.registration.auth0.client-secret=c
spring.security.oauth2.client.registration.auth0.scope=openid, p
spring.security.oauth2.client.registration.auth0.authorization-g
spring.security.oauth2.client.registration.auth0.redirect-uri=ht
spring.security.oauth2.client.provider.auth0.authorization-uri=h
spring.security.oauth2.client.provider.auth0.token-uri=https://d
spring.security.oauth2.client.provider.auth0.issuer-uri=https://
spring.security.oauth2.client.provider.auth0.jwk-set-uri=https:/
```

OAuth2AuthorizationRequestRedirectFilter.java

```
@Override
protected void doFilterInternal(HttpServletRequest request, HttpServletResponse response, FilterChain filterChain)
        throws ServletException, IOException {
    try {
        OAuth2AuthorizationRequest authorizationRequest = this.authorizationRequestResolver.resolve(request);
        if (authorizationRequest != null) {
            this.sendRedirectForAuthorization(request, response, authorizationRequest);
            return;
        }
    }
```

https://dev-g4t3m70i6jqdcl6i.us.auth0.com/authorize

or

https://gitlab.com/oauth/authorize

Invokes authorization-uri of the specific registration_id

http://localhost:8080/login/oauth2/code/gitlab

or

http://localhost:8080/login/oauth2/code/auth

Redirect API got invoked by authorization server, with Authorization Code present in the response

OAuth2LoginAuthenticationFilter.java

```
@Override
public Authentication attemptAuthentication(HttpServletRequest request, HttpServletResponse response)
        throws AuthenticationException {
    OAuth2AuthorizationResponse authorizationResponse = OAuth2AuthorizationResponseUtils.convert(params
            redirectUri);
    Object authenticationDetails = this.authenticationDetailsSource.buildDetails(request);
    OAuth2LoginAuthenticationToken authenticationRequest = new OAuth2LoginAuthenticationToken(clientReg
            new OAuth2AuthorizationExchange(authorizationRequest, authorizationResponse));
    authenticationRequest.setDetails(authenticationDetails);
    OAuth2LoginAuthenticationToken authenticationResult = (OAuth2LoginAuthenticationToken) this
            .getAuthenticationManager()
            .authenticate(authenticationRequest);
    OAuth2AuthenticationToken oauth2Authentication = this.authenticationResultConverter
            .convert(authenticationResult);
    Assert.notNull(oauth2Authentication, message: "authentication result cannot be null");
    oauth2Authentication.setDetails(authenticationDetails);
    OAuth2AuthorizedClient authorizedClient = new OAuth2AuthorizedClient(
            authenticationResult.getClientRegistration(), oauth2Authentication.getName(),
            authenticationResult.getAccessToken(), authenticationResult.getRefreshToken());

    this.authorizedClientRepository.saveAuthorizedClient(authorizedClient, oauth2Authentication, reques
    return oauth2Authentication;
}
```

OidcAuthorizationCodeAuthenticationProvider.*java*

```
@Override
public Authentication authenticate(Authentication authentication) throws AuthenticationException {
    OAuth2LoginAuthenticationToken authorizationCodeAuthentication = (OAuth2LoginAuthenticationToke
    // Section 3.1.2.1 Authentication Request -
    // https://openid.net/specs/openid-connect-core-1_0.html#AuthRequest
    // scope
    // REQUIRED. OpenID Connect requests MUST contain the "openid" scope value.
```

```
if (!authorizationCodeAuthentication.getAuthorizationExchange() OAuth2AuthorizationExchange
        .getAuthorizationRequest() OAuth2AuthorizationRequest
        .getScopes() Set<String>
        .contains(OidcScopes.OPENID)) {
    // This is NOT an OpenID Connect Authentication Request so return null
    // and let OAuth2LoginAuthenticationProvider handle it instead
    return null;
}
OAuth2AuthorizationRequest authorizationRequest = authorizationCodeAuthentication.getAuthorizati
        .getAuthorizationRequest();
OAuth2AuthorizationResponse authorizationResponse = authorizationCodeAuthentication.getAuthoriza
        .getAuthorizationResponse();
if (authorizationResponse.statusError()) {
    throw new OAuth2AuthenticationException(authorizationResponse.getError(),
            authorizationResponse.getError().toString());
}
if (!authorizationResponse.getState().equals(authorizationRequest.getState())) {
    OAuth2Error oauth2Error = new OAuth2Error(INVALID_STATE_PARAMETER_ERROR_CODE);
    throw new OAuth2AuthenticationException(oauth2Error, oauth2Error.toString());
}
OAuth2AccessTokenResponse accessTokenResponse = getResponse(authorizationCodeAuthentication);
ClientRegistration clientRegistration = authorizationCodeAuthentication.getClientRegistration()
Map<String, Object> additionalParameters = accessTokenResponse.getAdditionalParameters();
if (!additionalParameters.containsKey(OidcParameterNames.ID_TOKEN)) {
    OAuth2Error invalidIdTokenError = new OAuth2Error(INVALID_ID_TOKEN_ERROR_CODE,
            description: "Missing (required) ID Token in Token Response for Client Registration: "
                    + clientRegistration.getRegistrationId(),
            uri: null);
    throw new OAuth2AuthenticationException(invalidIdTokenError, invalidIdTokenError.toString())
}
OidcIdToken idToken = createOidcToken(clientRegistration, accessTokenResponse);
```

https://dev-g4t3m70i6jqdcl6i.us.auth0.com/oauth/token

or

https://gitlab.com/oauth/token

Returns
Access Token and ID_TOKEN

Now, OAuth2LoginAuthenticationFilter.java stores this tokens and creates a HTTPSession

Login Successful home page is called with Session id in cookie

localhost

Request URL: http://localhost:8080/

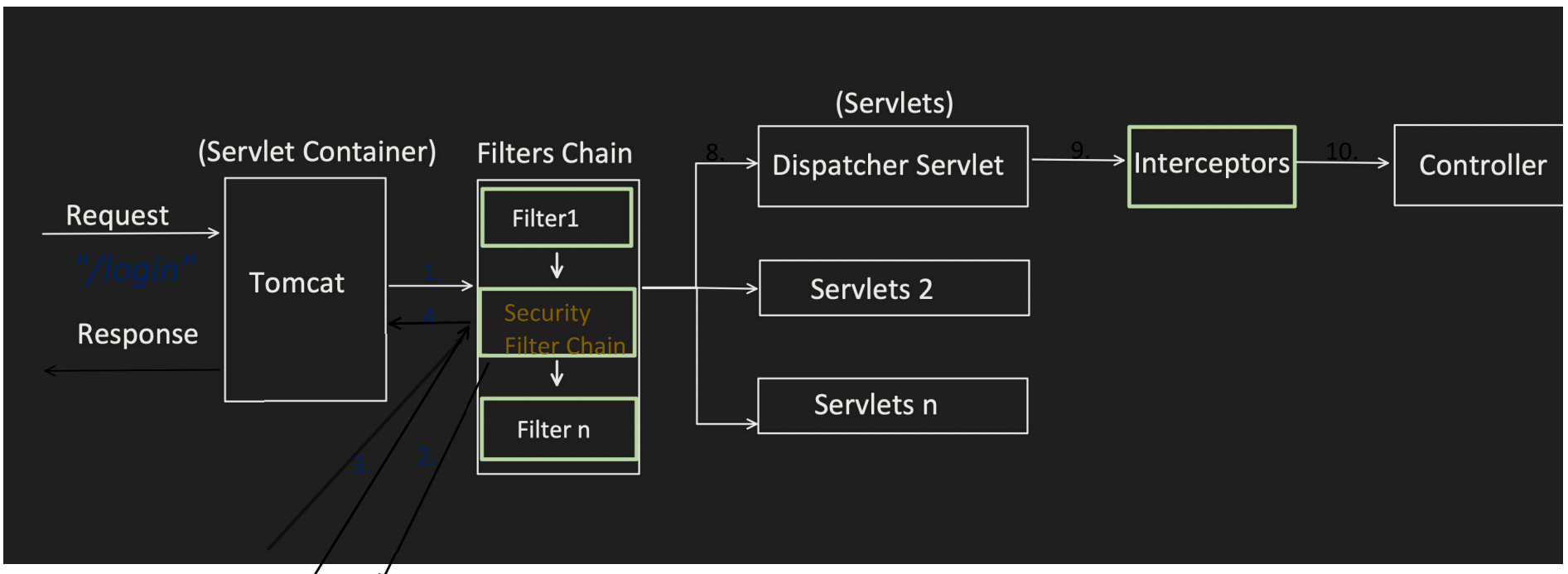
Request Method:	GET
Status Code:	200 OK
Remote Address:	[::1]:8080
Referrer Policy:	strict-origin-when-cross-origin

▼Response Headers ☐ Raw

Cache-Control:	no-cache, no-store, max-age=0, must-revalidate
Connection:	keep-alive
Content-Length:	40
Content-Type:	text/html;charset=UTF-8
Date:	Mon, 14 Apr 2025 10:17:43 GMT
Expires:	0
Keep-Alive:	timeout=60
Pragma:	no-cache
X-Content-Type-Options:	nosniff
X-Frame-Options:	DENY
X-Xss-Protection:	0

▼Request Headers ☐ Raw

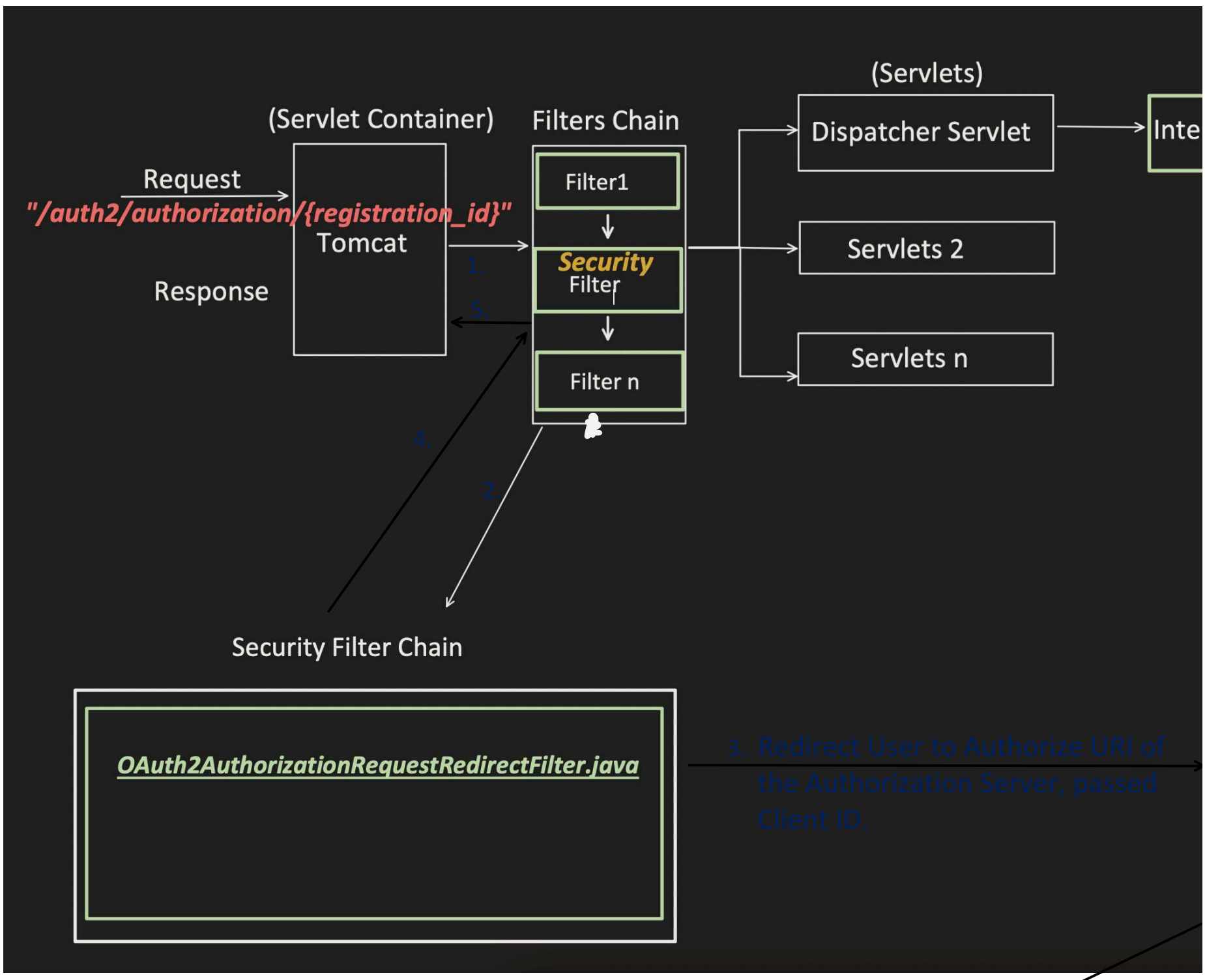
Accept:	text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7
Accept-Encoding:	gzip, deflate, br, zstd
Accept-Language:	en-GB,en-US;q=0.9,en;q=0.8
Connection:	keep-alive
Cookie:	SESSION=ZWU2YjNhNGEtNWExOC00NDE5LTk5ZTUtOTZhOTk0ZjA2ZmNi



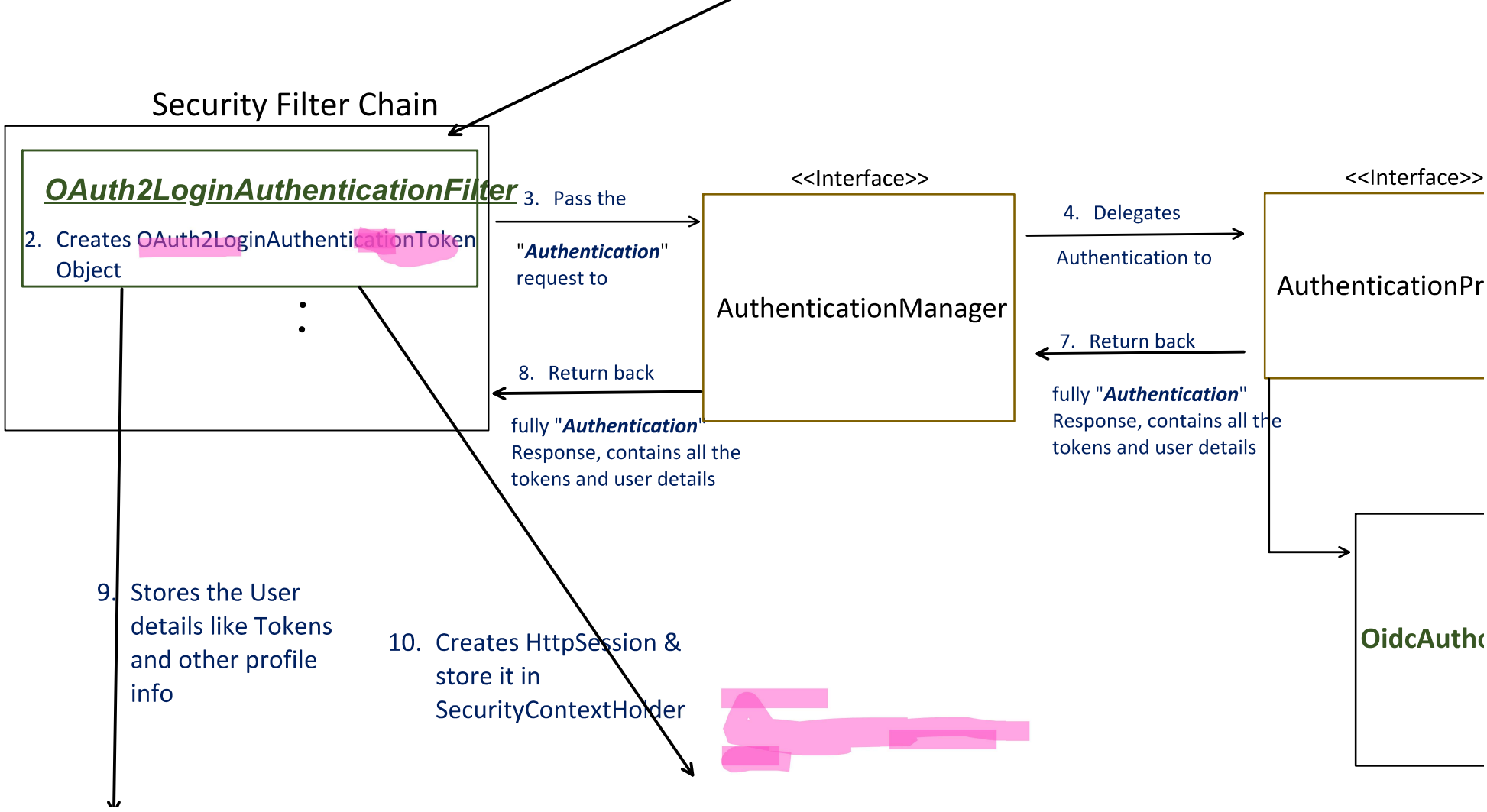
Security Filter Chain

DefaultLoginPageGeneratingFilter

(returns the list of authorization servers supported)



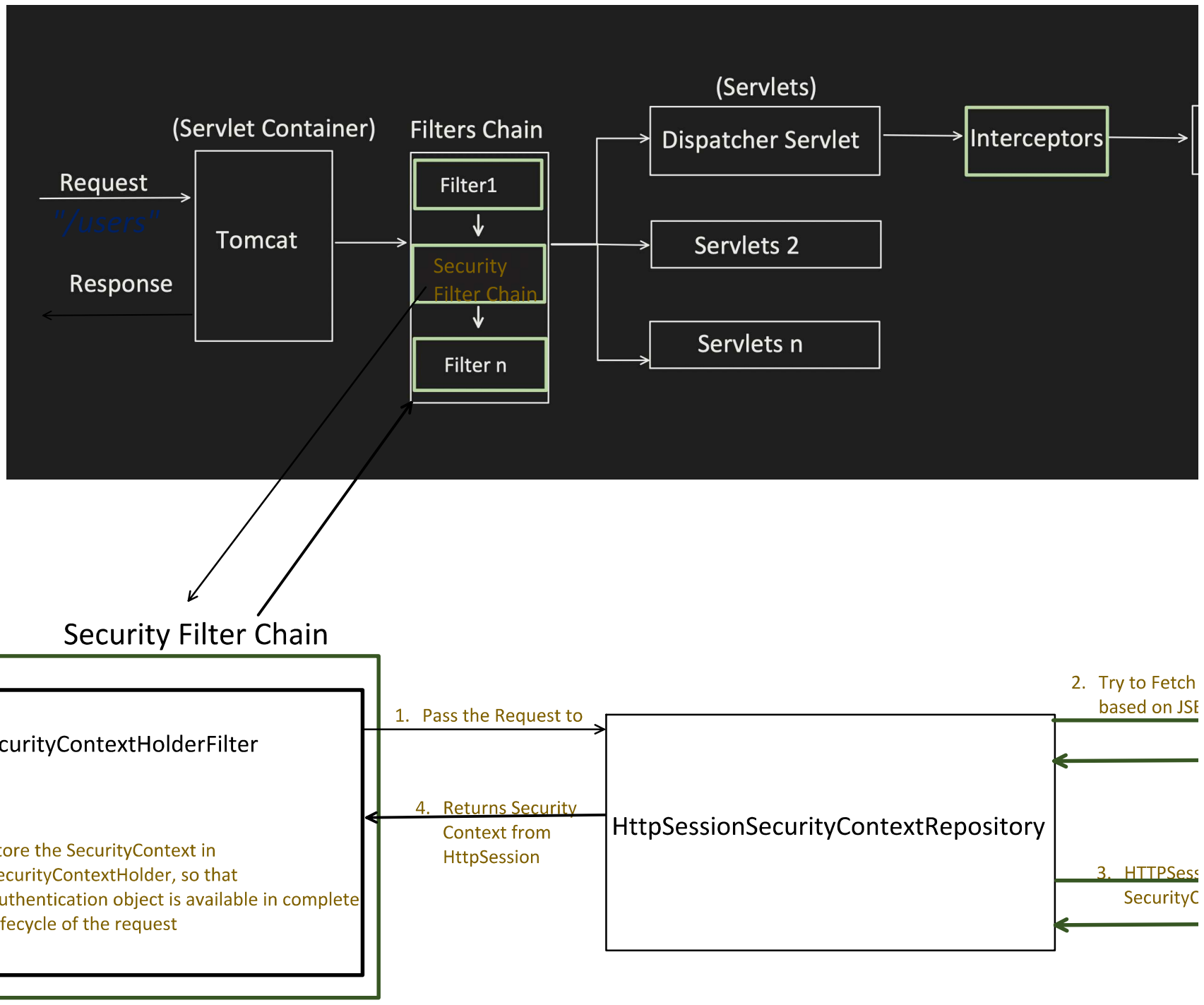
1. Once User Auth
Server calls the
Authorization C



InMemoryOAuth2AuthorizedClientService

SecurityContextHolder

SeucurityContext object holds Authentication Object

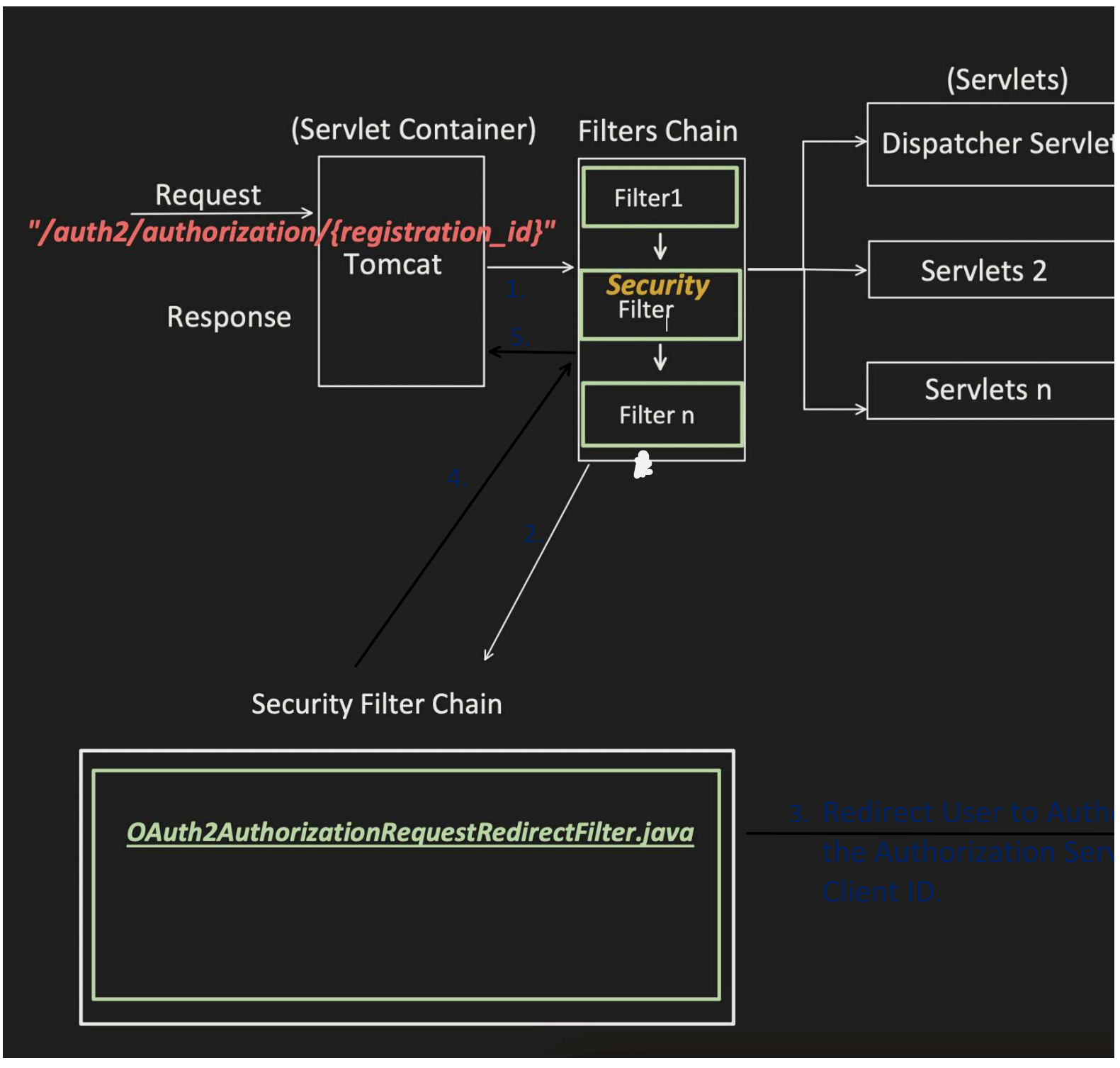


- That's why, when we hit the request from Postman, it again asked for Login, because Ses
- Also, when session is set, it might not validate the token with each request, till Session is scenario that ID Token becomes invalidated but Session is still active.

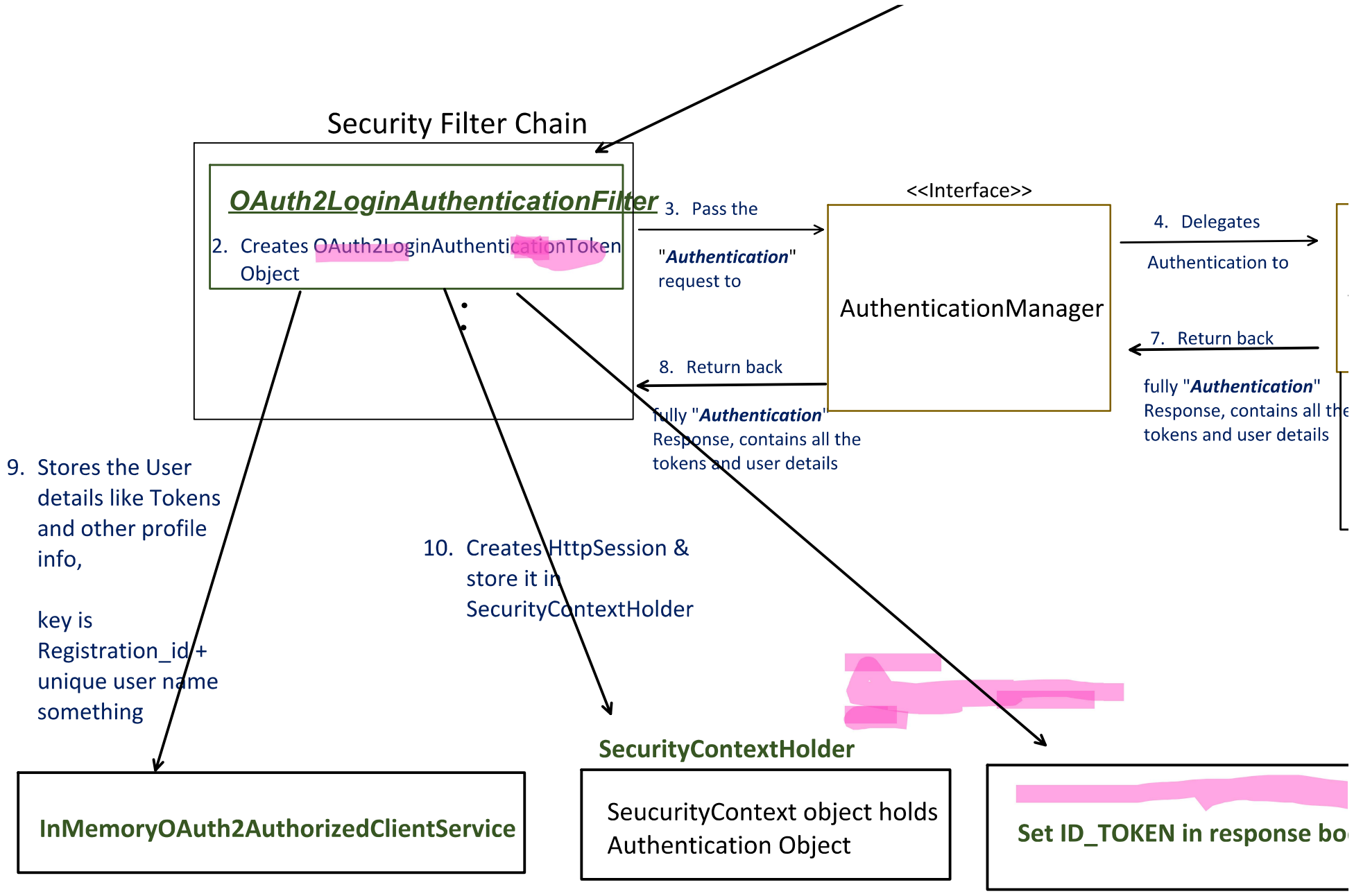
  - And this also makes OAuth Stateful.

So, how to fix this ?

- Lets make it STATELESS
- And return the ID_TOKEN in the response
- With each request, client will pass the Token and we will validate the token.



1. C
S
A



OAuth2LoginAuthenticationFilter parent class has one method "onAuthenticationSuccess(process completes, I have overwrite that method and set the ID_TOKEN in the response b

```
@Component
public class CustomOAuth2SuccessHandler implements AuthenticationSuccessHandler {

    private final OAuth2AuthorizedClientService clientService;

    @Autowired
    public CustomOAuth2SuccessHandler(OAuth2AuthorizedClientService clientServic
        this.clientService = clientService;
    }
    @Override
    public void onAuthenticationSuccess(HttpServletRequest request, HttpServletR
                                        Authentication authentication) throws IO
        OAuth2AuthenticationToken authToken = (OAuth2AuthenticationToken) authen
        OAuth2AuthorizedClient client = clientService.loadAuthorizedClient(
                authToken.getAuthorizedClientRegistrationId(), authToken.getName

        if (client != null) {
```

```
        String idToken = null;
        if (authToken.getPrincipal() instanceof OidcUser) {
            OidcUser oidcUser = (OidcUser) authToken.getPrincipal();
            idToken = oidcUser.getIdToken().getTokenValue();
        }

        // Send the access token in the response (JSON)
        response.setContentType("application/json");
        response.getWriter().write( s: "{ \"id_token\": \"" + idToken + "\" }
        response.getWriter().flush();
    } else {
        response.sendError(HttpServletResponse.SC_UNAUTHORIZED,  s: "Authoriz
    }
}
}
```

Start the application

```
Hibernate:
    drop table if exists user_register cascade
Hibernate:
    create table user_register (
        id bigint generated by default as identity,
        password varchar(255) not null,
        role varchar(255),
        username varchar(255) not null unique,
        primary key (id)
    )
2025-04-14T18:40:51.827+05:30  INFO 31908 --- [           main] j.LocalContainerEntityManagerFactoryB
2025-04-14T18:40:53.200+05:30  WARN 31908 --- [           main] JpaBaseConfiguration$JpaWebConfigurat
2025-04-14T18:40:53.242+05:30  INFO 31908 --- [           main] r$InitializeUserDetailsManagerConfigu
2025-04-14T18:40:53.487+05:30  INFO 31908 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServ
2025-04-14T18:40:53.492+05:30  INFO 31908 --- [           main] c.c.l.SpringbootApplication
2025-04-14T18:40:57.746+05:30  INFO 31908 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]
2025-04-14T18:40:57.746+05:30  INFO 31908 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet
2025-04-14T18:40:57.747+05:30  INFO 31908 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet
```

After Authorizing from the Author

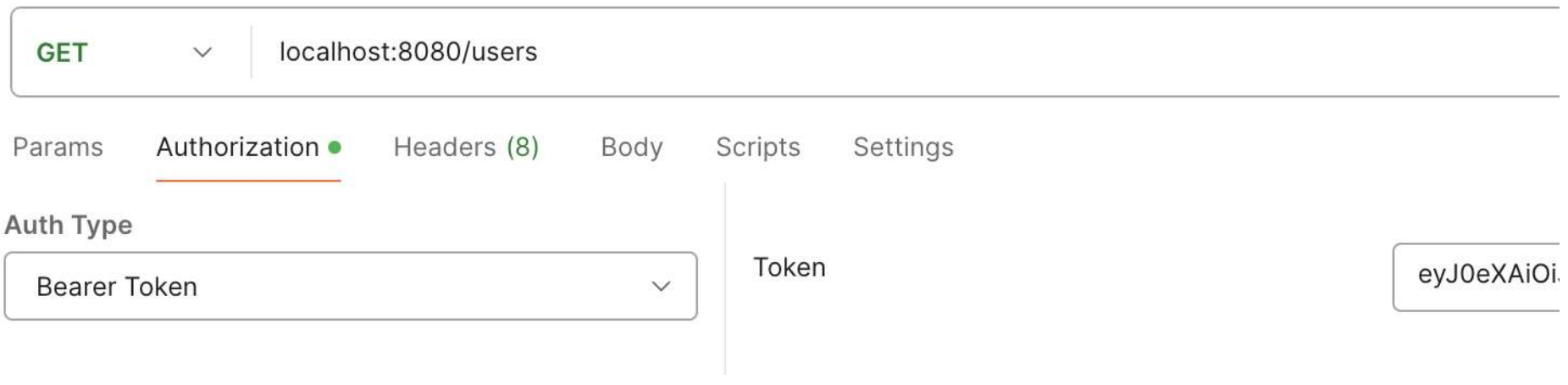
localhost:8080/login/oauth2/code/gitlab?code=4f811edfc5aa3fb24fe48f5abddb6093723553710ed2d2d994e7edf

Pretty print

{ "id_token":
"eyJ0eXAiOiJKV1QiLCJraWQiOiJrZXdpUXE5amlDODRDdlNzSllPQi1ONkE4V0ZMU1YyME1iLXk3SWxXRFNRIiwiYWxn
iZTIyNDMwNzkxMGIxZmQ3YjA4N2IzNjA3NmNkNmQxMTQ4OGJkODI5ZGE5OWZmOWMyZGVhM2NkNzUxNmI4N2U0NSIsImV4
ZUd1JXQlBNdE1MQXV2czg5alswMGMiLCJhdXRoX3RpbWUiOjE3NDQ1NTUyMzgsInN1Yl9sZWdhY3kiOiIwNiE2ZiNmZDk

SI6InNocmF5YW5zaCBqYWluIiwibmlja25hbWUiOiJzaHJheWFuc2g4IiwicHJlZmVycmVkX3VzZXJuYW1lIjoic2hyYX
dHRwczovL3NlY3VyZS5ncmF2YXRhci5jb20vYXZhdGFyL2ZmNzA0OTc0YTNhOWFkMjlmNDhhOGU3ODExMDkwNTVlYzYwN
iOltdfQ.f96dQeWof_TPB-_BjEZaGsENykgh0sJr0CpXZxwJrcIN5l6nJA47nOA2No0bBhW_cNs_EMnkRRL6_Kh-wqA9S
byTUyc0vLeZNNEPjjdlX2x_w_9Np4Bm9vXzYapV7y8_tNS1Z-aWHkj8mnPA-R31X20YPKzzECydzsJjo-
gyEB1DlQZ617Qb7PfnhqXUuvPVpIXHKaSYNKbps7r1HDRV1rLV64VB23pAy63l8H7izvW5eCnQn61vUXf88LvYL7PK3wd
zTkXhbCsZUKgm28PrGUXWVoijOgS6yemkX5ue4USIyMth_asVoXHEpA7ZiDGqhzTrsagNlYGyxbPKuU6DsRb-jY0y4jJ8
3Q2cbSkxmWZh5xjDEe1ioSA94HSyeSKyb0nxoUlWh8RqWFwD_kFNgL8cFtPs_0D8JcjFjQDI0iv0BN1_i4KMPAwNRBQUG

Now, only 1 task left, now Client will pass this TOKEN w

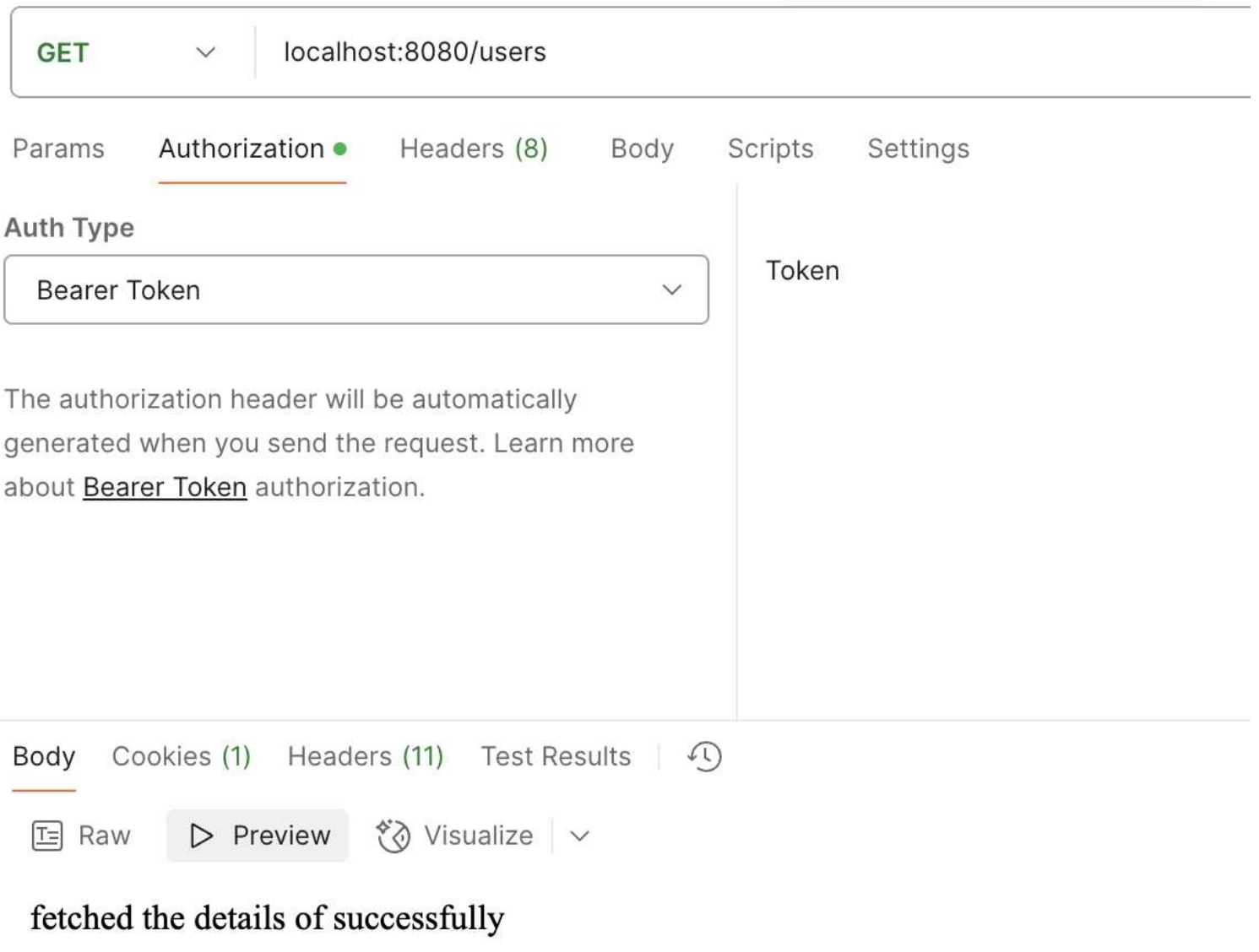


Created New Filter to Validate the token

```java
public class OAuthValidationFilter extends OncePerRequestFilter {

    private final OAuthTokenValidatorUtil tokenValidatorUtil;

    @Autowired
    public OAuthValidationFilter(OAuthTokenValidatorUtil tokenValidatorUtil) {
        this.tokenValidatorUtil = tokenValidatorUtil;
    }

    @Override
    protected void doFilterInternal(HttpServletRequest request,
                                    HttpServletResponse response,
                                    FilterChain filterChain) throws ServletException, IO

        String token = extractJwtFromRequest(request);
        if (token != null) {
```

```
            String username = tokenValidatorUtil.isTokenValid(token);
            if (StringUtil.isNullOrEmpty(username)) {
                response.sendError(HttpServletResponse.SC_UNAUTHORIZED, s: "Invalid or e
                return;
            }
            Authentication auth = new UsernamePasswordAuthenticationToken(username, cred
            SecurityContextHolder.getContext().setAuthentication(auth);
        }
        filterChain.doFilter(request, response);
    }

    private String extractJwtFromRequest(HttpServletRequest request) {
        String bearerToken = request.getHeader( s: "Authorization");
        if (bearerToken != null && bearerToken.startsWith("Bearer ")) {
            return bearerToken.substring( beginIndex: 7);
        }
        return null;
    }
}
```

```
@Configuration
@EnableWebSecurity
public class SecurityConfig {

    @Autowired
    private OAuthTokenValidatorUtil tokenValidatorUtil;

    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity htt
                                                   CustomOAuth2Succ
        http.authorizeHttpRequests(auth -> auth
                        .anyRequest().authenticated())
                .sessionManagement(session -> session
                        .sessionCreationPolicy(SessionCreationPolic
                .csrf(csrf -> csrf.disable())
                .oauth2Login(oauth -> oauth
                        .successHandler(successHandler))
                .addFilterBefore(new OAuthValidationFilter(tokenVal

        return http.build();
    }
}
```


GET
localhost:8080/users
Params
Authorization
Headers (8)
Body
Scripts
Settings
Auth Type
Bearer Token
Token
The authorization header will be automatically generated when you send the request. Learn more about Bearer Token authorization.
Body
Cookies (1)
Headers (11)
Test Results
Raw
Preview
Visualize
fetched the details of successfully