Angular WebApp Security with Cookie SessionId

(Reverse engineering Jhipster Generated Code)

arnaud.nauwynck@gmail.com

this document: https://github.com/Arnaud-Nauwynck/Presentations/Web/CookieSessionId-Angular-ReverseEngineering-Jhipster

Comparison Only with the JWT Mode

This document focus on comparisons
of the default "**JWT**" mode  "**HTTP Session**" Mode

For more detailed screenshots => please see document part 1

Step 1: re-generate ANOTHER Jhipster project with same name

```
# check already install globally
npm install -g generator-jhipster

# create a new empty dir
mkdir demo-sessionAuth
cd demo-sessionAuth

# launch jhipster generator
jhipster
```

jhipster

when prompted for project name (inferred for current directory name) => change to be same as "demo" project instead of "demo-sessionAuth"

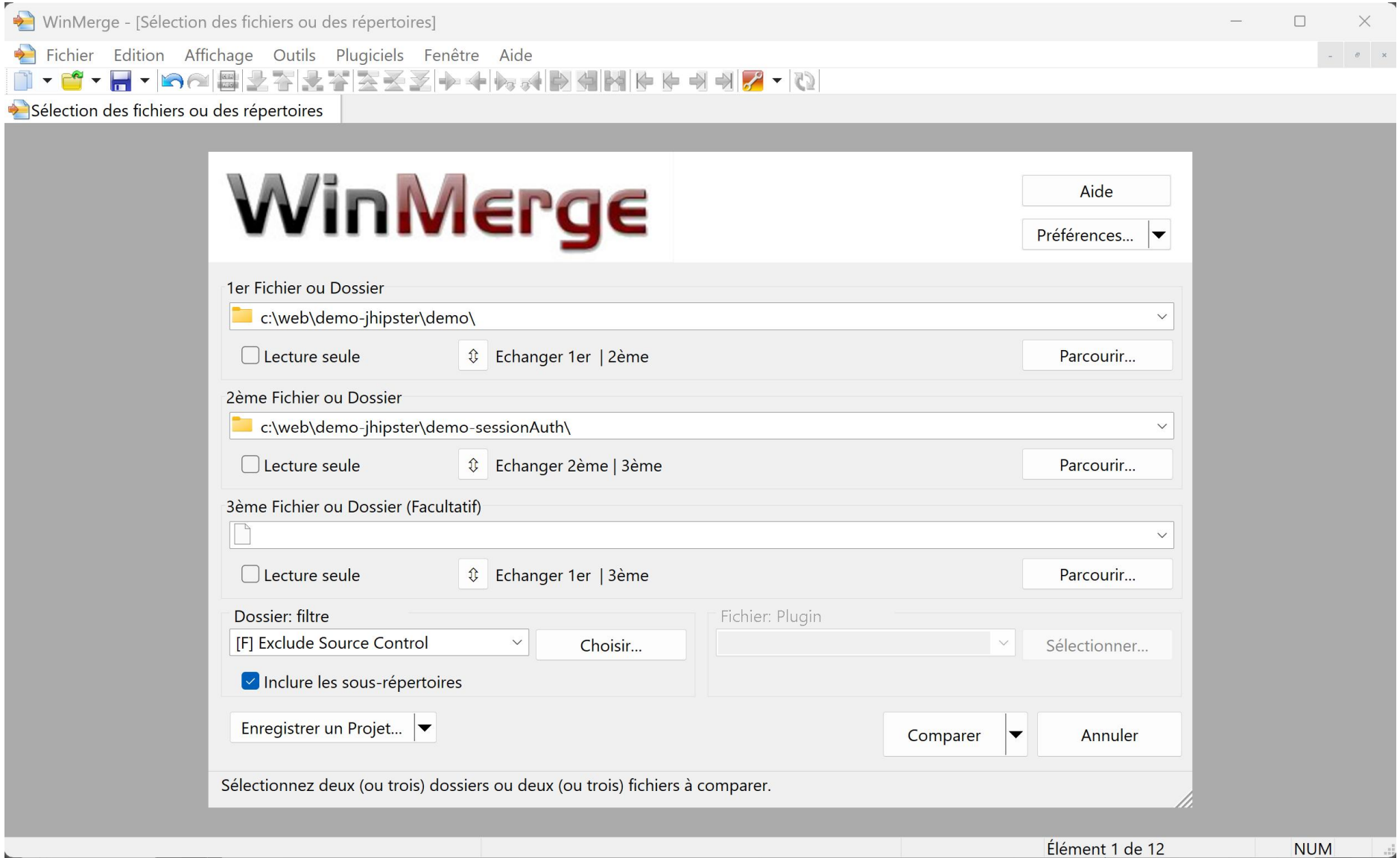
on question "Which type of authentication.. ?" change from default choice #1 to choice #3: HTTP Session Authentication

```
✔What is the base name of your application? demoSessionAuth
✔Which *type* of application would you like to create? Monolithic application (recommended for simple projects)
✔What is your default Java package name? com.mycompany.myapp
✔Would you like to use Maven or Gradle for building the backend? Maven
✔Do you want to make it reactive with Spring WebFlux? no
? Which *type* of authentication would you like to use?
  JWT authentication (stateless, with a token)
  OAuth 2.0 / OIDC Authentication (stateful, works with Keycloak and Okta)
❯ HTTP Session Authentication (stateful, default Spring Security mechanism)
```

… Wait few minutes

```
added 1692 packages, and audited 1693 packages in 2m

292 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
✔Application successfully committed to Git from c:/web/demo-jhipster/demo-sessionAuth.
✔Spring Boot application generated successfully.
  Run your Spring Boot application:
  ./mvnw (mvnw if using Windows Command Prompt)
✔Angular application generated successfully.
  Start your Webpack development server with:
  npm start


Congratulations, JHipster execution is complete!
If you find JHipster useful consider sponsoring the project https://www.jhipster.tech/sponsors/

Thanks for using JHipster!

c:\web\demo-jhipster\demo-sessionAuth>
```

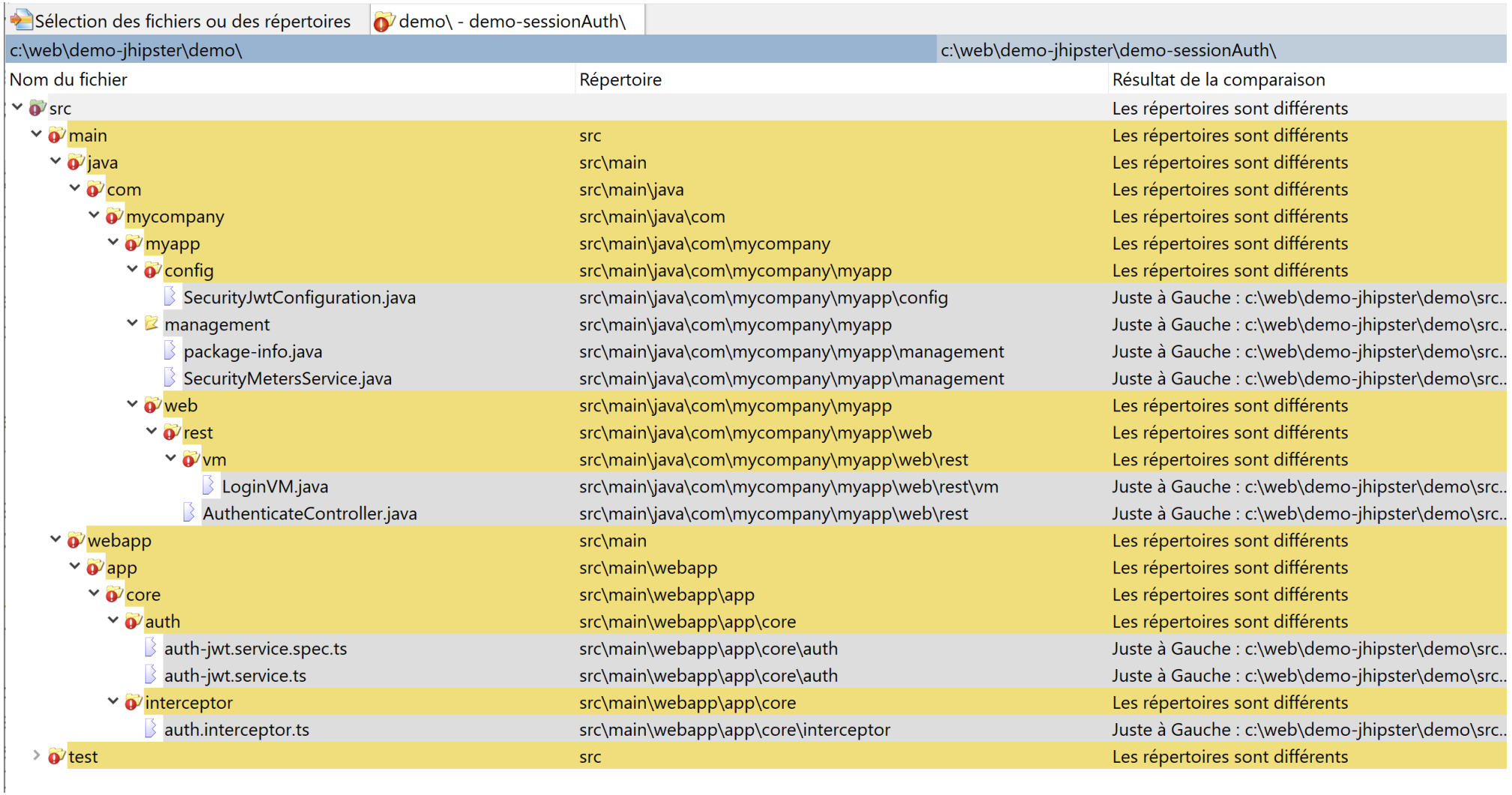
Comparing (WinMerge) Generated Apps JWT <-> HTTP Session



Differences JWT <-> HTTP Session

c:\web\demo-jhipster\demo\		c:\web\demo-jhipster\demo-sessionAuth\
Nom du fichier	Répertoire	Résultat de la comparaison
.husky		Les répertoires sont différents
src		Les répertoires sont différents
main	src	Les répertoires sont différents
docker	src\main	Les répertoires sont différents
java	src\main	Les répertoires sont différents
com	src\main\java	Les répertoires sont différents
mycompany	src\main\java\com	Les répertoires sont différents
myapp	src\main\java\com\mycompany	Les répertoires sont différents
config	src\main\java\com\mycompany\myapp	Les répertoires sont différents
domain	src\main\java\com\mycompany\myapp	Les répertoires sont différents
management	src\main\java\com\mycompany\myapp	Juste à Gauche : c:\web\demo-jhipster\demo
repository	src\main\java\com\mycompany\myapp	Les répertoires sont différents
security	src\main\java\com\mycompany\myapp	Les répertoires sont différents
service	src\main\java\com\mycompany\myapp	Les répertoires sont différents
web	src\main\java\com\mycompany\myapp	Les répertoires sont différents
resources	src\main	Les répertoires sont différents
config	src\main\resources	Les répertoires sont différents
.h2.server.properties	src\main\resources	Les fichiers de texte sont différents
webapp	src\main	Les répertoires sont différents
app	src\main\webapp	Les répertoires sont différents
account	src\main\webapp\app	Les répertoires sont différents
core	src\main\webapp\app	Les répertoires sont différents
auth	src\main\webapp\app\core	Les répertoires sont différents
interceptor	src\main\webapp\app\core	Les répertoires sont différents
layouts	src\main\webapp\app	Les répertoires sont différents
navbar	src\main\webapp\app\layouts	Les répertoires sont différents
login	src\main\webapp\app	Les répertoires sont différents
swagger-ui	src\main\webapp	Les répertoires sont différents
test	src	Les répertoires sont différents
.yo-rc.json		Les fichiers de texte sont différents
package-lock.json		Les fichiers de texte sont différents
pom.xml		Les fichiers de texte sont différents
sonar-project.properties		Les fichiers de texte sont différents

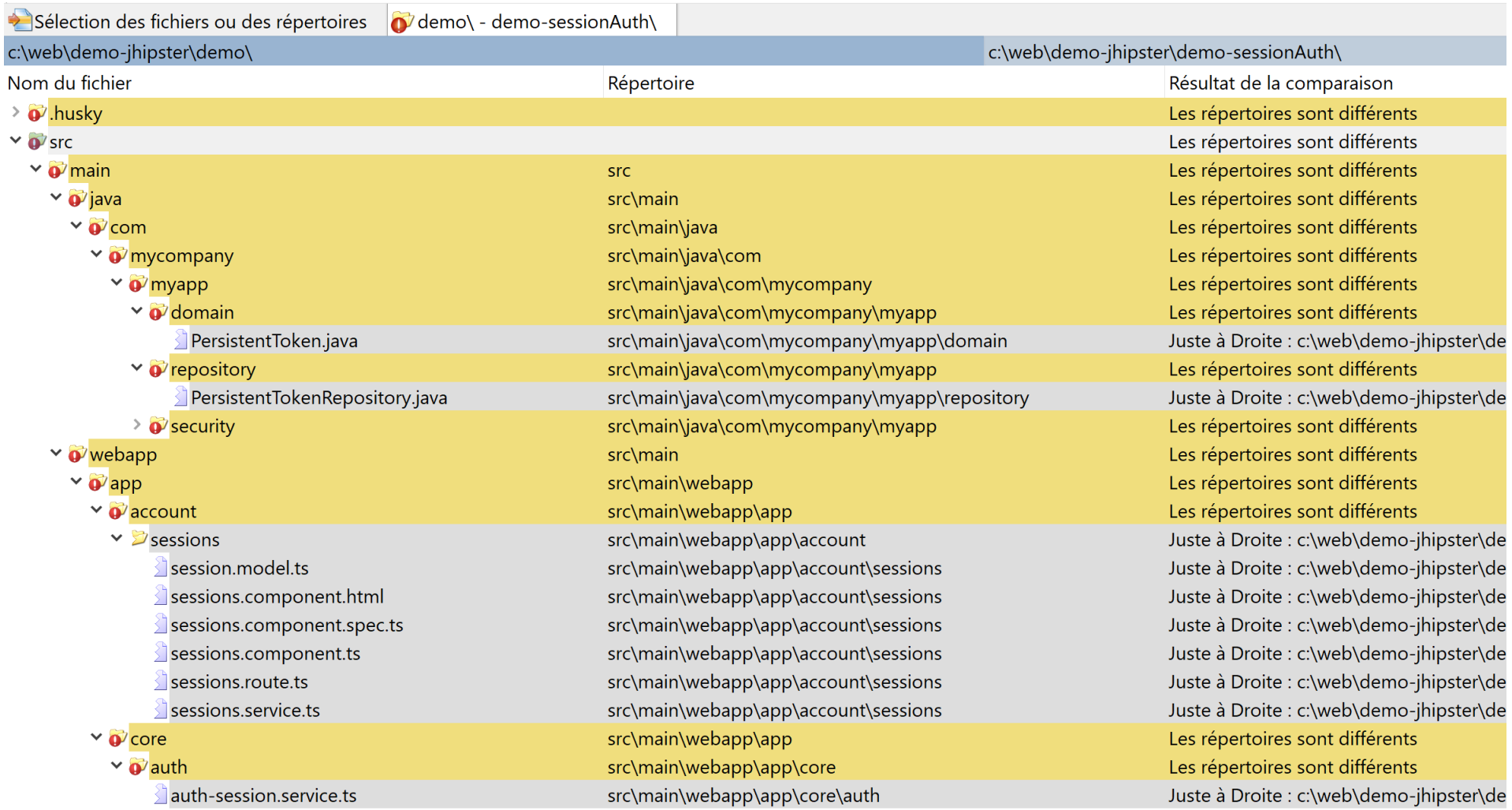
Files Only in "JWT" mode



Sélection des fichiers ou des répertoires | demo\ - demo-sessionAuth\

c:\web\demo-jhipster\demo\	c:\web\demo-jhipster\demo-sessionAuth\	
Nom du fichier	Répertoire	Résultat de la comparaison
src		Les répertoires sont différents
main	src	Les répertoires sont différents
java	src\main	Les répertoires sont différents
com	src\main\java	Les répertoires sont différents
mycompany	src\main\java\com	Les répertoires sont différents
myapp	src\main\java\com\mycompany	Les répertoires sont différents
config	src\main\java\com\mycompany\myapp	Les répertoires sont différents
SecurityJwtConfiguration.java	src\main\java\com\mycompany\myapp\config	Juste à Gauche : c:\web\demo-jhipster\demo\src..
management	src\main\java\com\mycompany\myapp	Juste à Gauche : c:\web\demo-jhipster\demo\src..
package-info.java	src\main\java\com\mycompany\myapp\management	Juste à Gauche : c:\web\demo-jhipster\demo\src..
SecurityMetersService.java	src\main\java\com\mycompany\myapp\management	Juste à Gauche : c:\web\demo-jhipster\demo\src..
web	src\main\java\com\mycompany\myapp	Les répertoires sont différents
rest	src\main\java\com\mycompany\myapp\web	Les répertoires sont différents
vm	src\main\java\com\mycompany\myapp\web\rest	Les répertoires sont différents
LoginVM.java	src\main\java\com\mycompany\myapp\web\rest\vm	Juste à Gauche : c:\web\demo-jhipster\demo\src..
AuthenticateController.java	src\main\java\com\mycompany\myapp\web\rest	Juste à Gauche : c:\web\demo-jhipster\demo\src..
webapp	src\main	Les répertoires sont différents
app	src\main\webapp	Les répertoires sont différents
core	src\main\webapp\app	Les répertoires sont différents
auth	src\main\webapp\app\core	Les répertoires sont différents
auth-jwt.service.spec.ts	src\main\webapp\app\core\auth	Juste à Gauche : c:\web\demo-jhipster\demo\src..
auth-jwt.service.ts	src\main\webapp\app\core\auth	Juste à Gauche : c:\web\demo-jhipster\demo\src..
interceptor	src\main\webapp\app\core	Les répertoires sont différents
auth.interceptor.ts	src\main\webapp\app\core\interceptor	Juste à Gauche : c:\web\demo-jhipster\demo\src..
test	src	Les répertoires sont différents

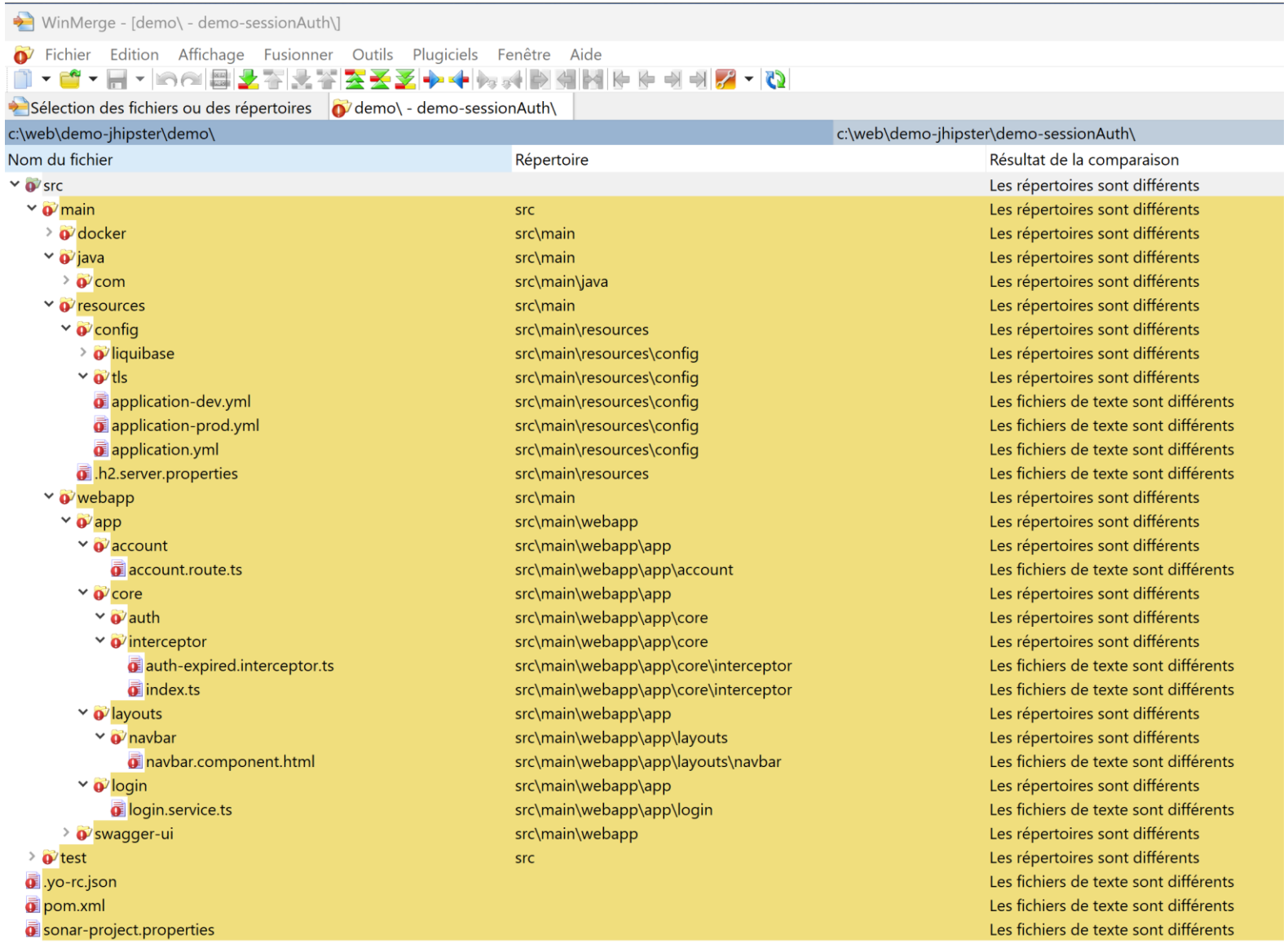
Files Only in "Http Session" Mode



Sélection des fichiers ou des répertoires | demo\ - demo-sessionAuth\

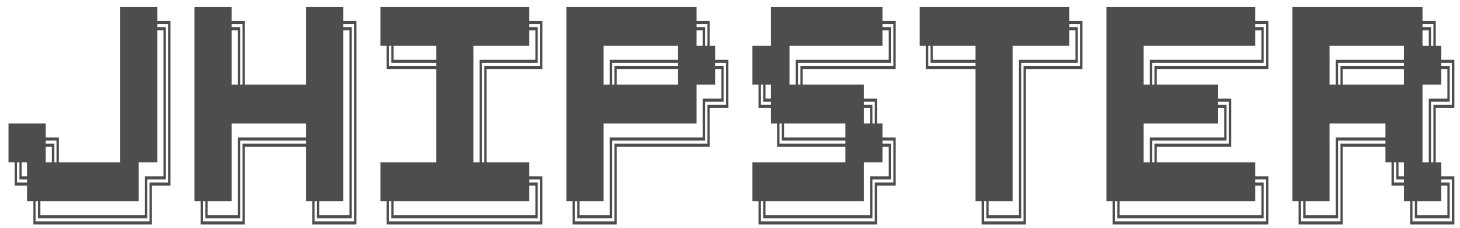
c:\web\demo-jhipster\demo\		c:\web\demo-jhipster\demo-sessionAuth\
Nom du fichier	Répertoire	Résultat de la comparaison
.husky		Les répertoires sont différents
src		Les répertoires sont différents
main	src	Les répertoires sont différents
java	src\main	Les répertoires sont différents
com	src\main\java	Les répertoires sont différents
mycompany	src\main\java\com	Les répertoires sont différents
myapp	src\main\java\com\mycompany	Les répertoires sont différents
domain	src\main\java\com\mycompany\myapp	Les répertoires sont différents
PersistentToken.java	src\main\java\com\mycompany\myapp\domain	Juste à Droite : c:\web\demo-jhipster\de
repository	src\main\java\com\mycompany\myapp	Les répertoires sont différents
PersistentTokenRepository.java	src\main\java\com\mycompany\myapp\repository	Juste à Droite : c:\web\demo-jhipster\de
security	src\main\java\com\mycompany\myapp	Les répertoires sont différents
webapp	src\main	Les répertoires sont différents
app	src\main\webapp	Les répertoires sont différents
account	src\main\webapp\app	Les répertoires sont différents
sessions	src\main\webapp\app\account	Juste à Droite : c:\web\demo-jhipster\de
session.model.ts	src\main\webapp\app\account\sessions	Juste à Droite : c:\web\demo-jhipster\de
sessions.component.html	src\main\webapp\app\account\sessions	Juste à Droite : c:\web\demo-jhipster\de
sessions.component.spec.ts	src\main\webapp\app\account\sessions	Juste à Droite : c:\web\demo-jhipster\de
sessions.component.ts	src\main\webapp\app\account\sessions	Juste à Droite : c:\web\demo-jhipster\de
sessions.route.ts	src\main\webapp\app\account\sessions	Juste à Droite : c:\web\demo-jhipster\de
sessions.service.ts	src\main\webapp\app\account\sessions	Juste à Droite : c:\web\demo-jhipster\de
core	src\main\webapp\app	Les répertoires sont différents
auth	src\main\webapp\app\core	Les répertoires sont différents
auth-session.service.ts	src\main\webapp\app\core\auth	Juste à Droite : c:\web\demo-jhipster\de

Different Files (maybe not for auth mode)



Launching Backend:
mvn -Pdev spring-boot:run

```
$ mvn -Pdev spring-boot:run
[INFO] Scanning for projects...
[INFO]
[INFO] ----------------------< com.mycompany.myapp:demo >----------------------
[INFO] Building Demo 0.0.1-SNAPSHOT
[INFO]   from pom.xml
[INFO] --------------------------------[ jar ]---------------------------------
[INFO]
[INFO] >>> spring-boot:3.3.5:run (default-cli) > test-compile @ demo >>>

[INFO] --- spring-boot:3.3.5:run (default-cli) @ demo ---
[INFO] Attaching agents: []
```



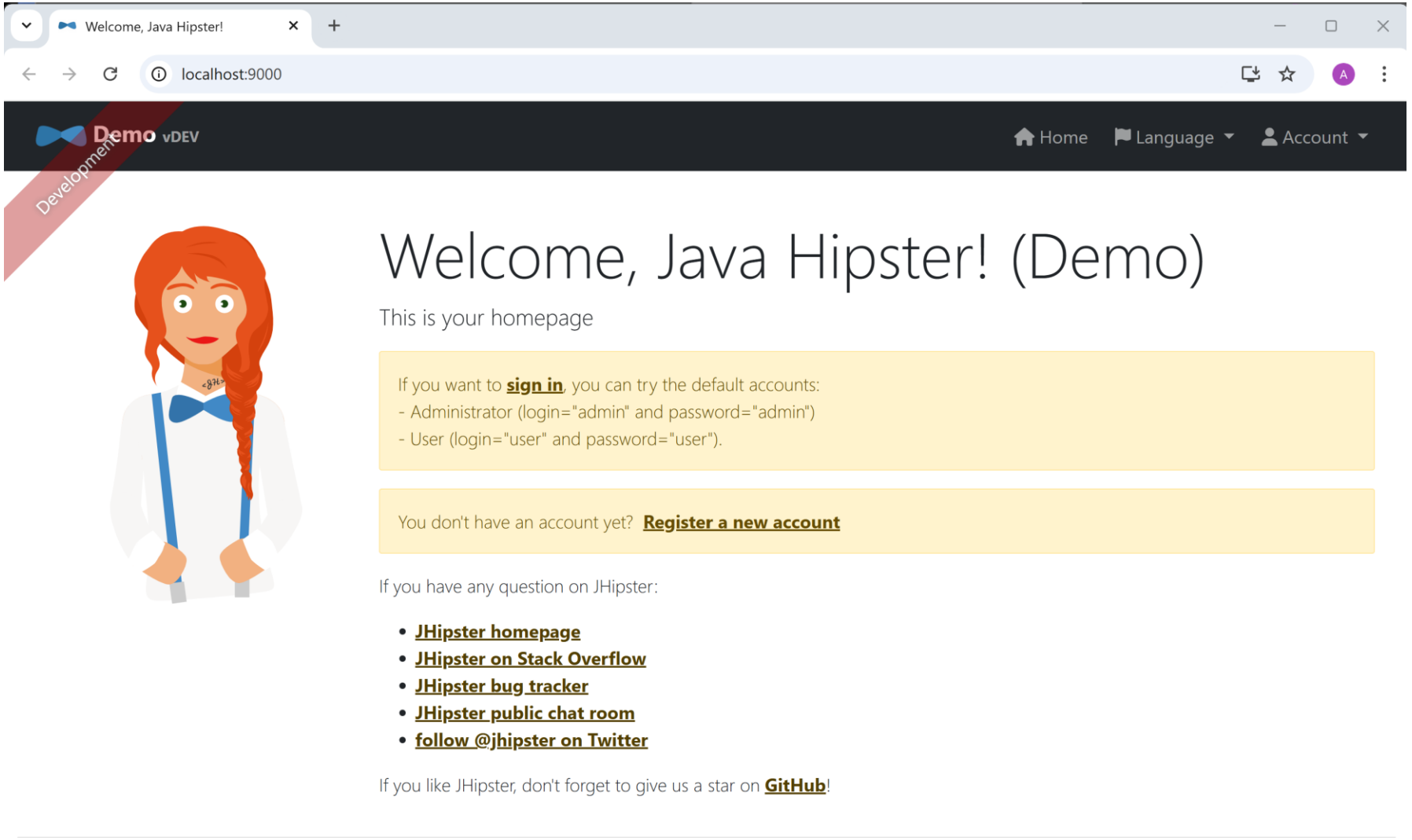

```
2024-11-19T17:05:44.632+01:00  INFO 7000 --- [  restartedMain] com.mycompany.myapp.DemoApp              : Started DemoApp in 11.616 seconds (process runni
ng for 12.128)
2024-11-19T17:05:44.644+01:00  INFO 7000 --- [  restartedMain] com.mycompany.myapp.DemoApp              :
----------------------------------------------------------
        Application 'demo' is running! Access URLs:
        Local:          http://localhost:8080/
        External:       http://192.168.0.50:8080/
        Profile(s):     [dev, api-docs]
----------------------------------------------------------
```

Launching Frontend:
npm run start (= ng serve)

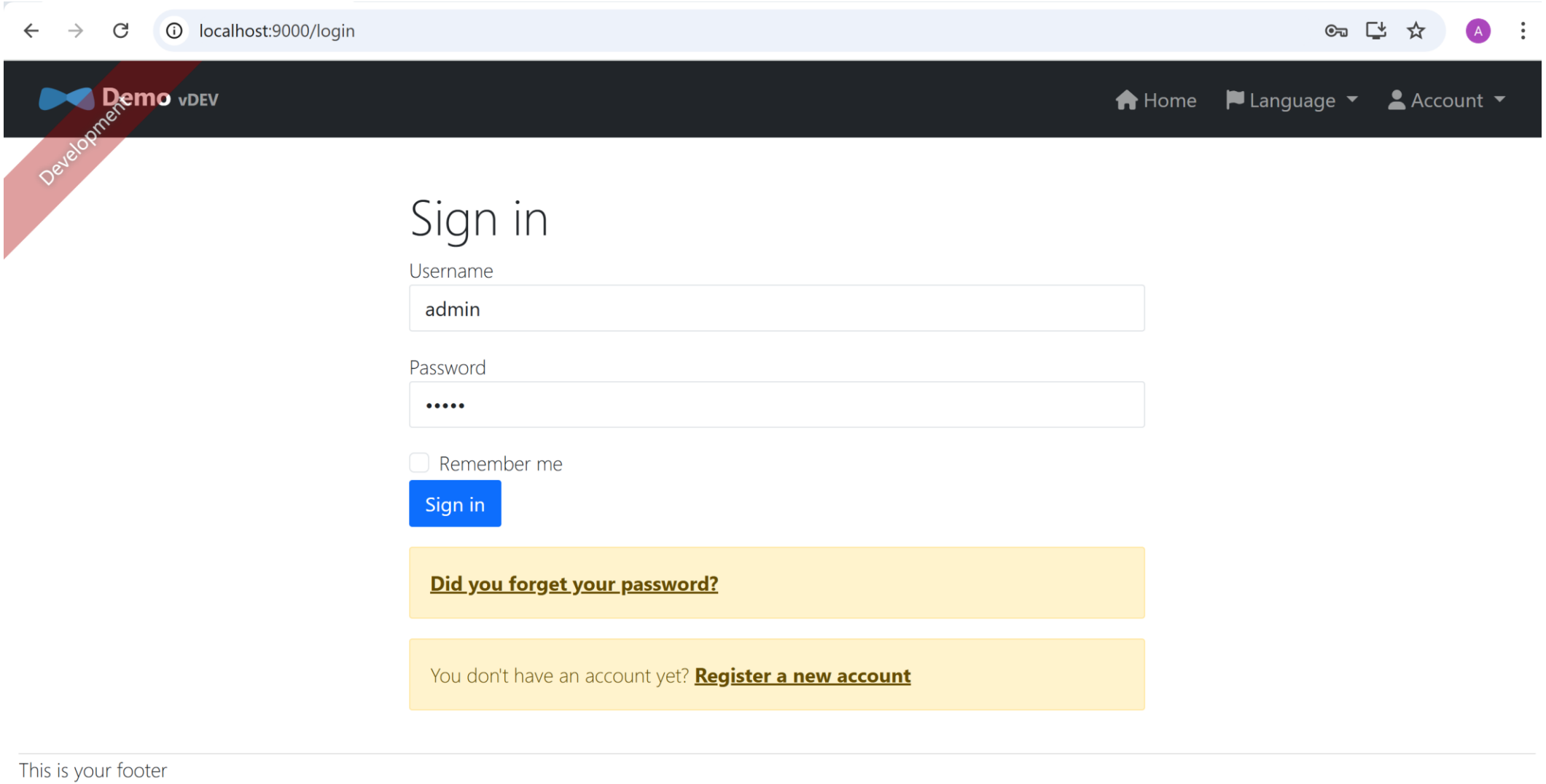
```
$ npm run start

> demo@0.0.1-SNAPSHOT start
> ng serve --hmr

Node.js version v23.2.0 detected.
```

```
** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **


√ Compiled successfully.
Notifications are disabled
Reason: DisabledByGroupPolicy Please make sure that the app id is set correctly.
Command Line: C:\web\demo-jhipster\demo\node_modules\node-notifier\vendor\snoreToast\snoretoast-x64.exe -pipeName \\.\pi
cbbca1e849 -p C:\web\demo-jhipster\demo\webpack\logo-jhipster.png -m "Build successful" -t Demo
[Browsersync] Proxying: http://localhost:4200
[Browsersync] Access URLs:
 ------------------------------------
       Local: http://localhost:9000
    External: http://192.168.0.50:9000
 ------------------------------------
          UI: http://localhost:3001
 UI External: http://192.168.0.50:3001
 ------------------------------------
```

Home page - not logged-in



Click sign-in => /login page



Home Page after logged-in

localhost:9000

Demo vDEV

Development

Home | Entities | Administration | Language | Account

Welcome, Java Hipster! (Demo)

This is your homepage

You are logged in as user "admin".

If you have any question on JHipster:

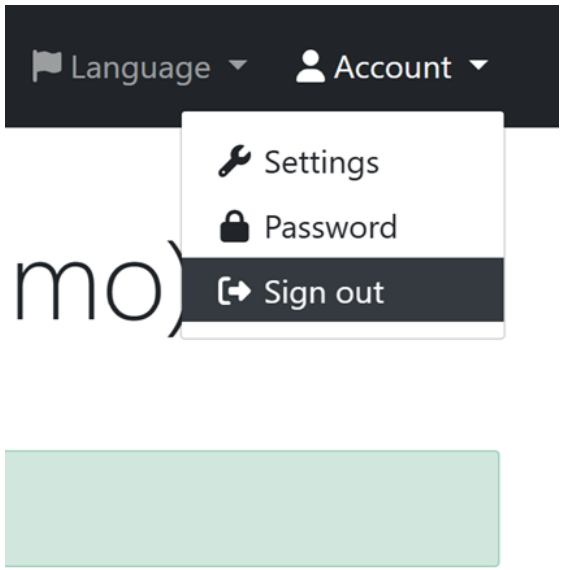
- **JHipster homepage**
- **JHipster on Stack Overflow**
- **JHipster bug tracker**
- **JHipster public chat room**
- **follow @jhipster on Twitter**

If you like JHipster, don't forget to give us a star on **GitHub**!

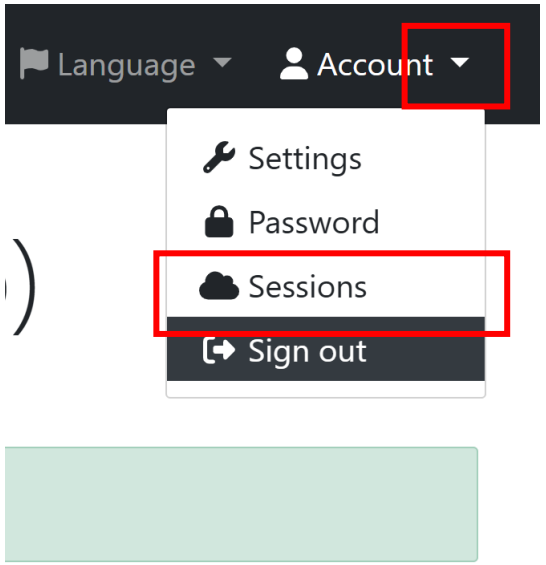
This is your footer

Different Menu for Account Sub Menu Item "Sessions"

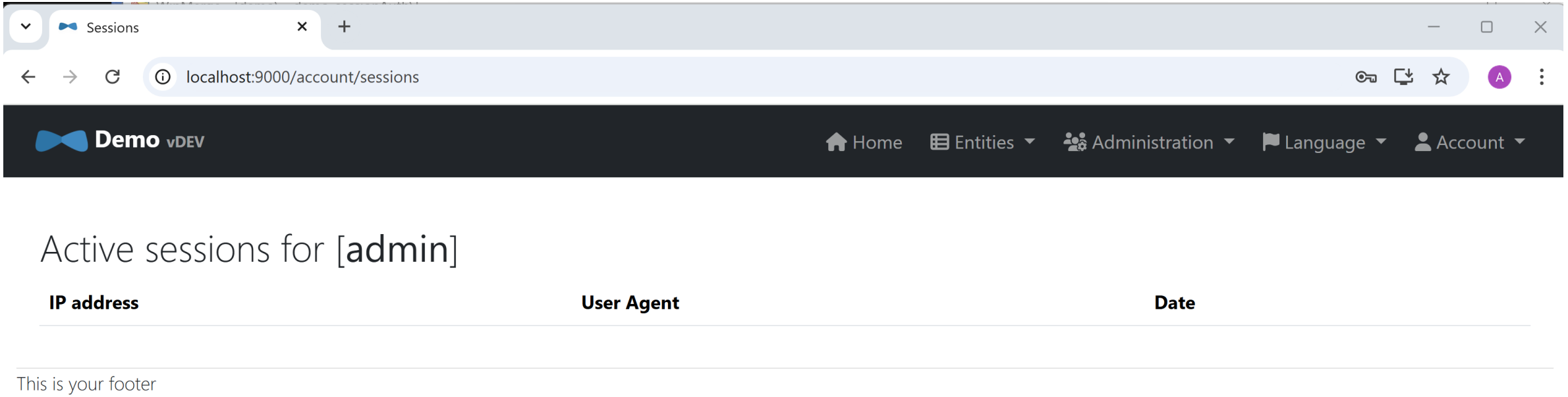
"**JWT**" Mode



"**Http Session**" Mode

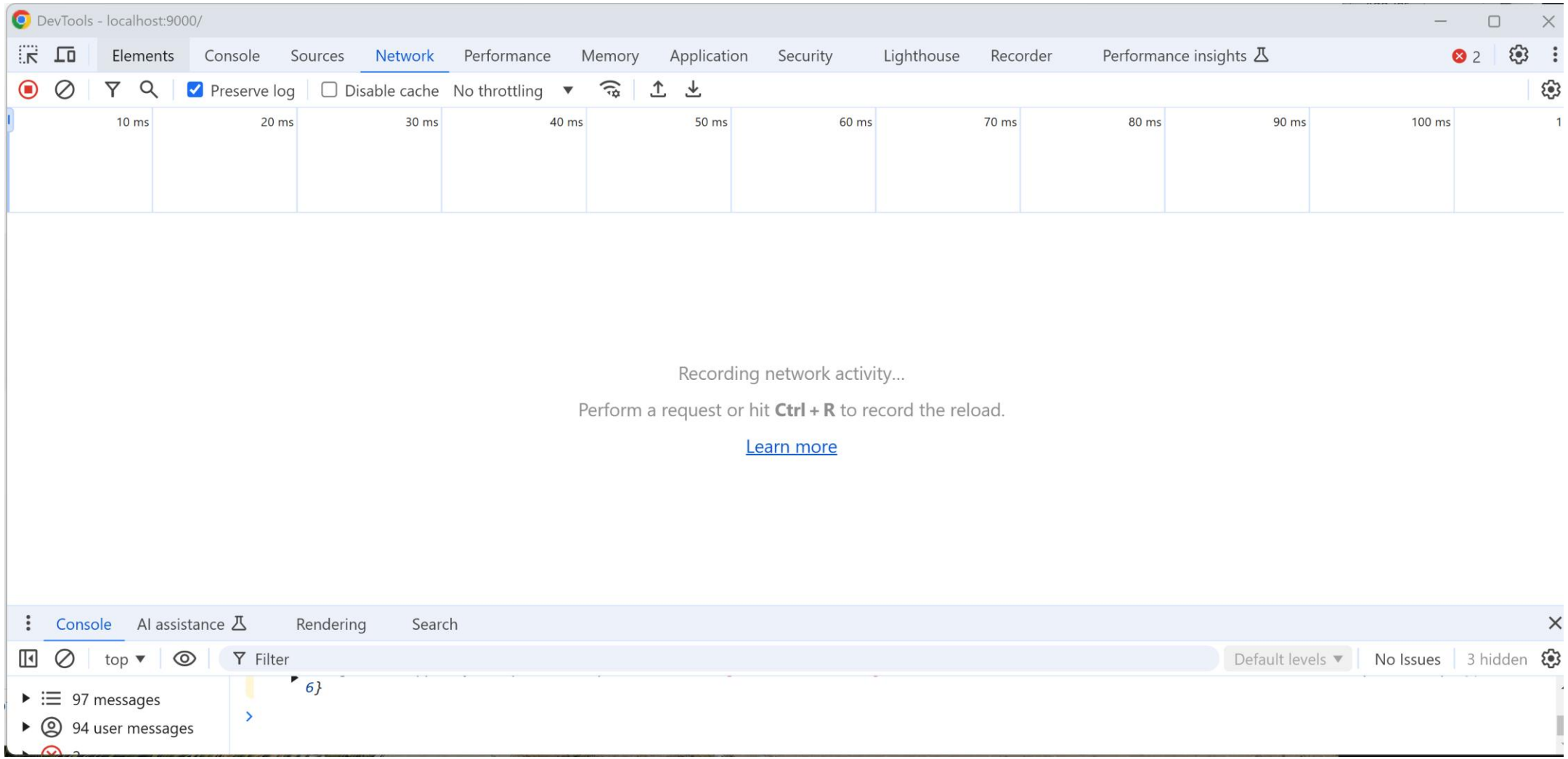


Sessions View for Current User

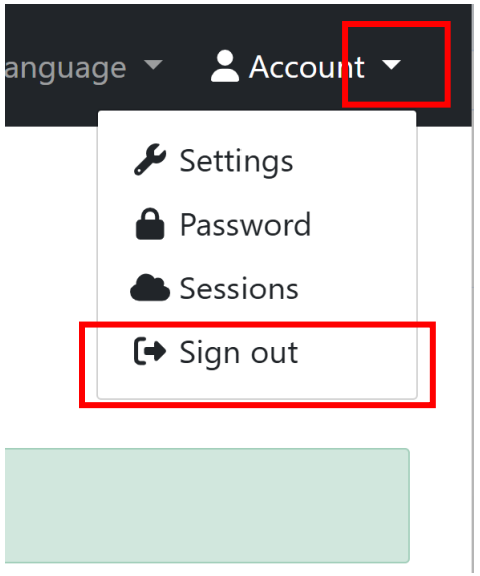


Debuging Security from "Http" viewpoint

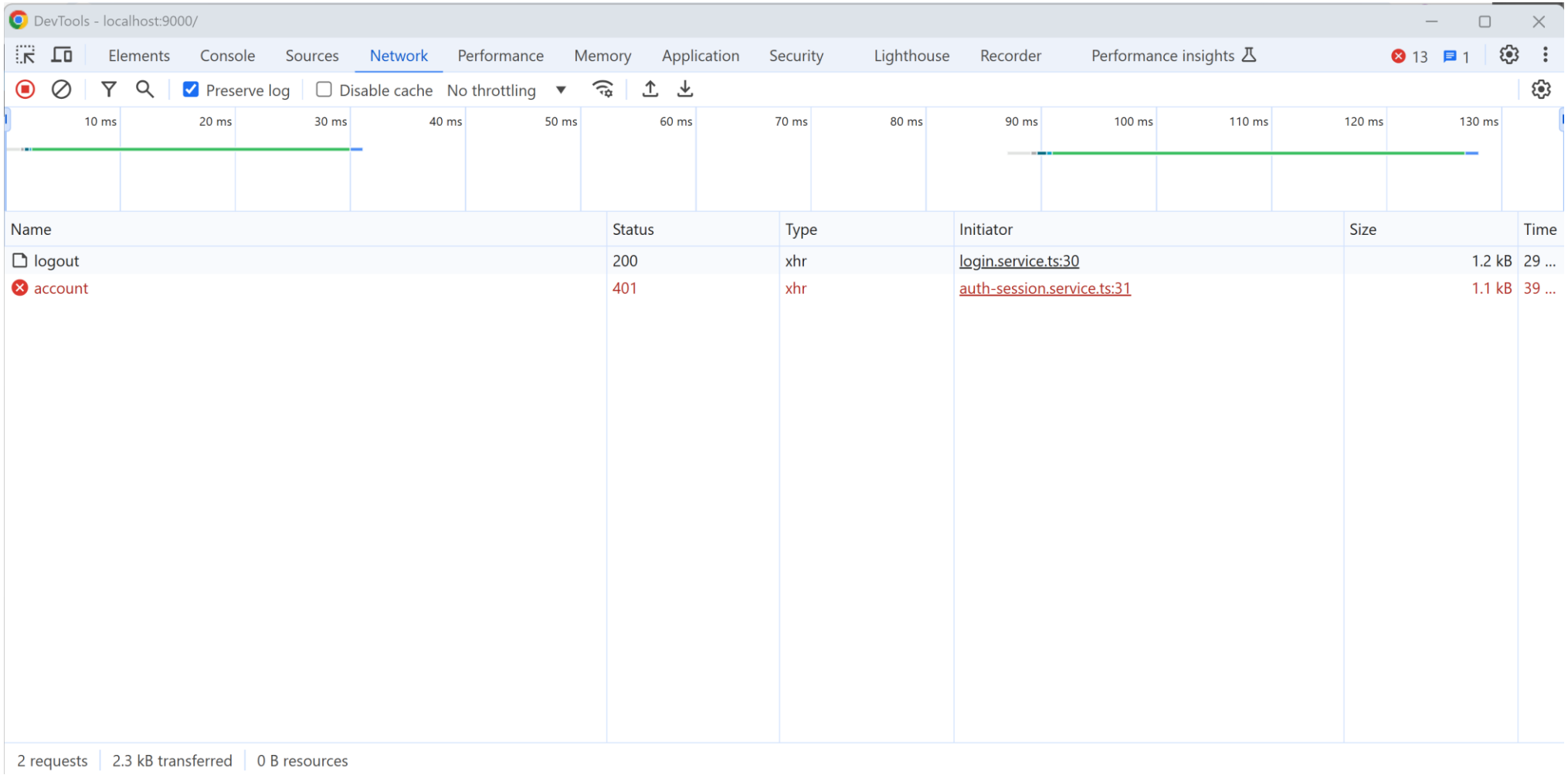
Chrome Dev Tools > Network



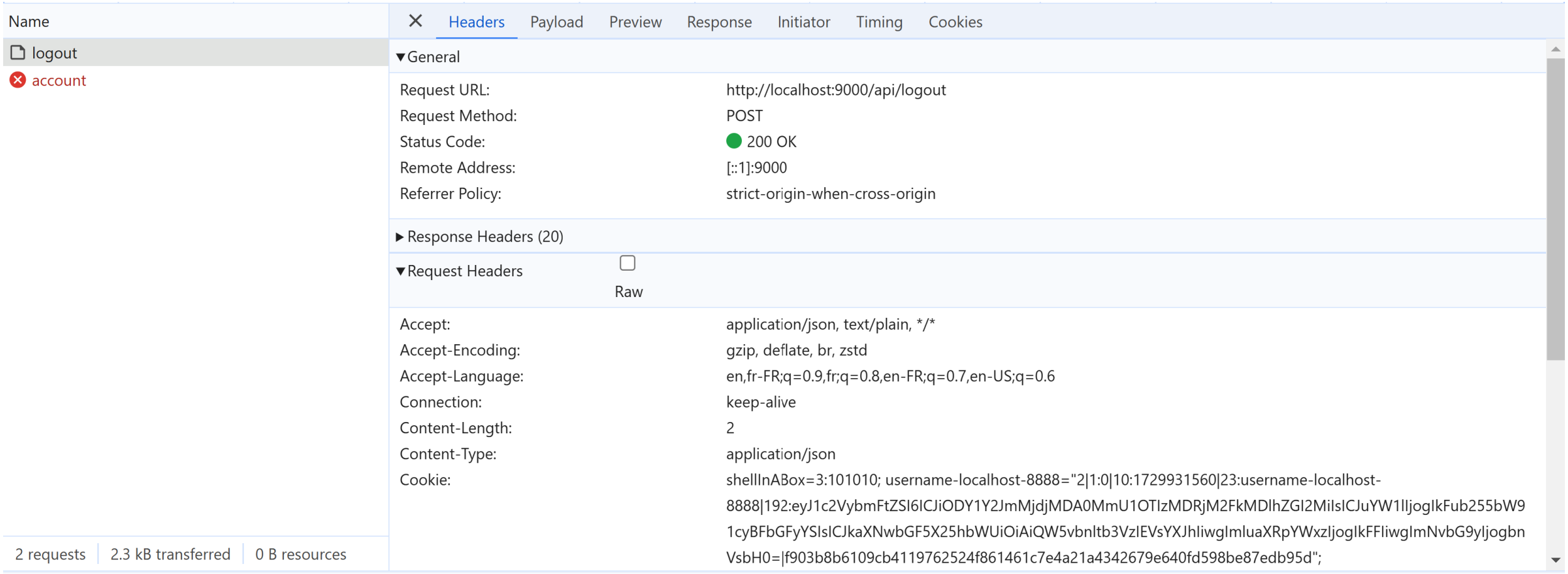
Sign Out => 2 Http calls (was 0 for JWT mode)



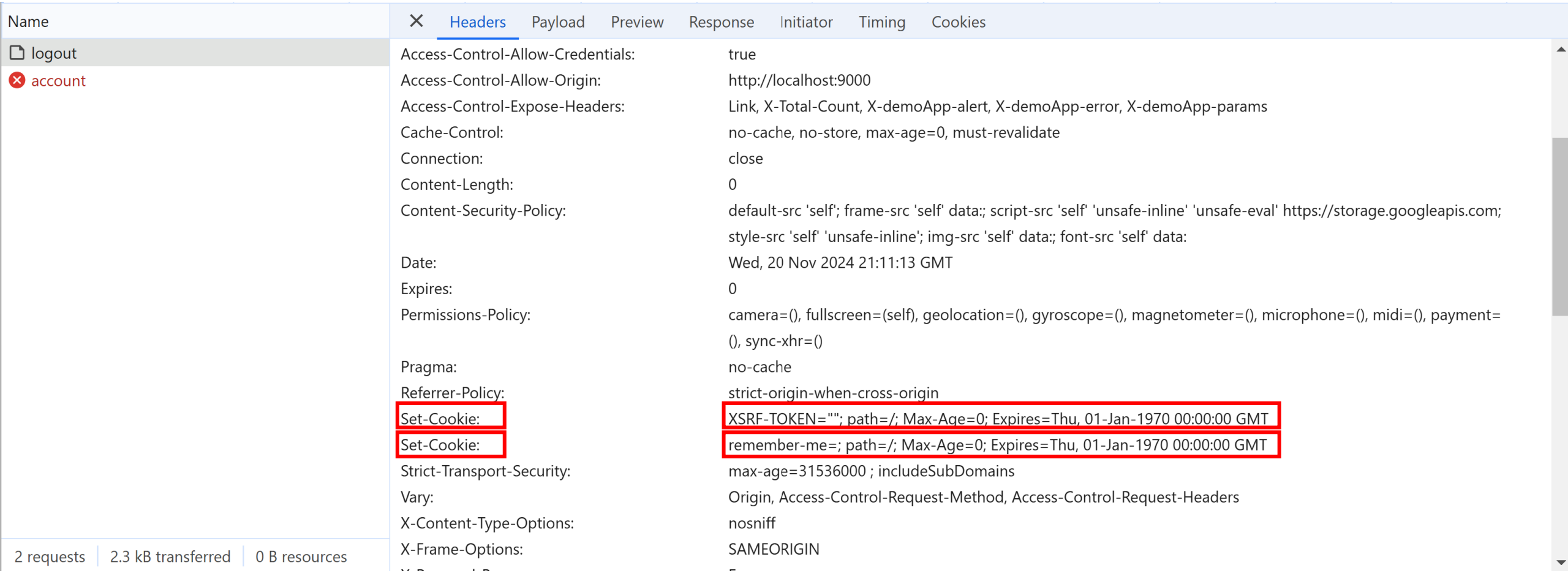
2 Http Requests "/api/logout", then refresh "/api/account"



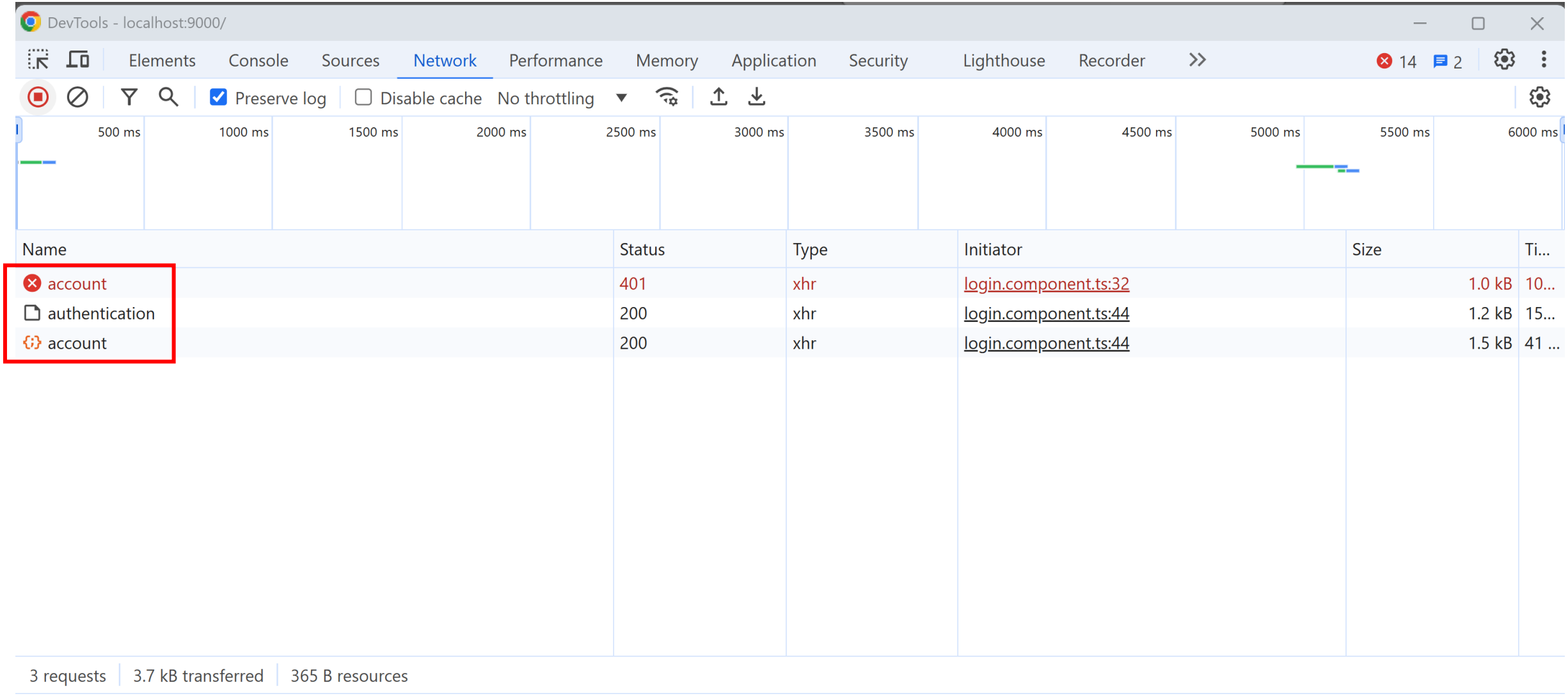
Http POST /api/logout



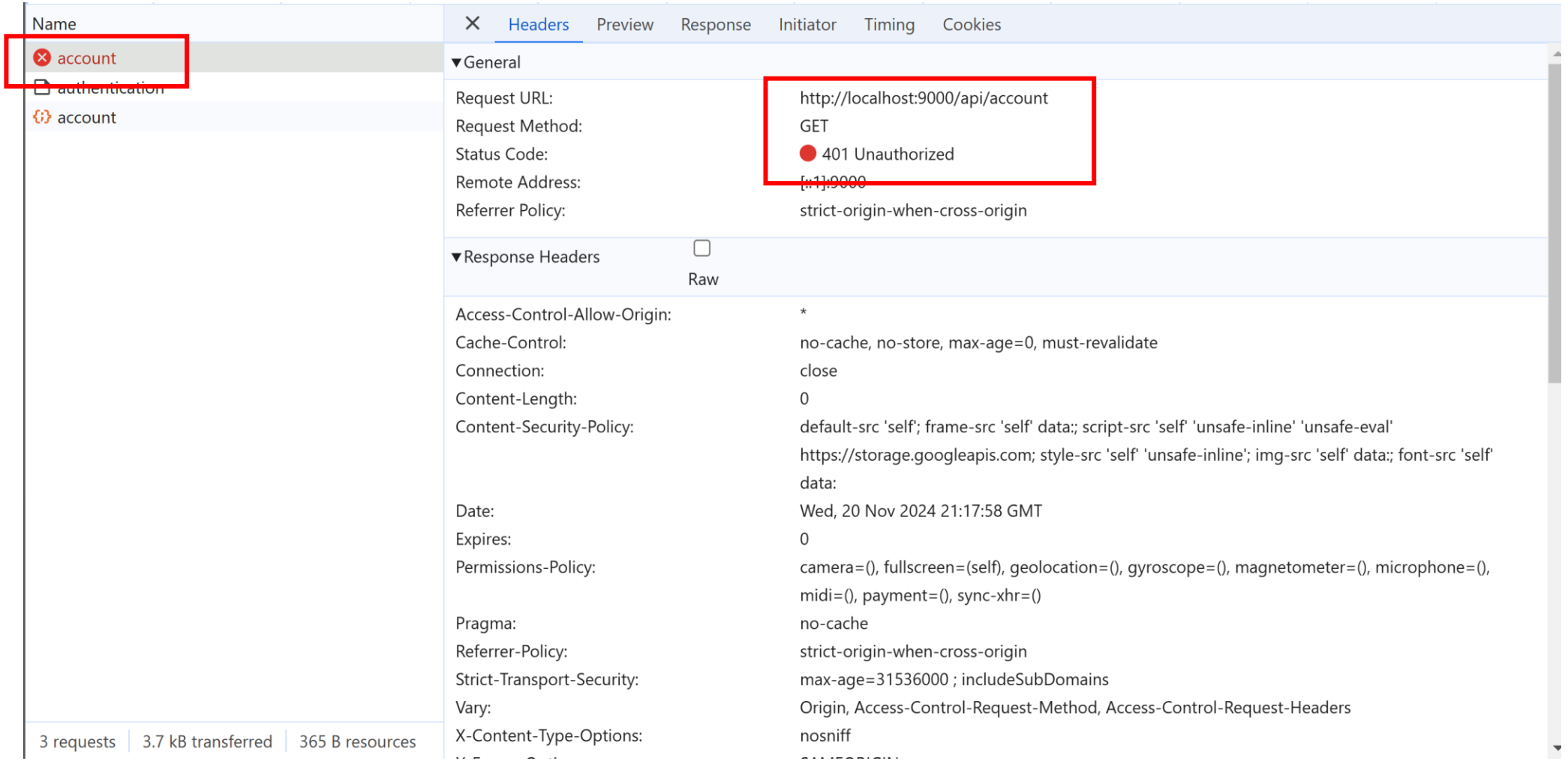
Http POST /api/logout => response Headers : 2 x Set-Cookie: ... to cleanup



Sign-in => 3 Http Requests



Request [1/3]: Http GET /api/account no auth Header => 401 Unauthorized



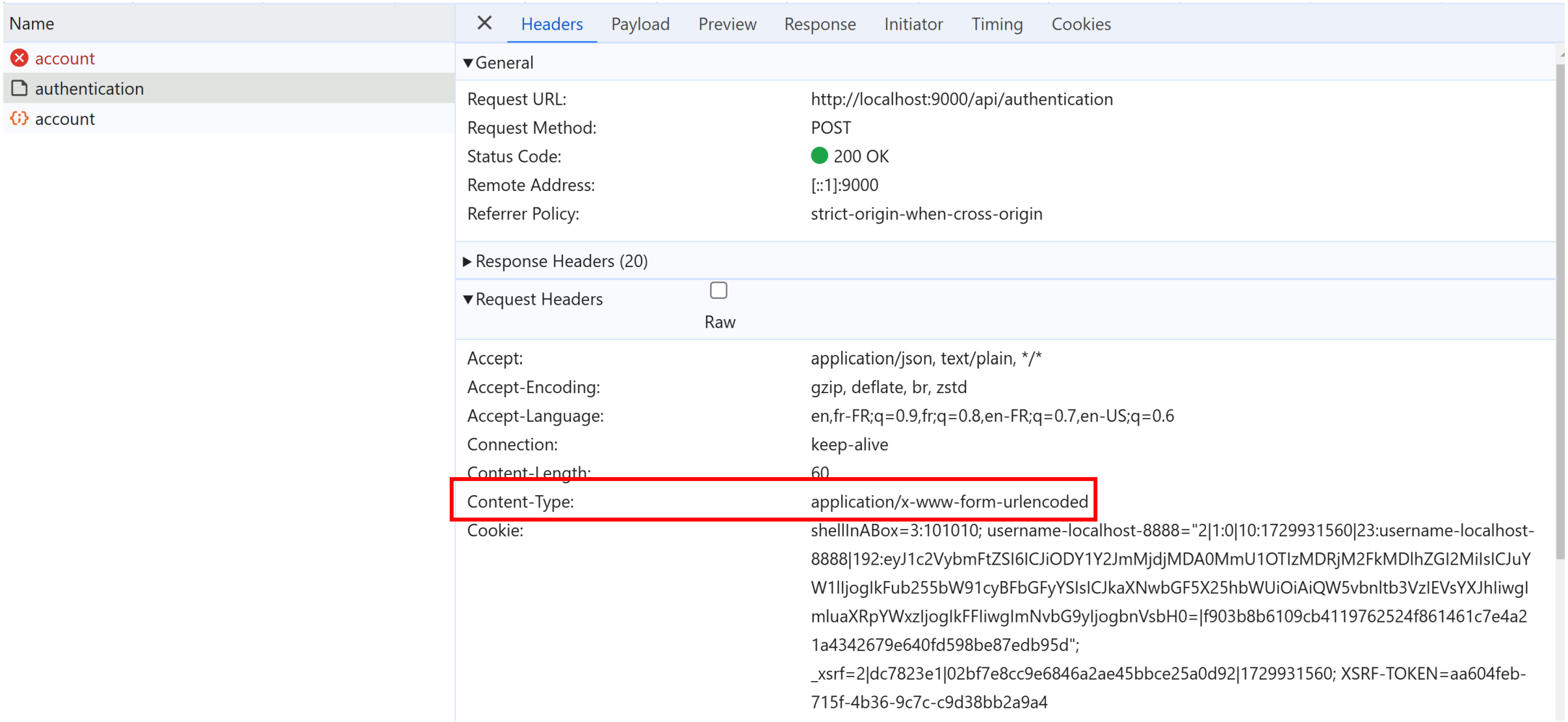
MISSING Header "Cookie"= "...; JSESSIONID=..." or Header "Authorization" => FAILING with 401

Request [1/3] http GET /api/account

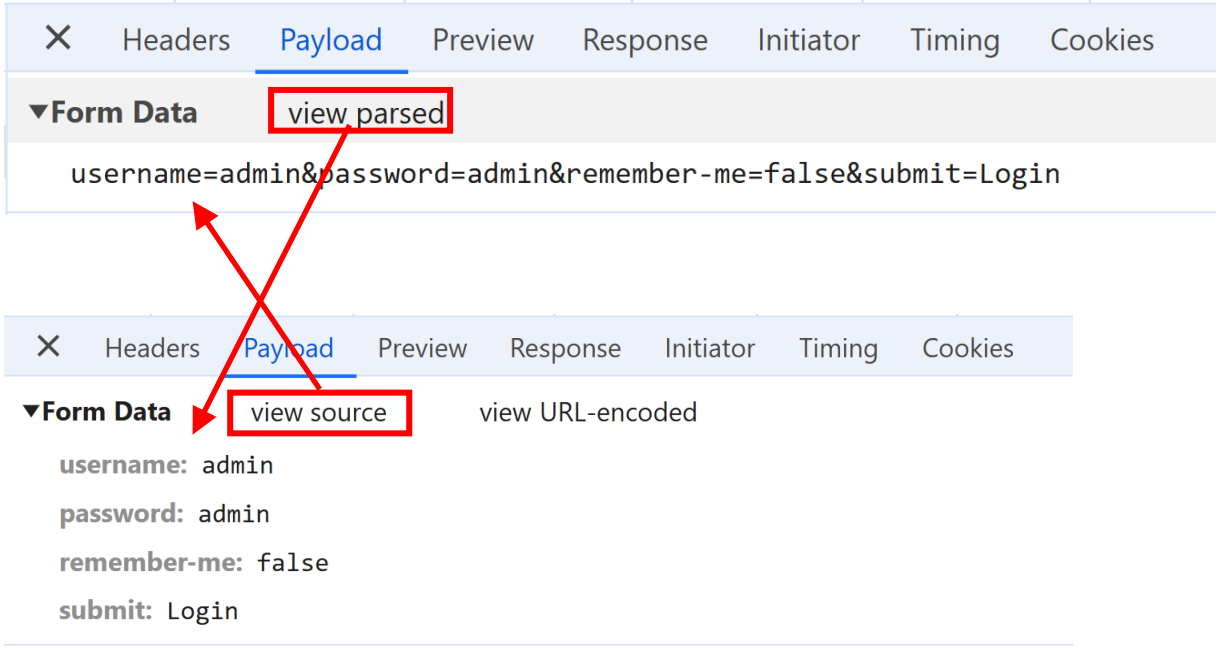
"**JWT**" Mode  "**Http Session**" Mode

IDEM as in JWT Mode

Request [2/3]: Http POST /api/authenticate content-Type: x-www-form-urlencoded



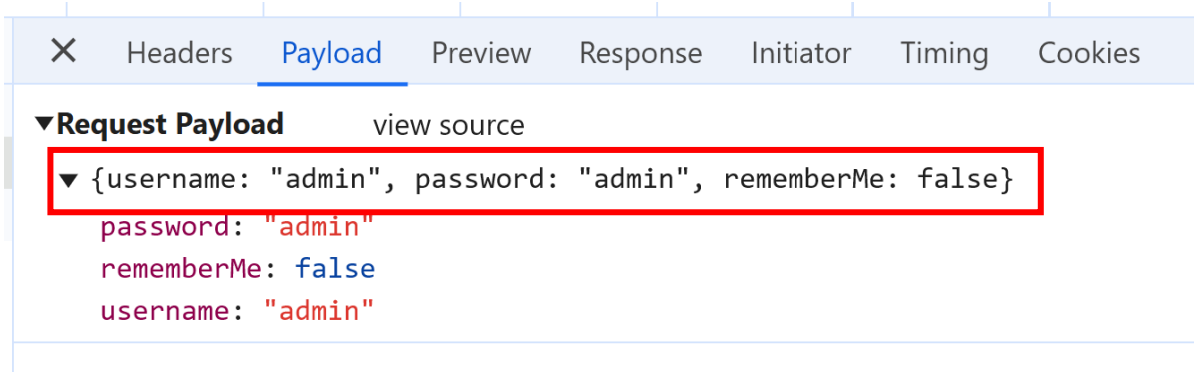
Request [2/3] Payload : username & password



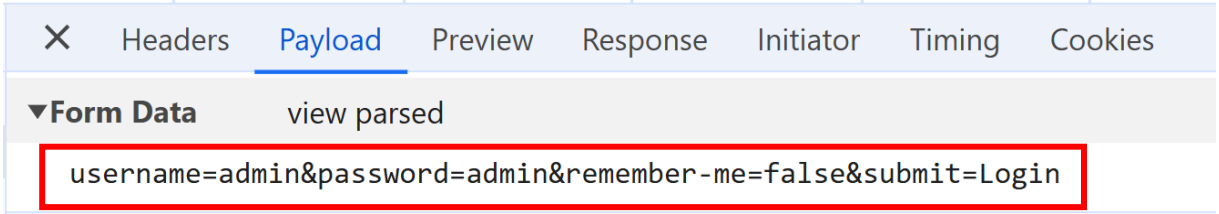
Request [2/3] different (but equivalent) to JWT Mode

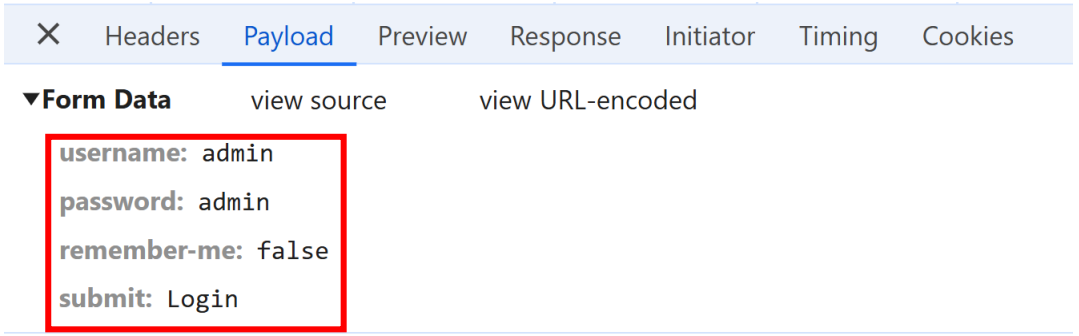
"**JWT**" Mode ⟺ "**Http Session**" Mode

Payload Request Body was in JSON

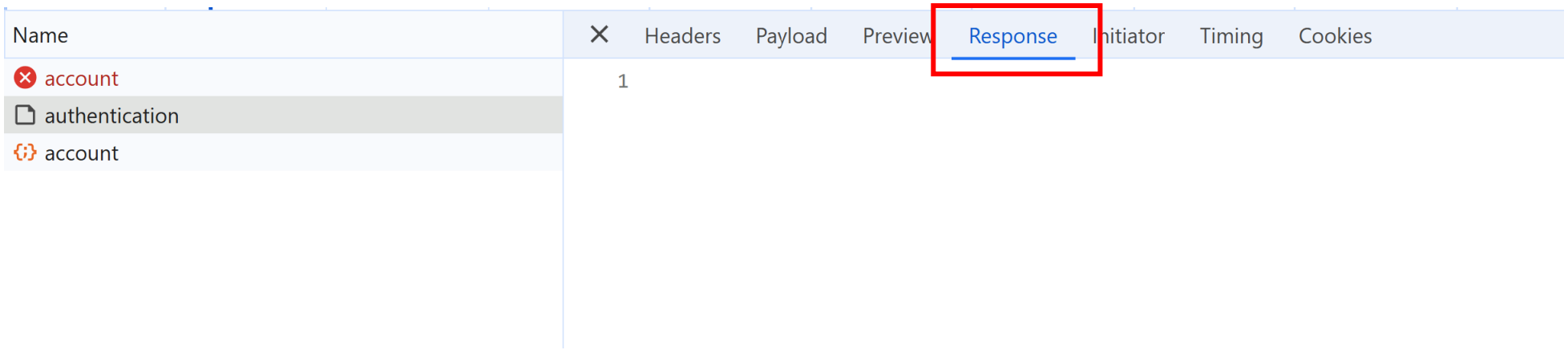


Payload Request in "URL-encoded"

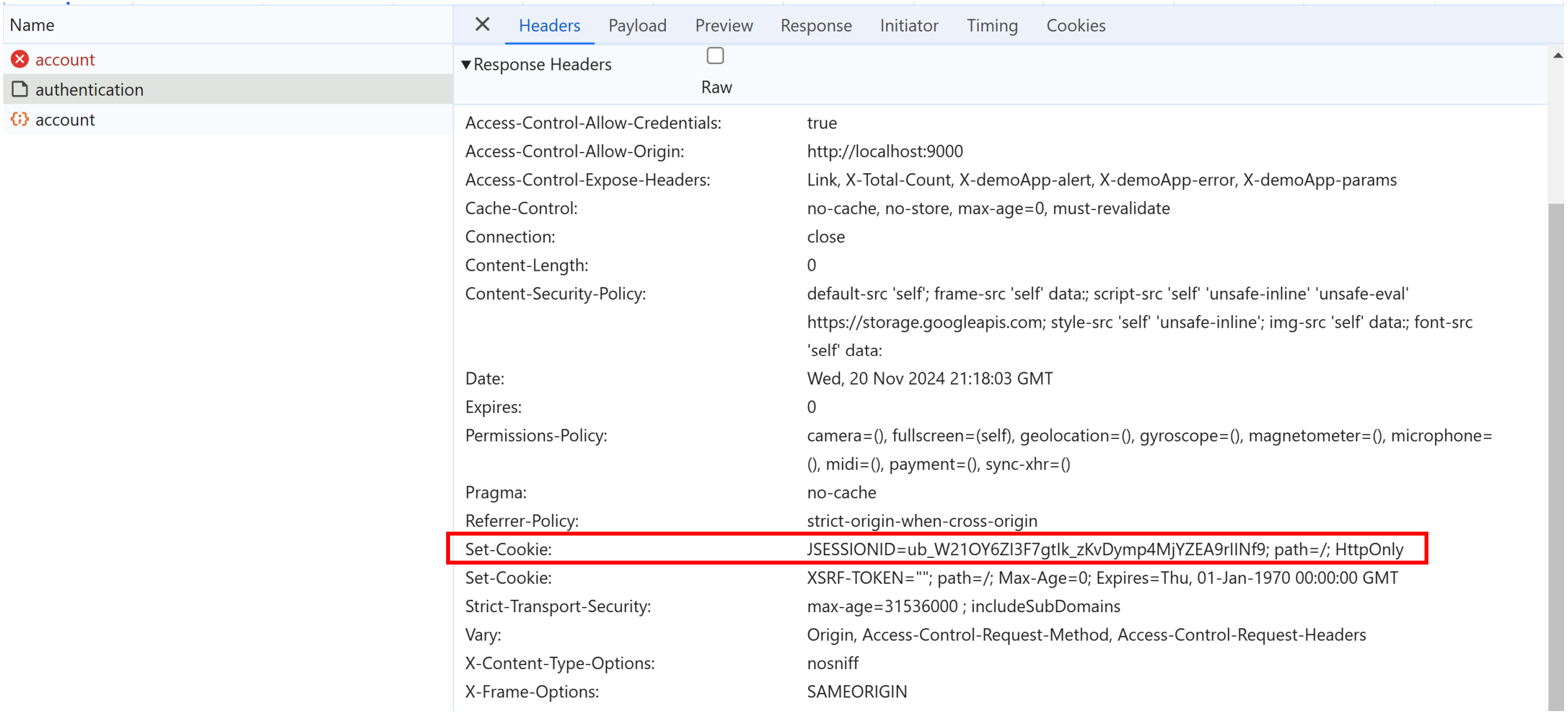




Request [2/3] Response Body : EMPTY !

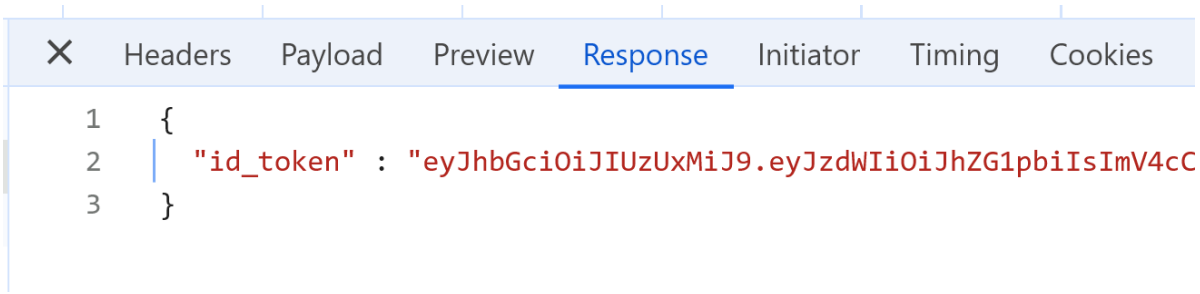


Request [2/3] : Response Header Set-Cookie: "JSESSIONID=xxxx"



Request [2/3] Difference with JWT Mode

"**JWT**" Mode ⟺ "**Http Session**" Mode

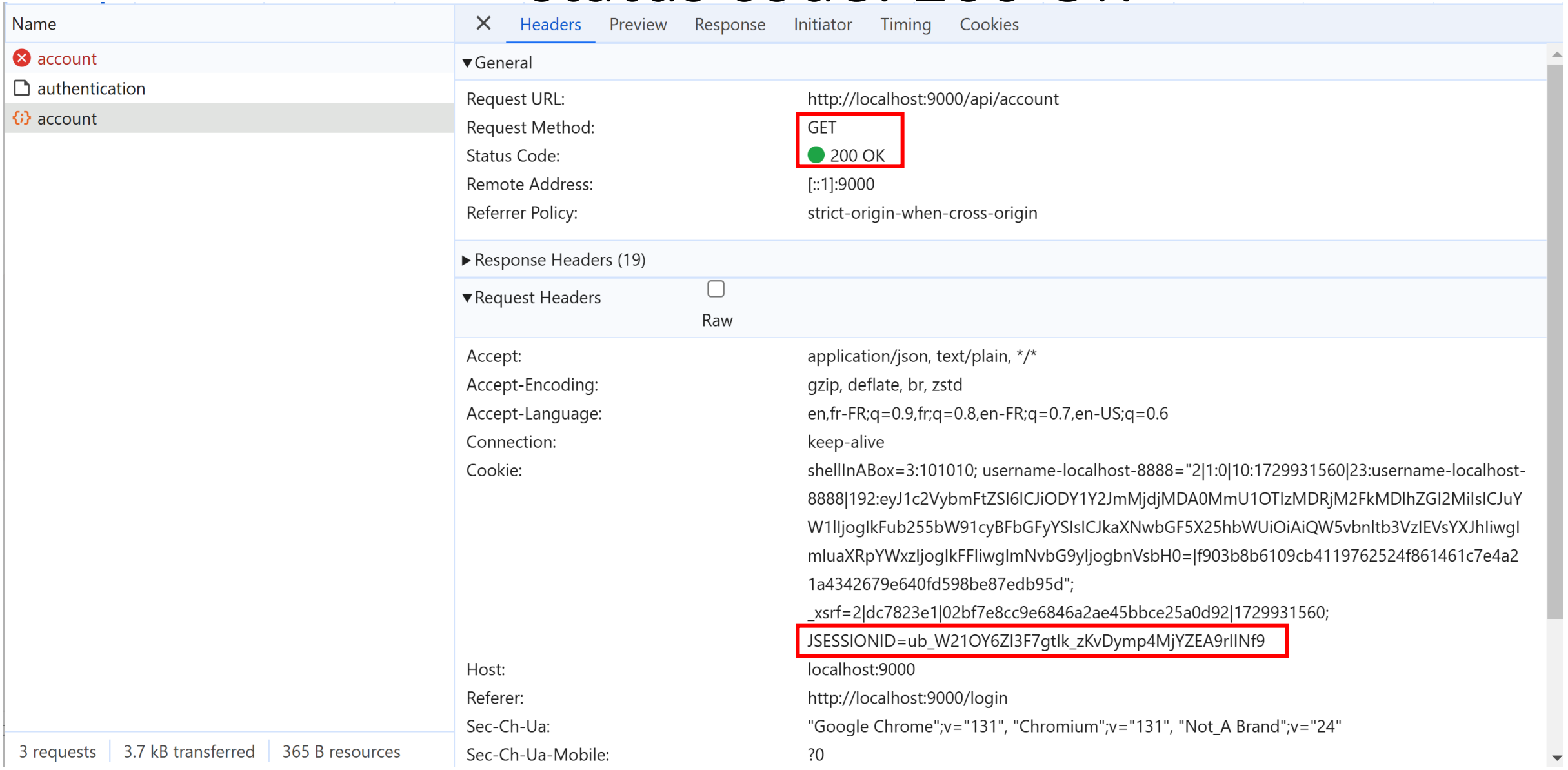


Response Body was in JSON,
containing the JWT Token

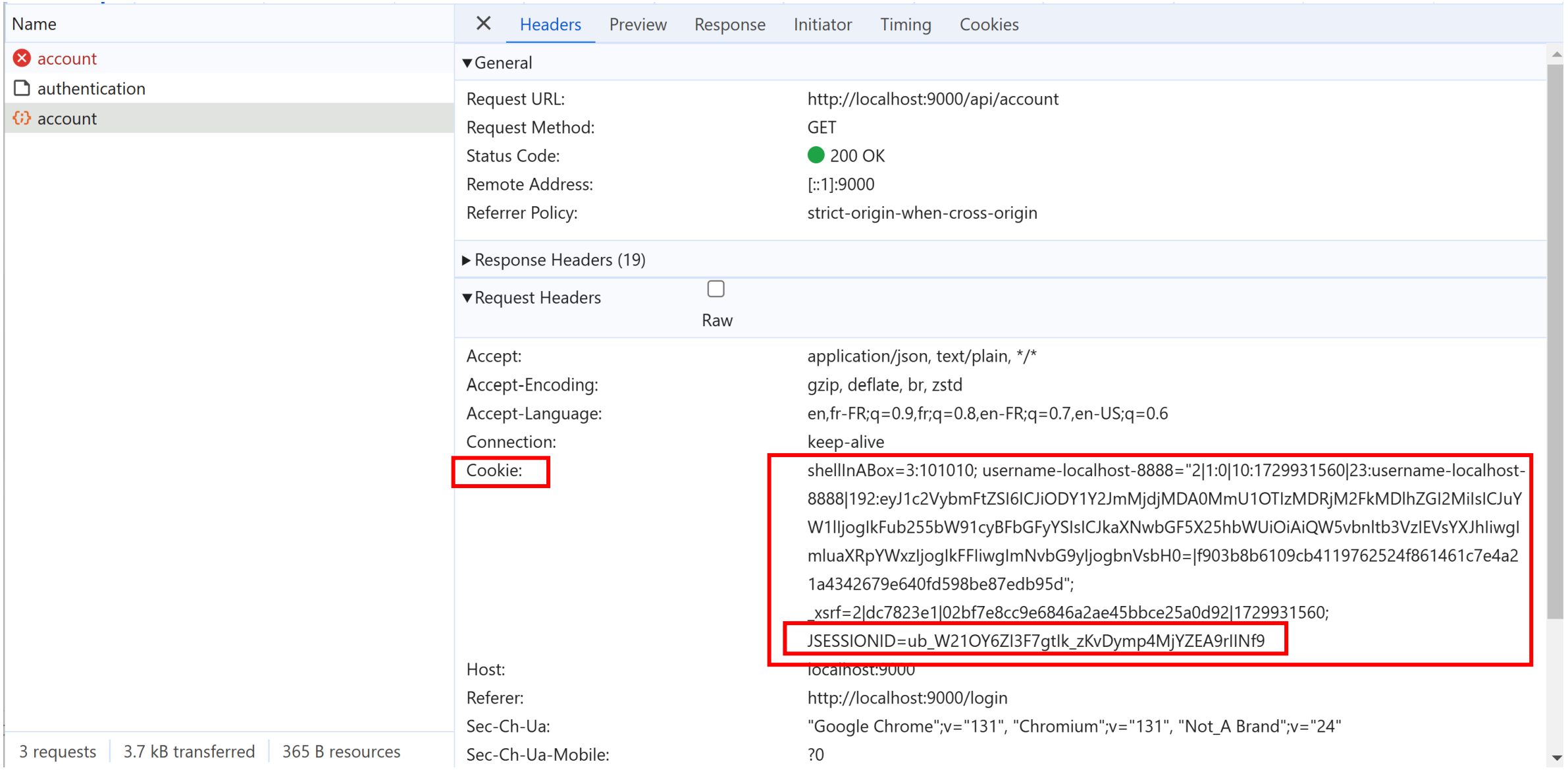
Response Body is EMPTY
the real response is in "SET-COOKIE" Header
to set "JSESSIONID=xxx"

the JSESSIONID is the secret

Request [3/3]: Http GET /api/account => status code: 200 OK



Difference between Failing Request [1/3] and Success Request [3/3] (Authenticated)



Difference Authentication with JWT Mode

"**JWT**" Mode  "**Http Session**" Mode

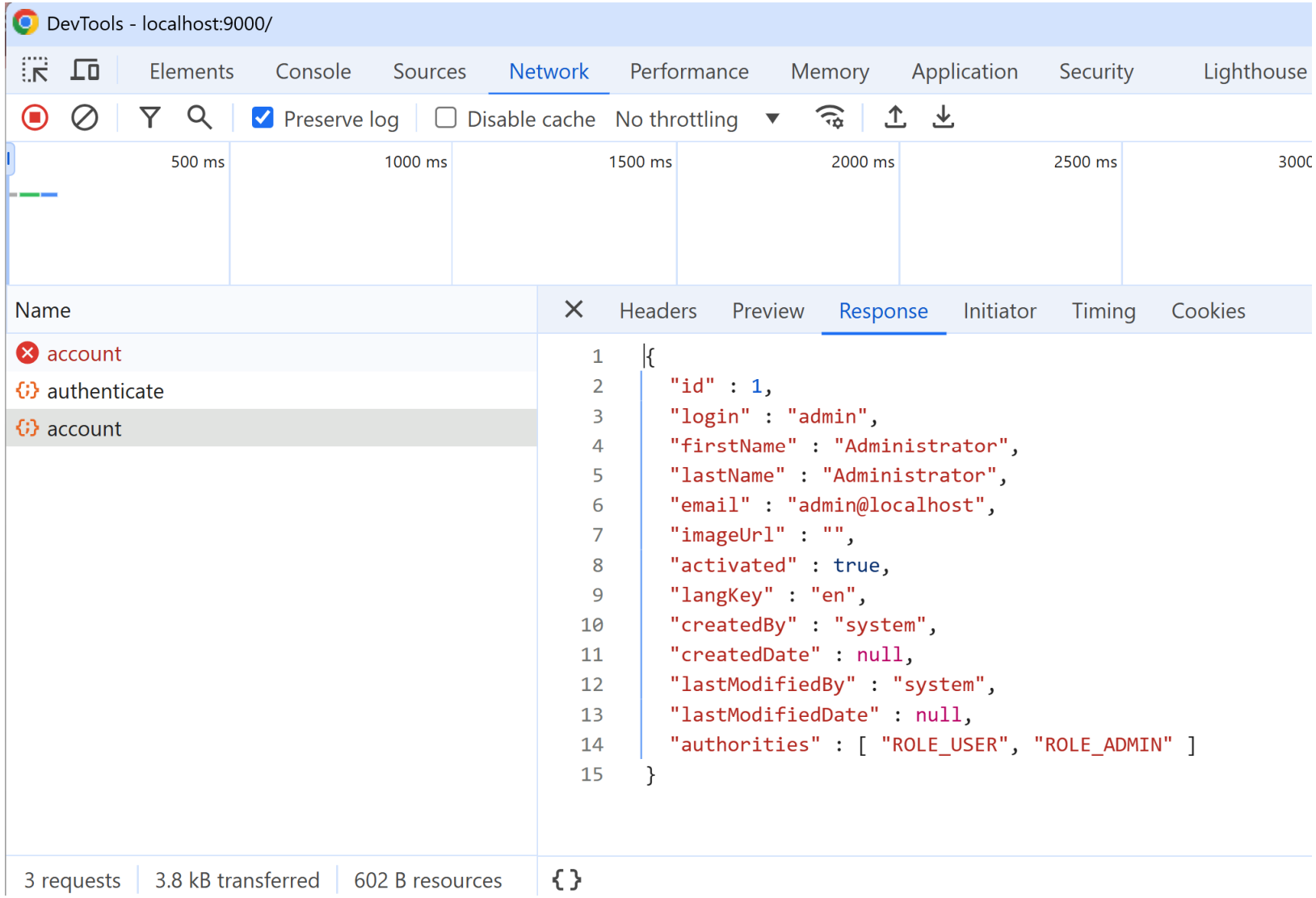
Authorization: Bearer
eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJhZG1pbiIsImV4cCI6MTcz
U0VSIiwiaWF0IjoxNzMyMDQ3OTA0fQ.44dwV-tYt34U5CpaS
OGQKph1BprwpWYt5whgKsTCCiINi4lvacdqM2eRTJ30NsA

Cookie: shellInABox=3:101010; username-localhost-8888="2|1:0|10:1729931560
8888|192:eyJ1c2VybmFtZSI6ICJiODY1Y2JmMjdjMDA0MmU1OTIzMDRjM
W1lIjogIkFub255bW91cyBFbGFyYSIsICJkaXNwbGF5X25hbWUiOiAiQW5
mluaXRpYWxzIjogIkFFIiwgImNvbG9yIjogbnVsbH0=|f903b8b6109cb411
1a4342679e640fd598be87edb95d";
_xsrf=2|dc7823e1|02bf7e8cc9e6846a2ae45bbce25a0d92|1729931560;
JSESSIONID=ub_W21OY6ZI3F7gtIk_zKvDymp4MjYZEA9rIINf9

the JWT secret is passed via Http Header
"**Autorization**": "**Bearer {token}**"

the JSESSIONID secret is passed via Http Header
"**Cookie**" : "**key=value; .... ; JSESSIONID={sessionid}**"

Request [3/3]: /api/account Response Body: Detailed Information on Account



NO Difference for Authorization with JWT Mode

"**JWT**" Mode  "**Http Session**" Mode

NO difference

in both cases, ROLE_* are retreived from Http GET "/api/account"

They were redundantly present in JWT Token, but not used

Backend Code for Http POST /api/authenticate ... framework code, rely on spring UserDetailService

SecurityConfiguration.java

```
38      public class SecurityConfiguration {                                                    2
58          public SecurityFilterChain filterChain(HttpSecurity http, MvcRequestMatcher.Builder mvc) throws Exception {
108             .exceptionHandling( ExceptionHandlingConfigurer<HttpSecurity> exceptionHanding ->
112                 )
113             )
114             .formLogin( FormLoginConfigurer<HttpSecurity> formLogin ->
115                 formLogin
116                     .loginPage("/")
117                     .loginProcessingUrl("/api/authentication")
118                     .successHandler(( HttpServletRequest request,  HttpServletResponse response,  Authentication authentication) -> resp
119                     .failureHandler(( HttpServletRequest request,  HttpServletResponse response,  AuthenticationException exception) -> r
120                     .permitAll()
121             )
122             .logout( LogoutConfigurer<HttpSecurity> logout ->
123                 logout.logoutUrl("/api/logout").logoutSuccessHandler(new HttpStatusReturningLogoutSuccessHandler()).permitAll()
124             );
```

Backend Code for Filter Request Authentication + JSESSIONID ...

???? This is NOT EVEN in spring framework

It is handled directly by the "Servlet" engine (Tomcat / Jetty / ..)

Difficulties for Load-Balancing on Many SErvers with COOKIE "JSESSIONID"

The "Http Session" mode with Cookie JSESSIONID is deprecated

It existed from long time ago, directly in Tomcat, and in servlet spec

It is NOT very efficient, and impose constraints on the way the servers save JSESSIONID

If you have only 1 server => OK, fine (JSESSIONID can be in-memory)
But if you have a load-balancer and many servers => how to share ?...

Angular Code for low-level Http POST /api/authenticate

auth-session.service.ts

```
1   import ...
8
9   @Injectable({ providedIn: 'root' })  Show usages
10  export class AuthServerProvider {
11    private readonly http : HttpClient  = inject(HttpClient);
12    private readonly applicationConfigService : ApplicationConfigService  = inject(ApplicationConfigService);
13
14    login(credentials: Login): Observable<{}> {  Show usages
15      const data : string  =
16        `username=${encodeURIComponent(credentials.username)}` +
17        `&password=${encodeURIComponent(credentials.password)}` +
18        `&remember-me=${credentials.rememberMe ? 'true' : 'false'}` +
19        '&submit=Login';
20
21      const headers : HttpHeaders  = new HttpHeaders().set('Content-Type', 'application/x-www-form-urlencoded');
22
23      return this.http.post(this.applicationConfigService.getEndpointFor('api/authentication'), data, { headers });
24    }
```

NO Need of Custom "HttpInterceptor" for transmitting the Cookie !

index.ts

```ts
import ...

export const httpInterceptorProviders : ({ provide: InjectionToken<readc
  {
    provide: HTTP_INTERCEPTORS,
    useClass: AuthExpiredInterceptor,
    multi: true,
  },
  {
    provide: HTTP_INTERCEPTORS,
    useClass: ErrorHandlerInterceptor,
    multi: true,
  },
  {
    provide: HTTP_INTERCEPTORS,
    useClass: NotificationInterceptor,
    multi: true,
  },
];
```

Difference HttpInterceptors with JWT

"**JWT**" Mode ⟺ "**Http Session**" Mode

```
7
8   export const httpInterceptorProviders : ({ provide: Injec
9     {
10      provide: HTTP_INTERCEPTORS,
11      useClass: AuthInterceptor,
12      multi: true,
13    },
14    {
15      provide: HTTP_INTERCEPTORS,
16      useClass: AuthExpiredInterceptor,
17      multi: true,
18    },
19    {
20      provide: HTTP_INTERCEPTORS,
21      useClass: ErrorHandlerInterceptor,
22      multi: true,
23    },
24    {
25      provide: HTTP_INTERCEPTORS,
26      useClass: NotificationInterceptor,
27      multi: true,
28    },
29  ];
```

index.ts

```
1   import ...
6
7   export const httpInterceptorProviders : ({ provide: InjectionToken<readc
8     {
9       provide: HTTP_INTERCEPTORS,
10      useClass: AuthExpiredInterceptor,
11      multi: true,
12    },
13    {
14      provide: HTTP_INTERCEPTORS,
15      useClass: ErrorHandlerInterceptor,
16      multi: true,
17    },
18    {
19      provide: HTTP_INTERCEPTORS,
20      useClass: NotificationInterceptor,
21      multi: true,
22    },
23  ];
```

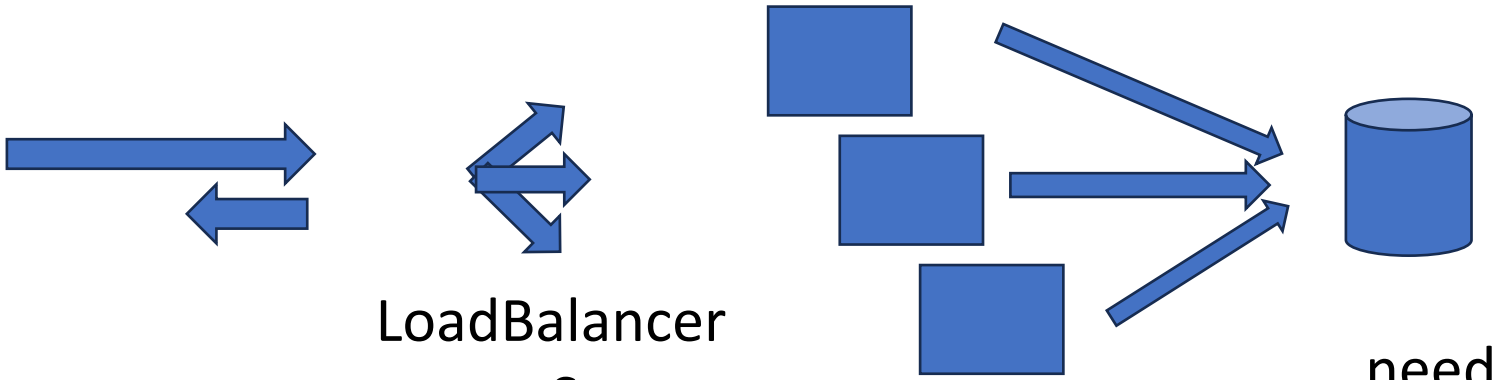
Authorisations

For Authorization (ROLE_*)
there is NO difference with the others mode JWT or OAuth2.

Please see previous detailed document on JWT

Network & Infrastructure Deployments Options

Servers are Statefull - Need Centralised Database

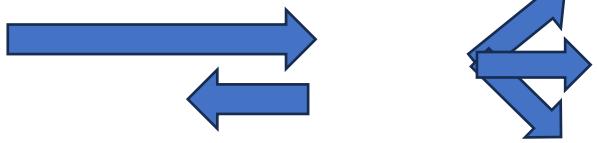


LoadBalancer
to any Server

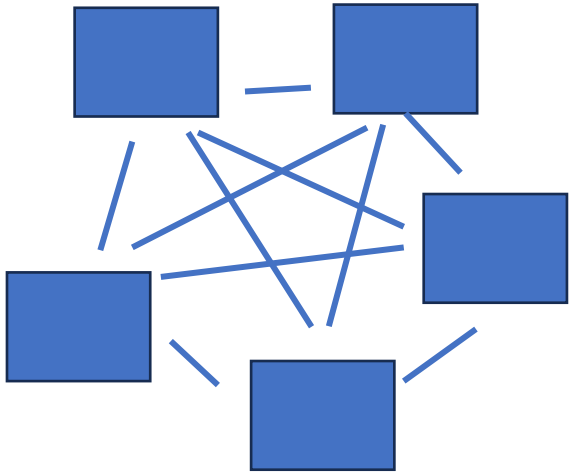
Statefull servers

need centralised Database !!
performance Bottleneck
& SPOF = "Single Point Of Failure"

Server Connected in Cluster "Distributed Cache" (=idem embedded database)



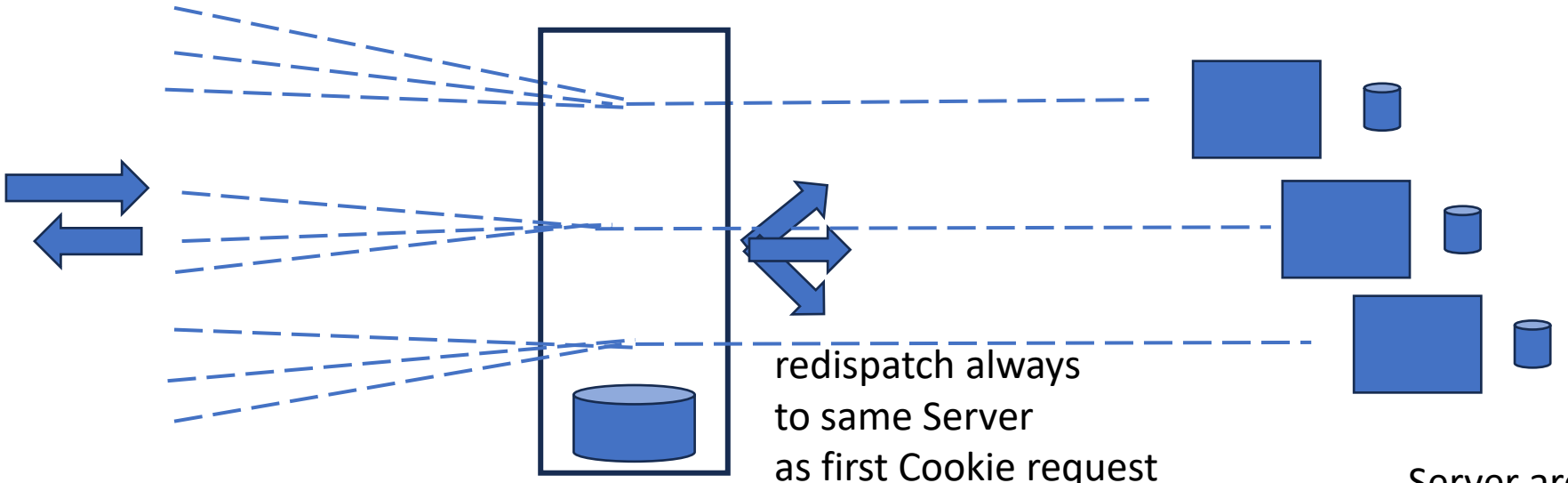
LoadBalancer
to any Server



Cluster of
Statefull servers

NO "SPOF"
(Single Point Of Failure")
But very complex

LoadBalancer with "Session Stickiness"
Statefull Server
(local DB or in-memory not persisted)



redispatch always
to same Server
as first Cookie request

LoadBalancer
with SessionStickiness

Server are Statefull

If Cookie are only "IN-MEMORY"
=> server crash = losing cookies

Next Steps

Previous document
1/3 : Debugging "JWT" security mode

This document
[2/3]: Debugging old-style COOKIE "jsessionId" security mode

Next document:
[3/3] Debugging the OAuth2 security mode

Questions

arnaud.nauwynck@gmail.com