Angular WebApp Security with JWT

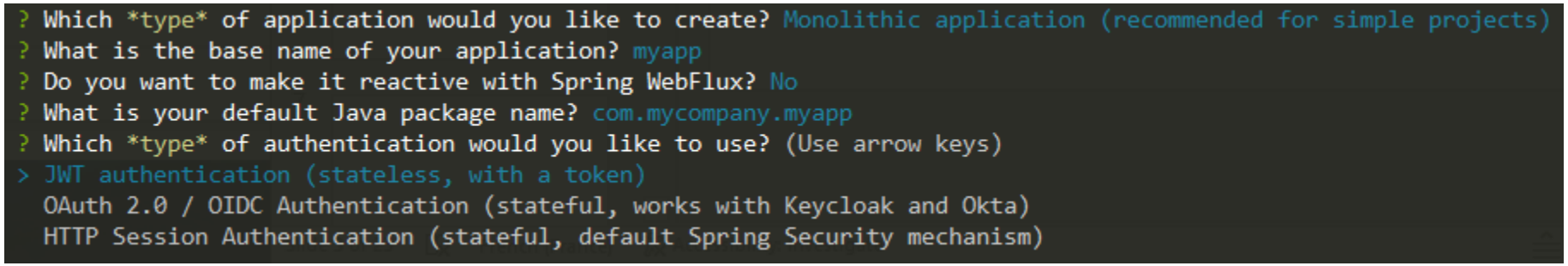
(Reverse engineering Jhipster Generated Code)

arnaud.nauwynck@gmail.com

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

Step 1: create Jhipster app

Choose App Type > Security Type



End Choices (Db, Angular, Maven, ..)

```
? Which *type* of database would you like to use? SQL (H2, PostgreSQL, MySQL, MariaDB, Oracle, MSSQL)
? Which *production* database would you like to use? PostgreSQL
? Which *development* database would you like to use? H2 with disk-based persistence
? Which cache do you want to use? (Spring cache abstraction) No cache - Warning, when using an SQL database, this will disable the Hibernate 2nd level cache!
? Would you like to use Maven or Gradle for building the backend? Maven
? Do you want to use the JHipster Registry to configure, monitor and scale your application? No
? Which other technologies would you like to use? WebSockets using Spring Websocket
? Which *Framework* would you like to use for the client? Angular
? Do you want to generate the admin UI? Yes
? Would you like to use a Bootswatch theme (https://bootswatch.com/)? Default JHipster
? Would you like to enable internationalization support? No
? Please choose the native language of the application English
? Besides JUnit and Jest, which testing frameworks would you like to use?
? Would you like to install other generators from the JHipster Marketplace? No
```

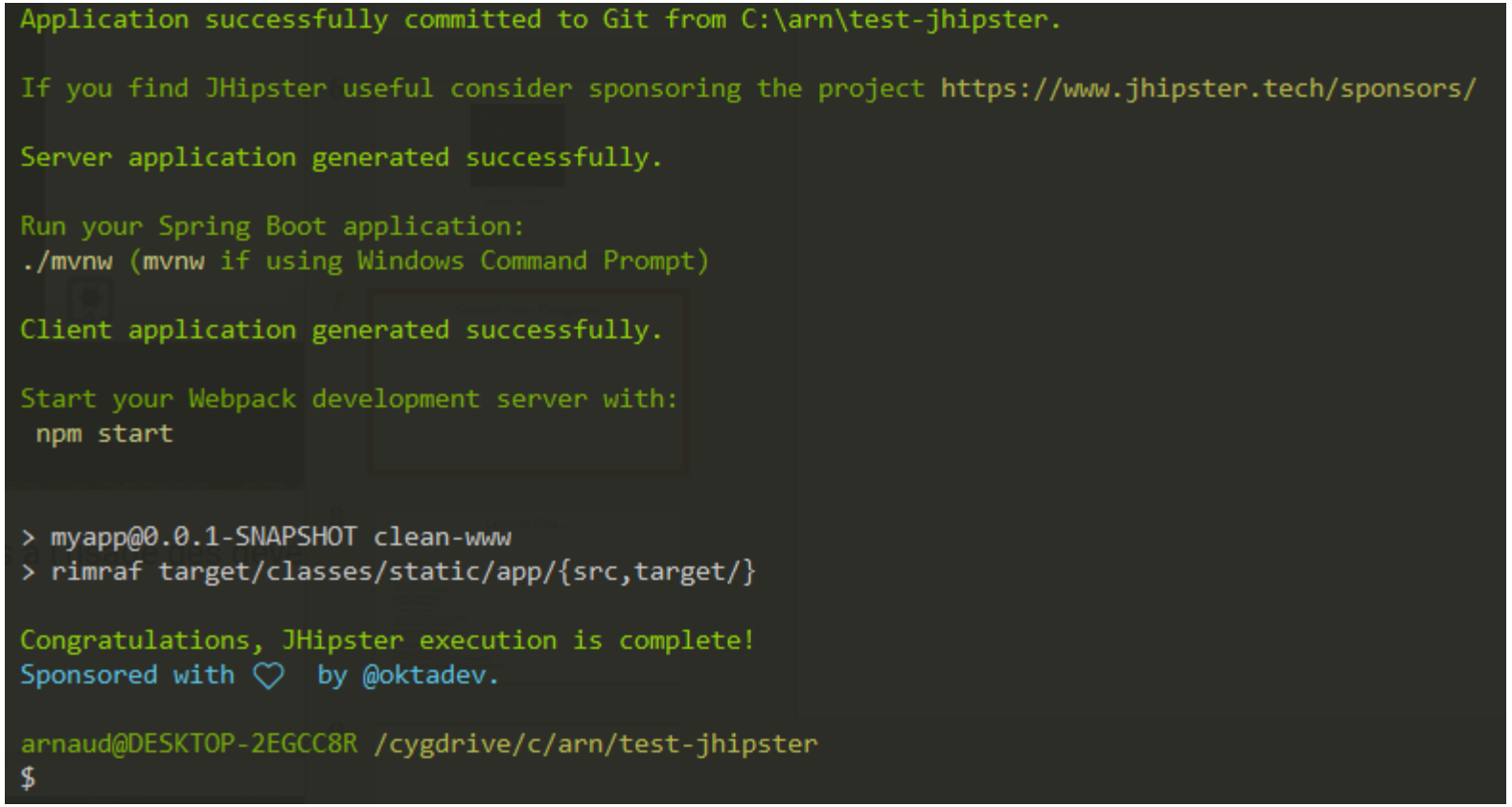
Start Generating

```
G□n□ration d'une paire de cl□s RSA de 2?048 bits et d'un certificat auto-sign□ (SHA256withRSA) d'une validit□ de 99?999 jours
        pour : CN=Java Hipster, OU=Development, O=com.mycompany.myapp, L=, ST=, C=

KeyStore 'C:\arn\test-jhipster\src\main\resources\config\tls\/keystore.p12' generated successfully.
```

Generating a RSA Key pair, for exposing https instead of http

(but self-signed certificate)

Finished Generating



Created Files

```
create .prettierrc
create .prettierignore
create package.json
 force .yo-rc.json
 force .yo-rc-global.json
create .gitattributes
create .editorconfig
create .gitignore
create sonar-project.properties
create .husky\pre-commit
create .lintstagedrc.js
create mvnw
create mvnw.cmd
create .mvn\jvm.config
create .mvn\wrapper\maven-wrapper.jar
create .mvn\wrapper\maven-wrapper.properties
create npmw
create npmw.cmd
create src\main\resources\banner.txt
```

(truncated …) total files :

```
$ find . | wc -l
105964
```

Lot of npm files (in node_modules/**)
… but ok .gitignore

```
$ git status
On branch master
nothing to commit, working tree clean
```

```
$ git log
commit 86ba746ae24834926b6b79e85cecef20481d44ab (HEAD -> master)
Author: Arnaud Nauwynck <arnaud.nauwynck@gmail.com>
Date:   Thu May 19 14:13:09 2022 +0200

    Initial version of myapp generated by generator-jhipster@7.8.1
```

```
$ cat .gitignore
######################
# Project Specific
######################
/src/main/webapp/content/css/main.css
/target/classes/static/**
/src/test/javascript/coverage/

######################
# Node
######################
/node/
node_tmp/
node_modules/
npm-debug.log.*
/.awcache/*
/.cache-loader/*
```

```
$ ls -1
README.md
angular.json
checkstyle.xml
jest.conf.js
mvnw
mvnw.cmd
ngsw-config.json
node_modules
npmw
npmw.cmd
package-lock.json
package.json
pom.xml
sonar-project.properties
src
target
tsconfig.app.json
tsconfig.json
tsconfig.spec.json
webpack
```

```
$ find . -maxdepth 1 -type d
.
./.devcontainer
./.git
./.husky
./.mvn
./node_modules
./src
./target
./webpack
```

Src Files

```
$ find src/main | wc -l
464
```

```
$ find src/main/java | wc -l
92

$ find src/main/webapp | wc -l
322

$ find src/main/resources | wc -l
29

$ find src/main/docker | wc -l
20
```

Created Files - Categories

```
$ find . -maxdepth 1 -type d
.
./.devcontainer
./.git
./.husky
./.mvn
./node_modules
./src
./target
./webpack
```

src/
(in git)

```
$ find ./src -maxdepth 2
./src
./src/main
./src/main/docker
./src/main/java
./src/main/resources
./src/main/webapp
./src/test
./src/test/java
./src/test/resources
```

src/main
(in artifact delivery)

src/test
(during test only)

src/main/webapp

Technical layers organization:
account, admin, config, core, entities, home, layouts, login, shared

```
$ find ./src/main/webapp/app -maxdepth 1
./src/main/webapp/app
./src/main/webapp/app/account
./src/main/webapp/app/admin
./src/main/webapp/app/app-routing.module.ts
./src/main/webapp/app/app.constants.ts
./src/main/webapp/app/app.module.ts
./src/main/webapp/app/config
./src/main/webapp/app/core
./src/main/webapp/app/entities
./src/main/webapp/app/home
./src/main/webapp/app/layouts
./src/main/webapp/app/login
./src/main/webapp/app/shared
```

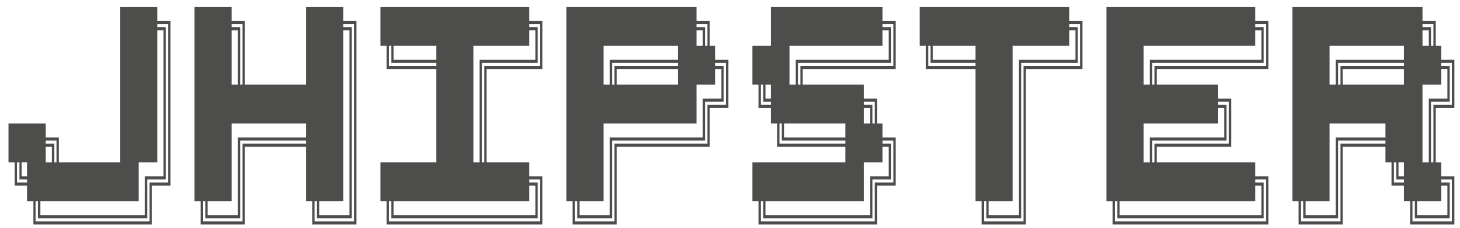
src/mainjava

Technical layers organization:
aop, config, domain, managment, security, service, web

```
$ find ./src/main/java -maxdepth 4
./src/main/java
./src/main/java/com
./src/main/java/com/mycompany
./src/main/java/com/mycompany/myapp
./src/main/java/com/mycompany/myapp/aop
./src/main/java/com/mycompany/myapp/ApplicationWebXml.java
./src/main/java/com/mycompany/myapp/config
./src/main/java/com/mycompany/myapp/domain
./src/main/java/com/mycompany/myapp/GeneratedByJHipster.java
./src/main/java/com/mycompany/myapp/management
./src/main/java/com/mycompany/myapp/MyappApp.java
./src/main/java/com/mycompany/myapp/repository
./src/main/java/com/mycompany/myapp/security
./src/main/java/com/mycompany/myapp/service
./src/main/java/com/mycompany/myapp/web
```

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: []
```



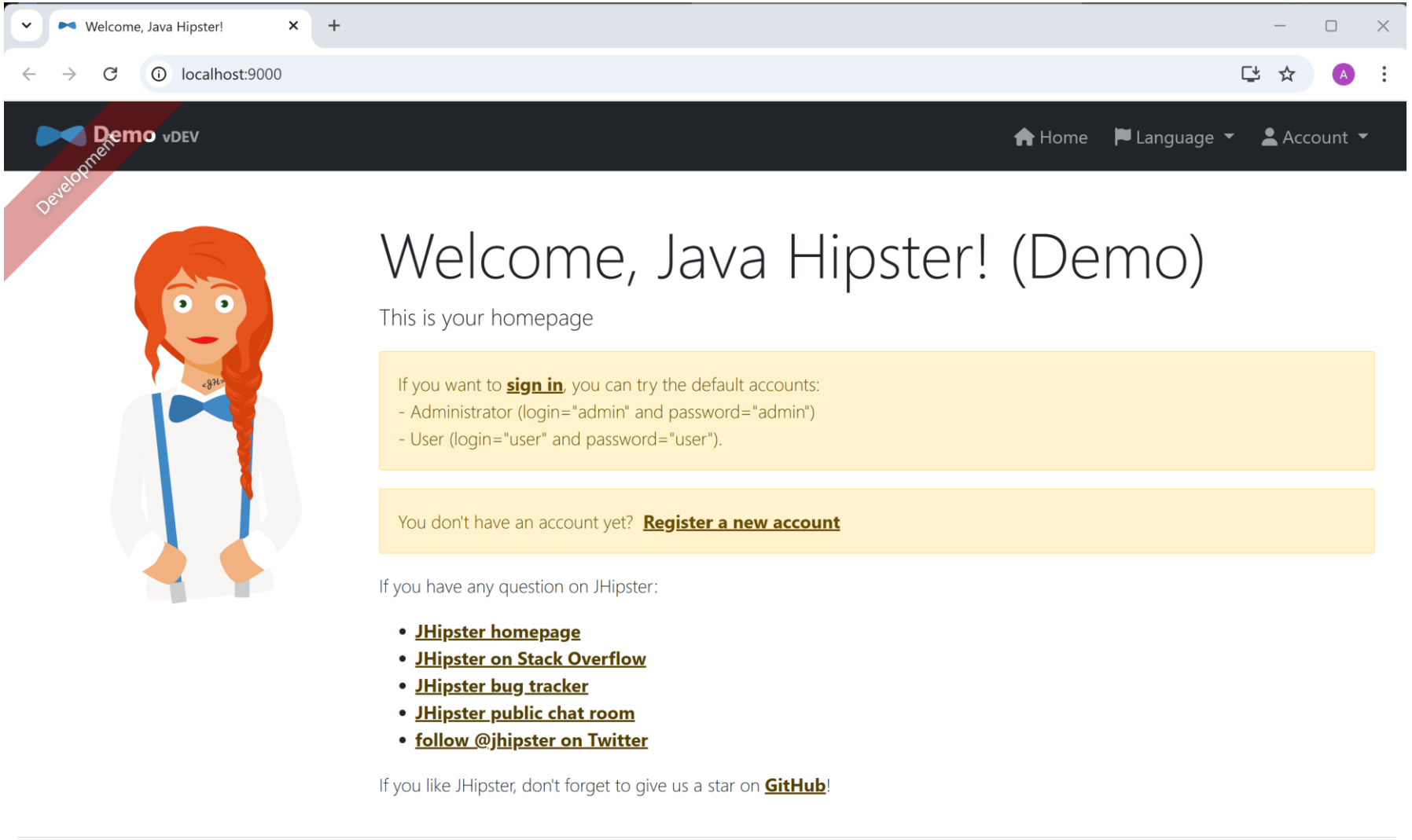

```
2024-11-19T17:05:44.632+01:00  INFO 7000 --- [  restartedMain] com.mycompany.myapp.DemoApp              : Started DemoApp in 11.616 seconds (process runni
ng for 12.128)
2024-11-19T17:05:44.644+01:00  INFO 7000 --- [  restartedMain] com.mycompany.myapp.DemoApp              :
----------------------------------------------------------
        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)

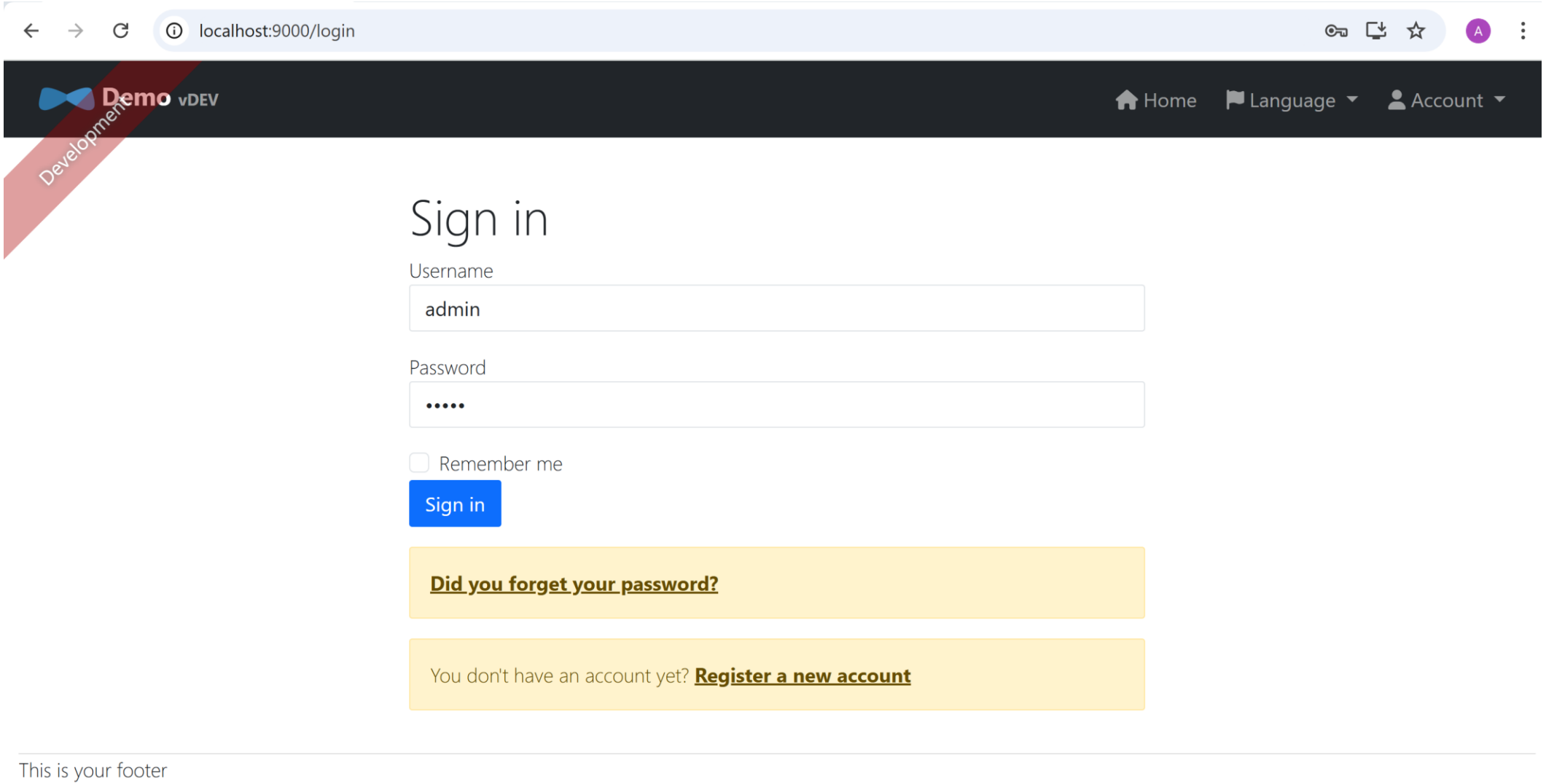
```
$ npm run start

> 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:

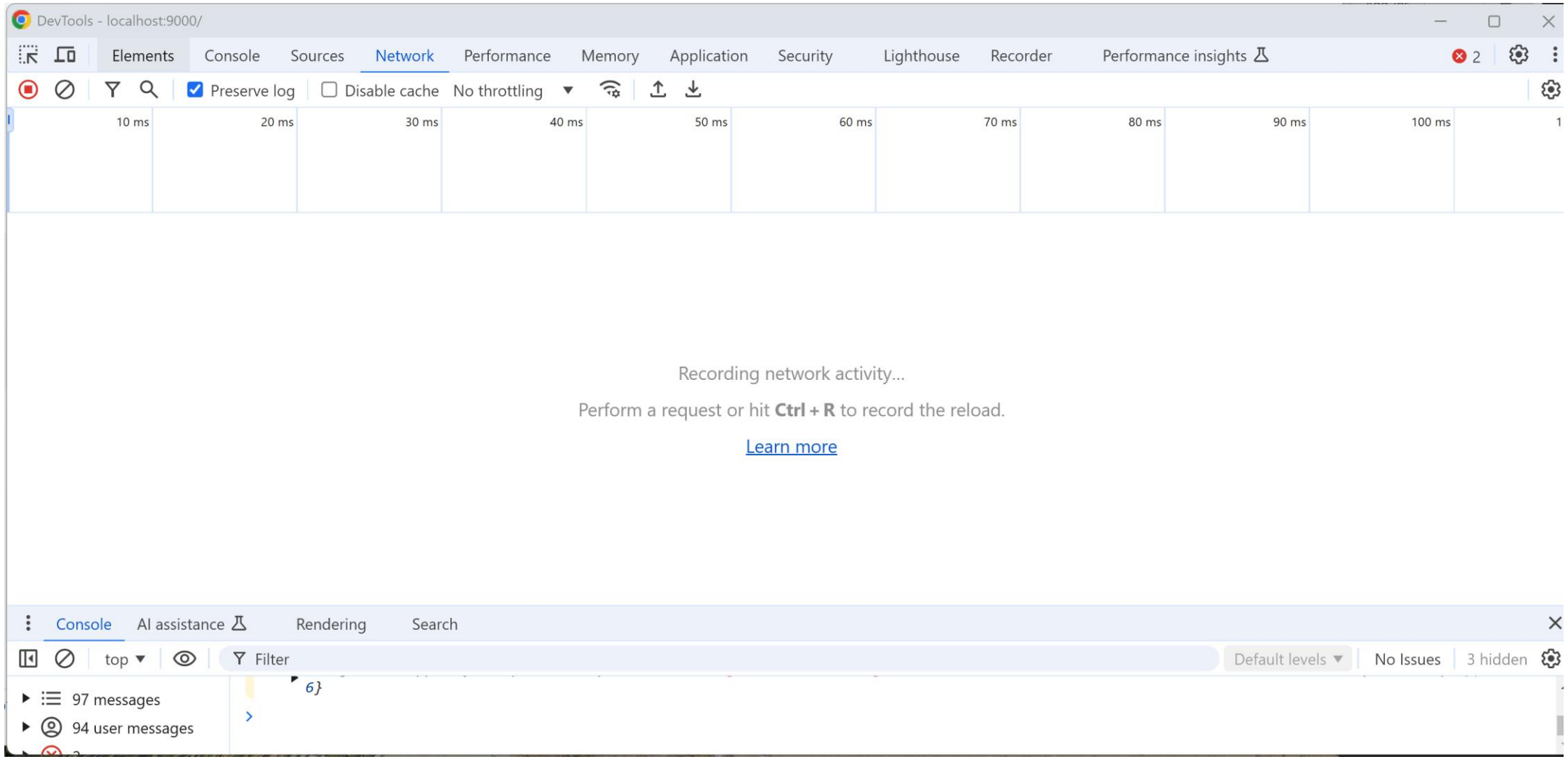
- **JHipster homepage**
- **JHipster on Stack Overflow**
- **JHipster bug tracker**
- **JHipster public chat room**
- **follow @jhipster on Twitter**

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

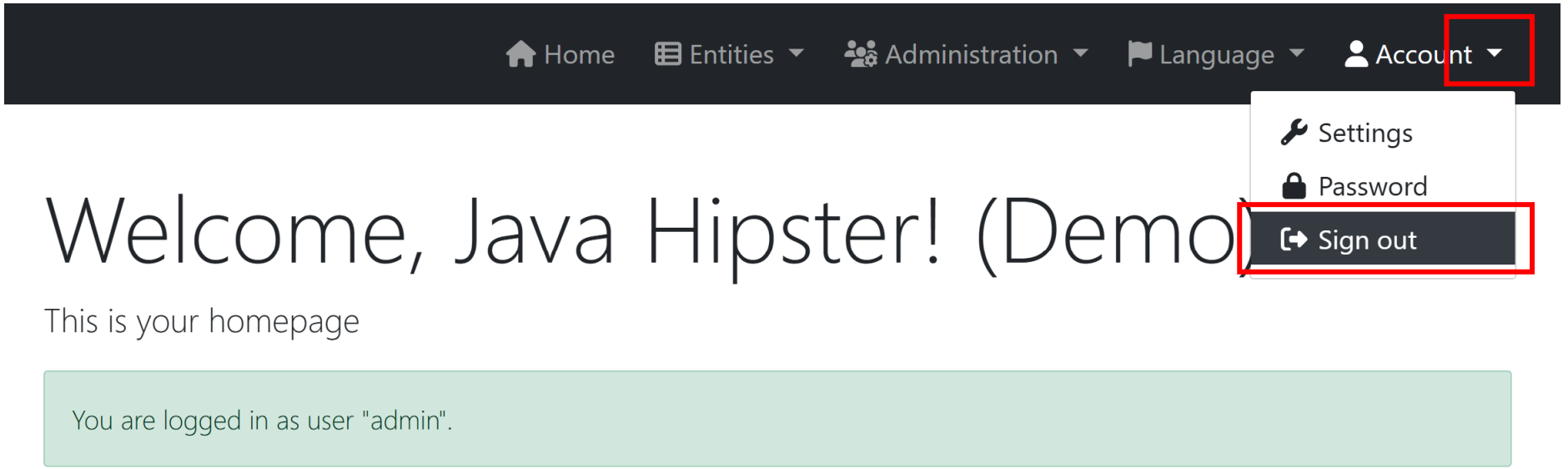
This is your footer

Debuging Security from "Http" viewpoint

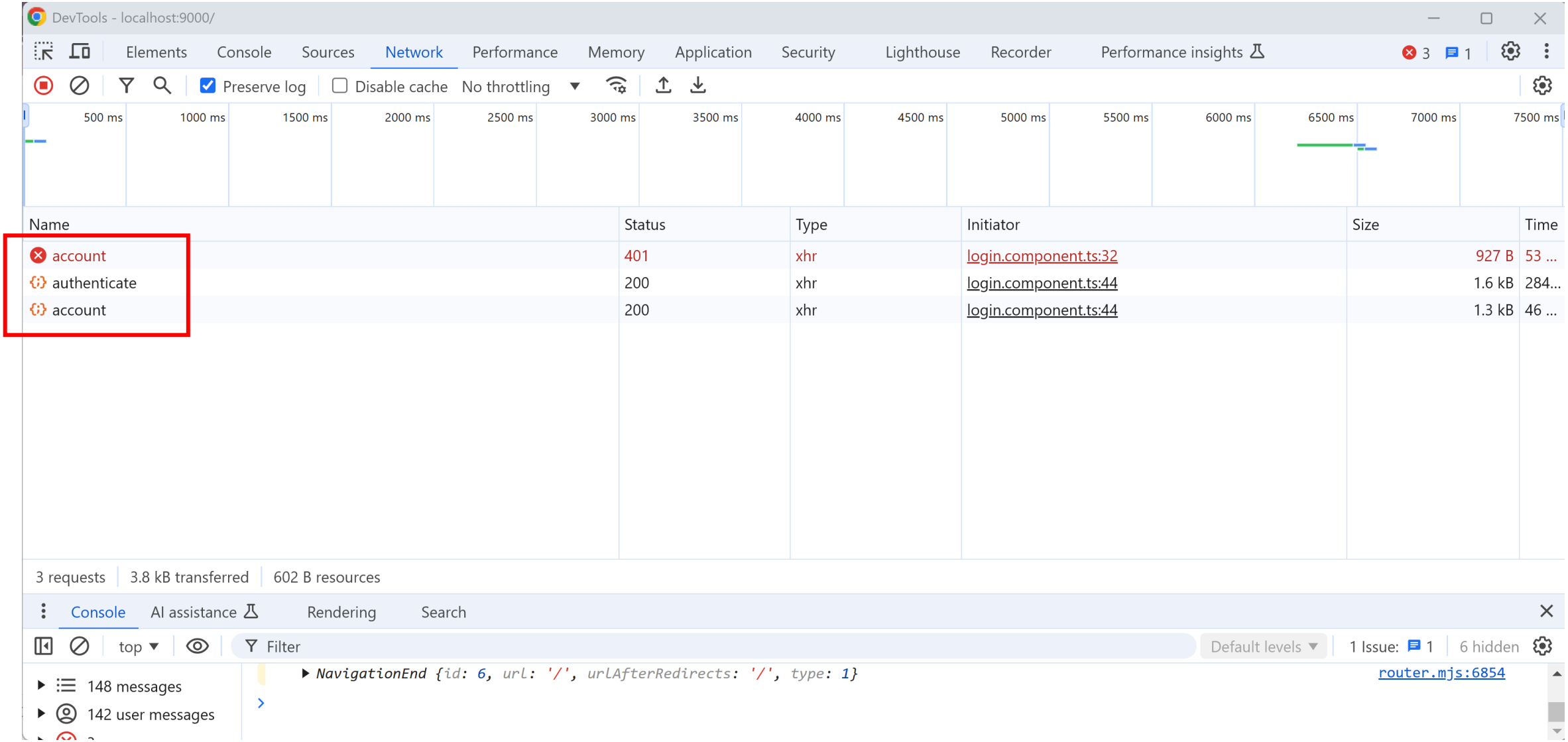
Chrome Dev Tools > Network



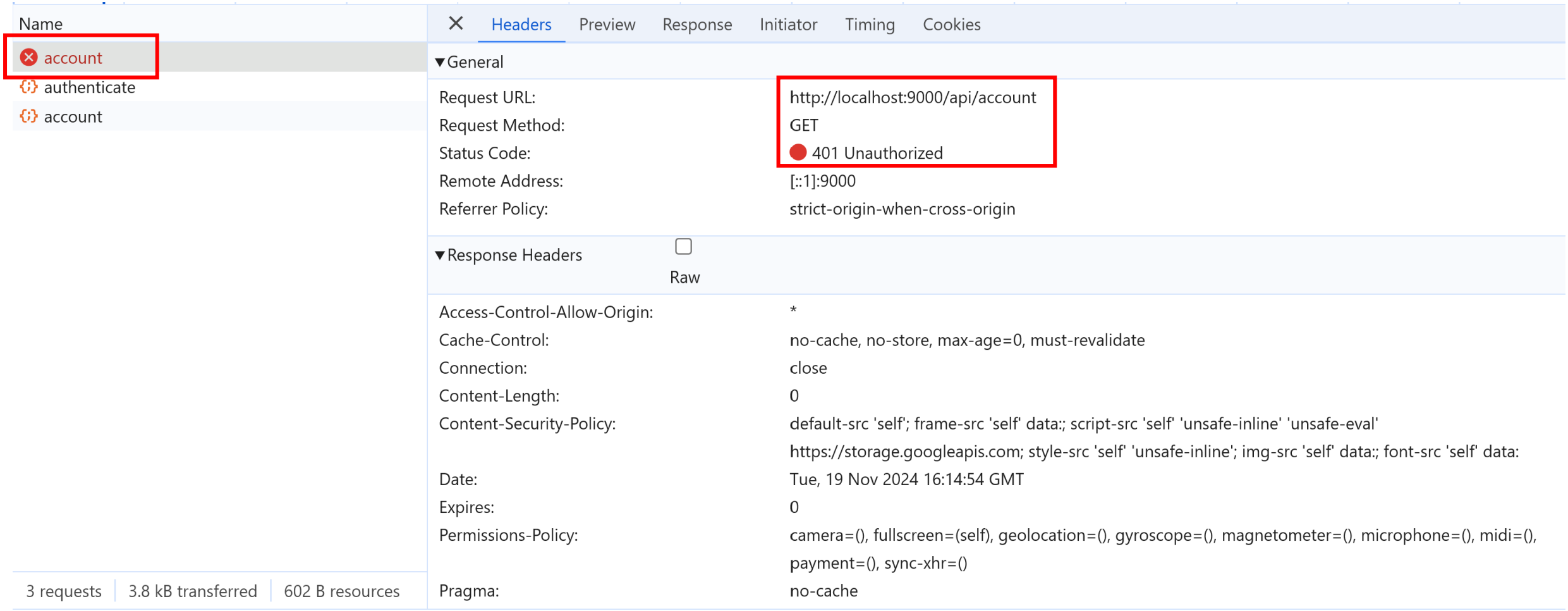
Sign Out => NO Http call



Sign-in => 3 Http Requests



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



MISSING Header "Authorization" => FAILING with 401

Notice : why first Failing http GET /api/account ?

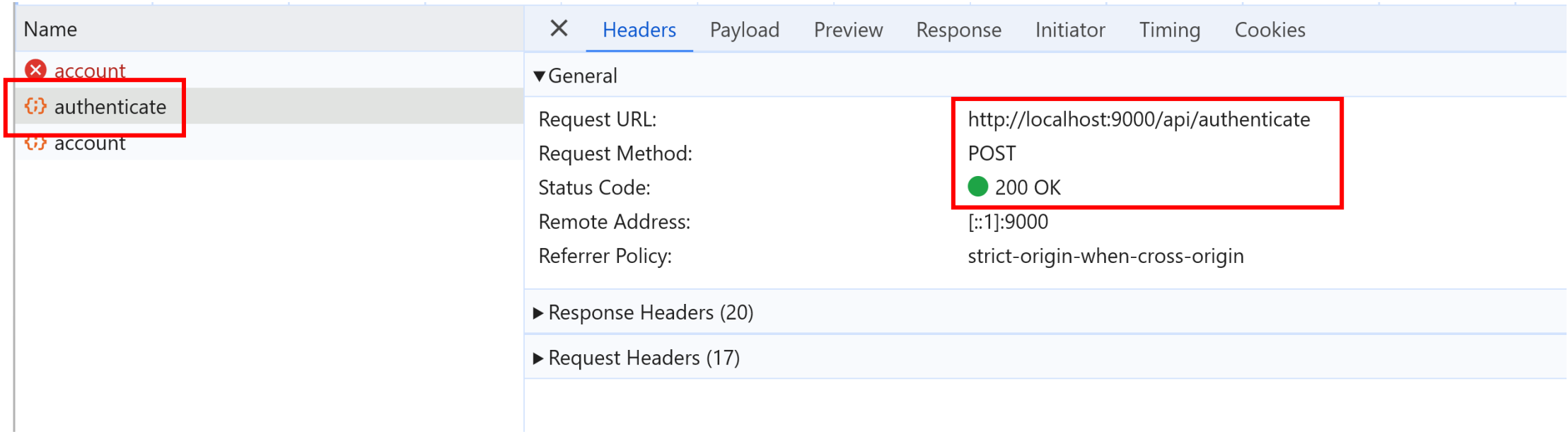
as soon as you click on the "SIGN IN" button
=> the authentication is immediatly cleared
=> JHipster immediatly refresh the current user account (? why? it could be just cleared)

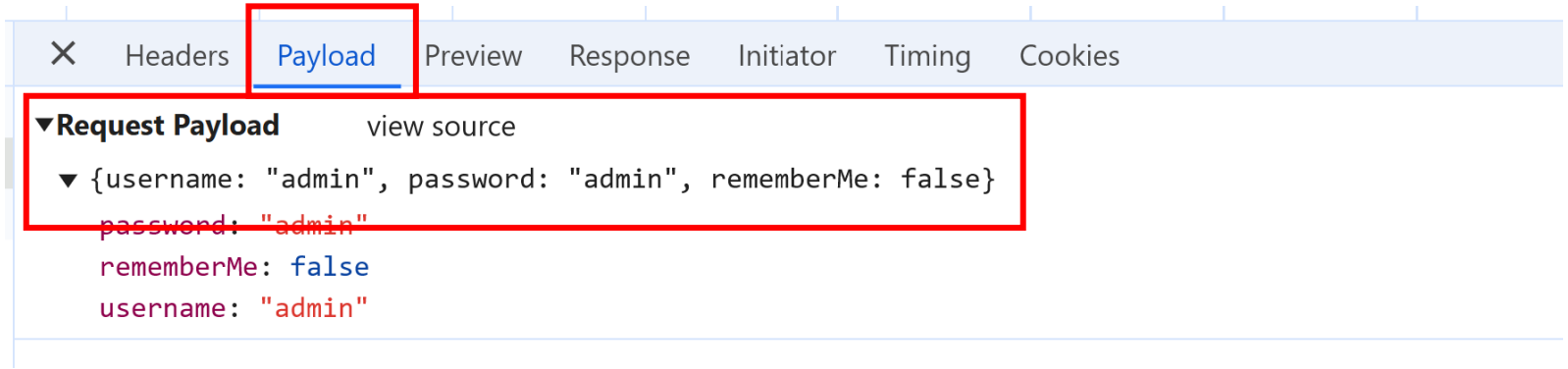
Then there is a pending call to "/api/authenticate"

(This could take for example 2 seconds... In the mid-time, still no calls are possible)

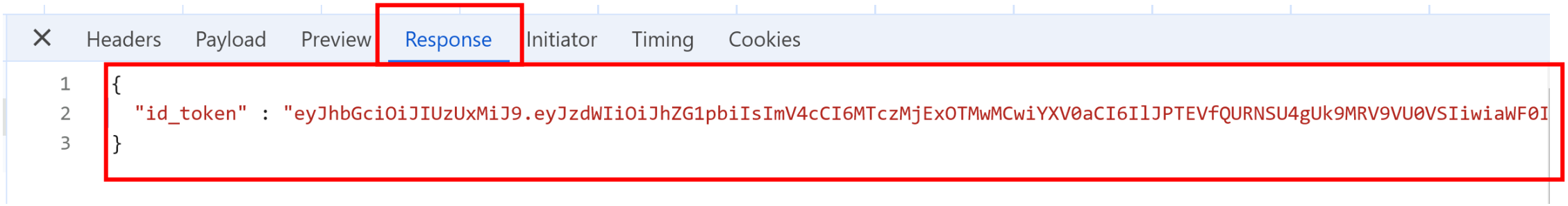
When receiving the "/api/authenticate" response, then immediatly after,
Jhipster redo another refresh for the current user account (this one will succeed)

Request [2/3]: Http POST /api/authenticate with request Body = { username, password }





Request [2/3] Response Body: { jwt token }

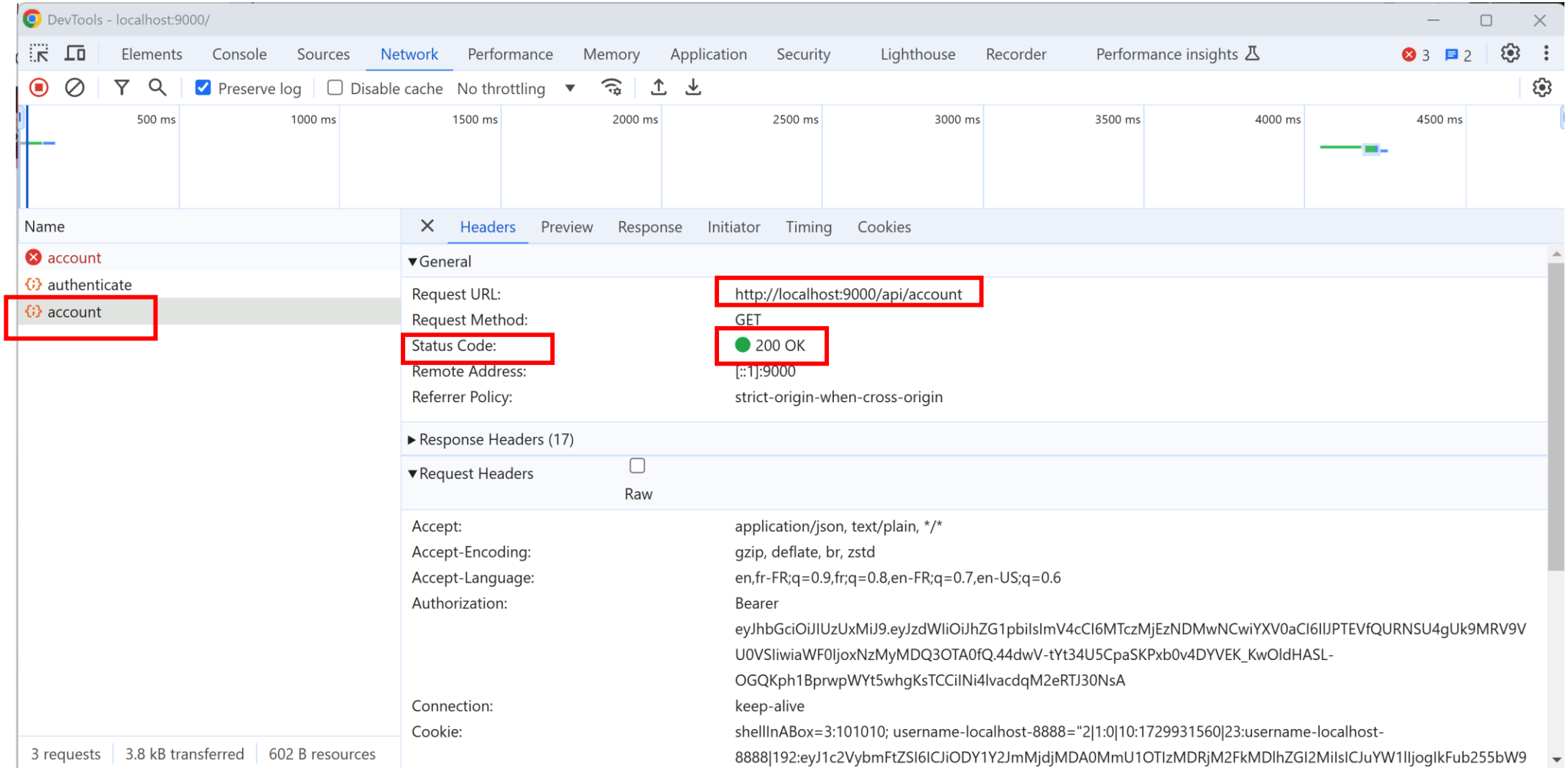


the response is a **JSON** object

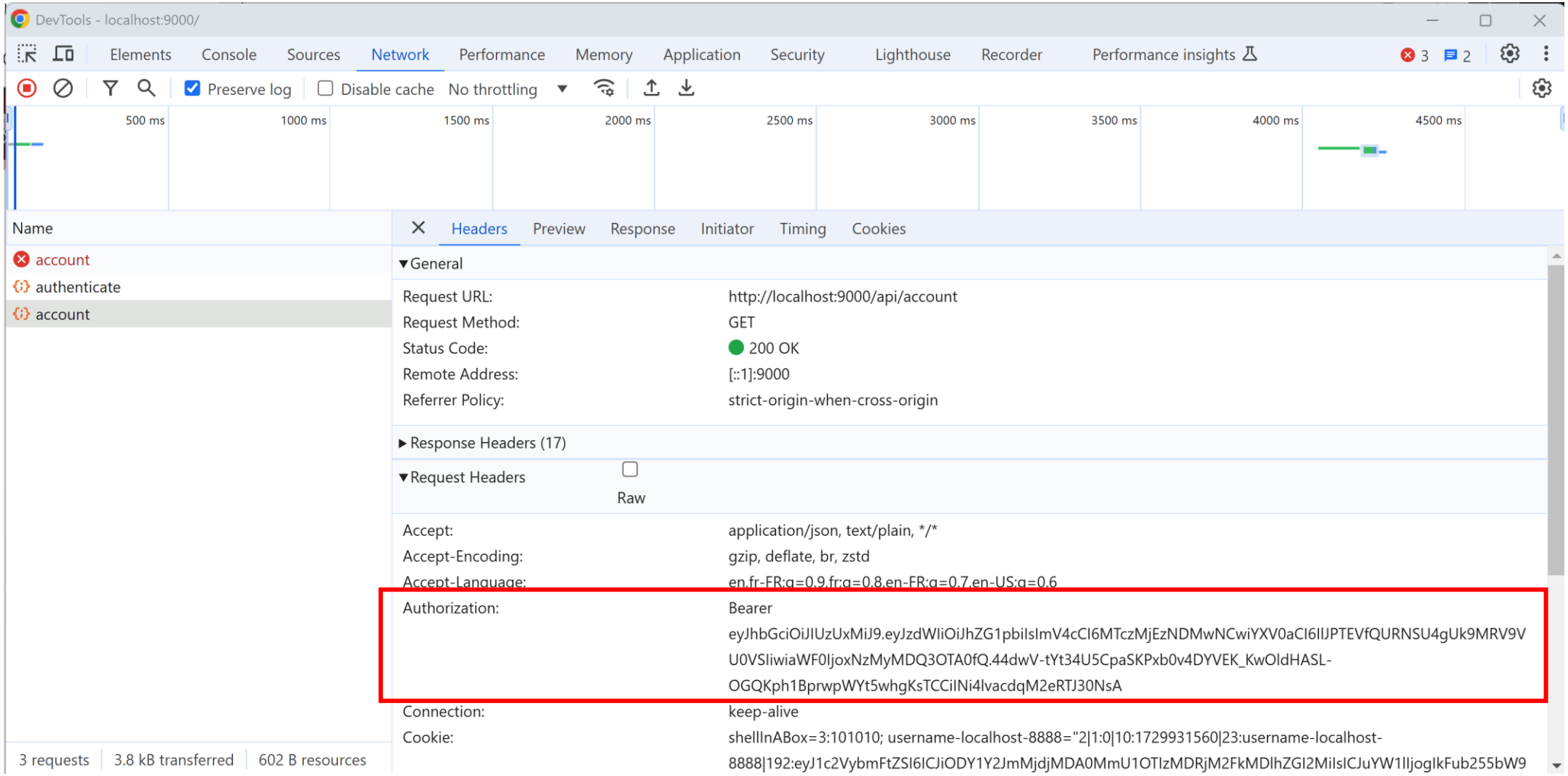
It contains only 1 field, named "**id_token**"

We will see next, that the string value "**eyJH...** " is called a **JWT Token**

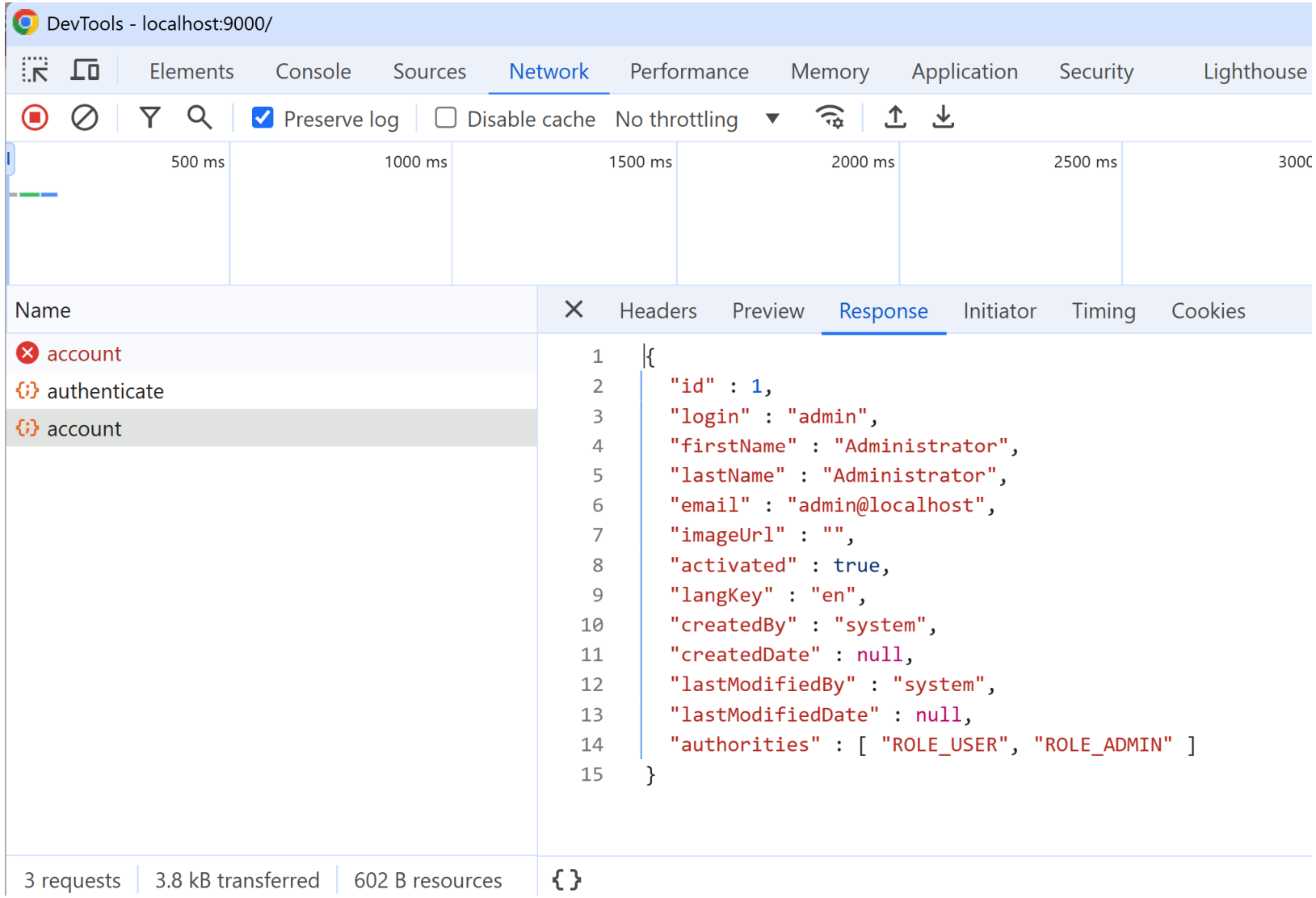
Request [3/3]: Http GET /api/account => status code: 200 OK



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



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



Http Request status 200..
ONLY If authenticated + authorized

Htpp client **always need to pass Http Header "Authorization" = "Bearer ${jwtToken}"**

=> this is for **Authentication** of the user (similar to checking its password)
The Token is a proof of identity

Then, only after, the server can check for Authorization for that calling user on a given action
=> this is "**Authorization ROLES checking**"

Example, some actions may be reserved only for Admins, some other only for normal users.
Most actions on resources depends on both the resource and the corresponding user (without ROLE).

Example: a Customer of a Bank can only view is personnal "Bank Account", not others accounts !!
Even if he has role "ROLE_CUSTOMER"

NOTICE Request [2/3] ... special explicit configuration on server to allow "un-authenticated"

on the backend (java) side, "http POST /api/authenticate" is an exceptional Rest API that accepts being called with un-authenticated mode (= ".permitAll()")

Other similar endpoints are for
- getting public content files,
- health-check, monitoring, ..
- asking an email to "reset your forgotten password"
- asking an email to "register" a new account

Backend Security Config
.permitAll() for special methods

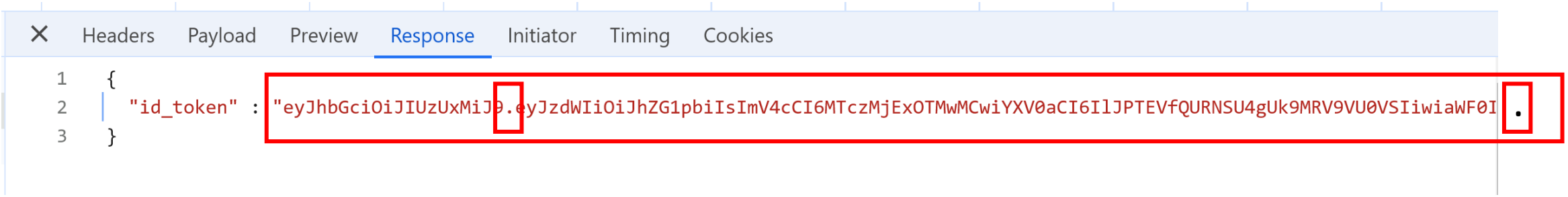
SecurityConfiguration.java

```
31  public class SecurityConfiguration {                                    1 2
48      public SecurityFilterChain filterChain(HttpSecurity http, MvcRequestMatcher.Builder mvc) throws Exception {
63          )
64          .authorizeHttpRequests( AuthorizationManagerRequestMat... authz ->
65              // prettier-ignore
66              authz
67                  .requestMatchers(mvc.pattern("/index.html"), mvc.pattern("/*.js"), mvc.pattern("/*.txt")
68                  .requestMatchers(mvc.pattern("/*.ico"), mvc.pattern("/*.png"), mvc.pattern("/*.svg"), mv
69                  .requestMatchers(mvc.pattern("/app/**")).permitAll()
70                  .requestMatchers(mvc.pattern("/i18n/**")).permitAll()
71                  .requestMatchers(mvc.pattern("/content/**")).permitAll()
72                  .requestMatchers(mvc.pattern("/swagger-ui/**")).permitAll()
73                  .requestMatchers(mvc.pattern(HttpMethod.POST, "/api/authenticate")).permitAll()
74                  .requestMatchers(mvc.pattern(HttpMethod.GET, "/api/authenticate")).permitAll()
75                  .requestMatchers(mvc.pattern("/api/register")).permitAll()
76                  .requestMatchers(mvc.pattern("/api/activate")).permitAll()
77                  .requestMatchers(mvc.pattern("/api/account/reset-password/init")).permitAll()
78                  .requestMatchers(mvc.pattern("/api/account/reset-password/finish")).permitAll()
79                  .requestMatchers(mvc.pattern("/api/admin/**")).hasAuthority(AuthoritiesConstants.ADMIN)
80                  .requestMatchers(mvc.pattern("/api/**")).authenticated()
81                  .requestMatchers(mvc.pattern("/v3/api-docs/**")).hasAuthority(AuthoritiesConstants.ADMIN)
82                  .requestMatchers(mvc.pattern("/management/health")).permitAll()
83                  .requestMatchers(mvc.pattern("/management/health/**")).permitAll()
84                  .requestMatchers(mvc.pattern("/management/info")).permitAll()
85                  .requestMatchers(mvc.pattern("/management/prometheus")).permitAll()
86                  .requestMatchers(mvc.pattern("/management/**")).hasAuthority(AuthoritiesConstants.ADMIN)
87          )
```

Backend Code for Http POST /api/authenticate … check password then create a JWT Token

```
34      @RestController
35      @RequestMapping("/api")
36      public class AuthenticateController {

55          @PostMapping("/authenticate")
56          public ResponseEntity<JWTToken> authorize(@Valid @RequestBody LoginVM loginVM) {
57              UsernamePasswordAuthenticationToken authenticationToken = new UsernamePasswordAuthenticationToken(
58                  loginVM.getUsername(),
59                  loginVM.getPassword()
60              );
61
62              Authentication authentication = authenticationManagerBuilder.getObject().authenticate(authenticationToken
63              SecurityContextHolder.getContext().setAuthentication(authentication);
64              String jwt = this.createToken(authentication, loginVM.isRememberMe());
65              HttpHeaders httpHeaders = new HttpHeaders();
66              httpHeaders.setBearerAuth(jwt);
67              return new ResponseEntity<>(new JWTToken(jwt), httpHeaders, HttpStatus.OK);
68          }
```

Analysis of a JWT Token



JWT Token = **3 parts separated by (2) "."**
part 1,2 are only made of base64 characters (no ".")
part3 is made of ascii

eyJhbGciOiJIUzUxMiJ9
.
eyJzdWIiOiJhZG1pbiIsImV4cCI6MTczMjExOTMwMCwiYXV0aCI6IlJPTEVfQURNSU4gUk9MRV9VU0VSIiwiaWF0IjoxNzMyM
.
G9uNF_VPgAjUsO86TlnRo-ihqvB62Ql_6vSfr6hr_f2BQPhofB7D6XQTLqe6O001lhQCHCvmuIJOjPh8ZbELuQ

JWT Token Part 1: token sign algorithm

decoding part 1 as base 64

```
$ echo eyJhbGciOiJIUzUxMiJ9 | base64 -d
{"alg":"HS512"}
```

JWT Token Part 2: token data

$ **echo**
eyJzdWIiOiJhZG1pbiIsImV4cCI6MTczMjExOTMwMCwiYXV0aCI6IlJPTEVfQURNSU4gUk9MRV9VU0VSIiwiaWF0IjoxNzMyMDMyOTAwfQ== **| base64 -d**
{"sub":"admin","exp":1732119300,"auth":"ROLE_ADMIN ROLE_USER","iat":1732032900}

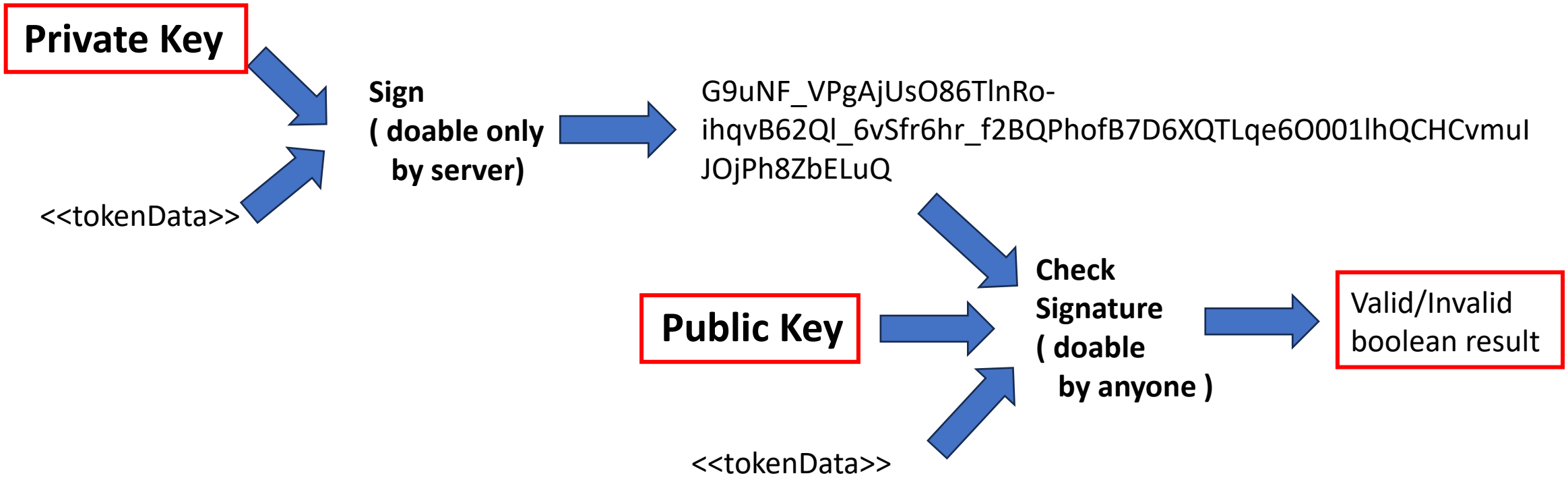
subject
(=username)
bearer of this token

expiry time
in seconds since epoch

authorized Roles

time
in seconds
since epoc

JWT Token Part 3: Cryptographic Signature



Webapp keep JWT Token + Info

on return of Http POST /api/authenticate, WebApp keep informations:

- **username** (=subject)
  currently authenticated

To display "already signed-in" / "anonymous" feedback

- **JWT Token**

To authenticate all following Http Rest API Requests
to backend server

then after Http GET /api/account, WebApp keep other informations:

- **list of roles**

To show/hide features that could not be called by client
(server would reject anyway)

- **detailed Account information**

To translate website in preferred user langage,
to display firstName, lastName, email ...

Login Page Component Html

a classical Form,
with 3 Fields

click to submit
=> call 'login()"

login.component.html

```
1   <div>
2     <div class="d-flex justify-content-center">
3       <div class="col-lg-6 col-md-8 col-sm-10">
9         }
10        <form class="form" (ngSubmit)="login()" [formGroup]="loginForm">
11          <div class="mb-3">
12            <label class="username-label" for="username" jhiTranslate="global.form.username.label">Username</lab
13            <input...>
23          </div>
24
25          <div class="mb-3">
26            <label for="password" jhiTranslate="login.form.password">Password</label>
27            <input...>
36          </div>
37
38          <div class="form-check">
39            <label class="form-check-label" for="rememberMe"...>
43          </div>
44
45          <button type="submit" class="btn btn-primary" data-cy="submit" jhiTranslate="login.form.button">Sign i
46        </form>
```

LoginComponent TypeScript

The code get the 3 values
{username,passowrd,rememberMe}
from the Form,
and call the service

At the end, the Login page is closed,
navigate back to Home page

login() / logout() High Level Facade Code

High-Level method called from component

login.service.ts

```
1   import ...
9
10  @Injectable({ providedIn: 'root' })  Show usages
11  export class LoginService {
12    private readonly accountService = inject(AccountService);
13    private readonly authServerProvider = inject(AuthServerProvider);
14
15    login(credentials: Login): Observable<Account | null> {  Show usages
16      return this.authServerProvider.login(credentials).pipe(mergeMap(() => this.accountService.identity(true)));
17    }
18
19    logout(): void {  Show usages
20      this.authServerProvider.logout().subscribe({ complete: () => this.accountService.authenticate(null) });
21    }
22  }
```

Low Level Http Code + Response

auth-jwt.service.ts

```
15  export class AuthServerProvider {
17    private readonly stateStorageService : StateStorageService  = inject(StateStorageService);
18    private readonly applicationConfigService : ApplicationConfigService  = inject(ApplicationConfigService);
19
20    getToken(): string {  Show usages
21      return this.stateStorageService.getAuthenticationToken() ?? '';
22    }
23
24    login(credentials: Login): Observable<void> {  Show usages
25      return this.http
26        .post<JwtToken>(this.applicationConfigService.getEndpointFor('api/authenticate'), credentials)
27        .pipe(map(response : JwtToken  => this.authenticateSuccess(response, credentials.rememberMe)));
28    }
29
30    logout(): Observable<void> {  Show usages
31      return new Observable(observer : Subscriber<void>  => {
32        this.stateStorageService.clearAuthenticationToken();
33        observer.complete();
34      });
35    }
36
37    private authenticateSuccess(response: JwtToken, rememberMe: boolean): void {  Show usages
38      this.stateStorageService.storeAuthenticationToken(response.id_token, rememberMe);
39    }
```

Low-Level Http POST Request

Callback to Save Response

save Json "id_token" to Storage

Storing JWT Token

if selected "rememberMe",

then save to LocalStorage

else to temporary SessionStorage

state-storage.service.ts

```
4   export class StateStorageService {
21
22    storeAuthenticationToken(authenticationToken: string, rememberMe: boolean): void {
23      authenticationToken = JSON.stringify(authenticationToken);
24      this.clearAuthenticationToken();
25      if (rememberMe) {
26        localStorage.setItem(this.authenticationKey, authenticationToken);
27      } else {
28        sessionStorage.setItem(this.authenticationKey, authenticationToken);
29      }
30    }
```

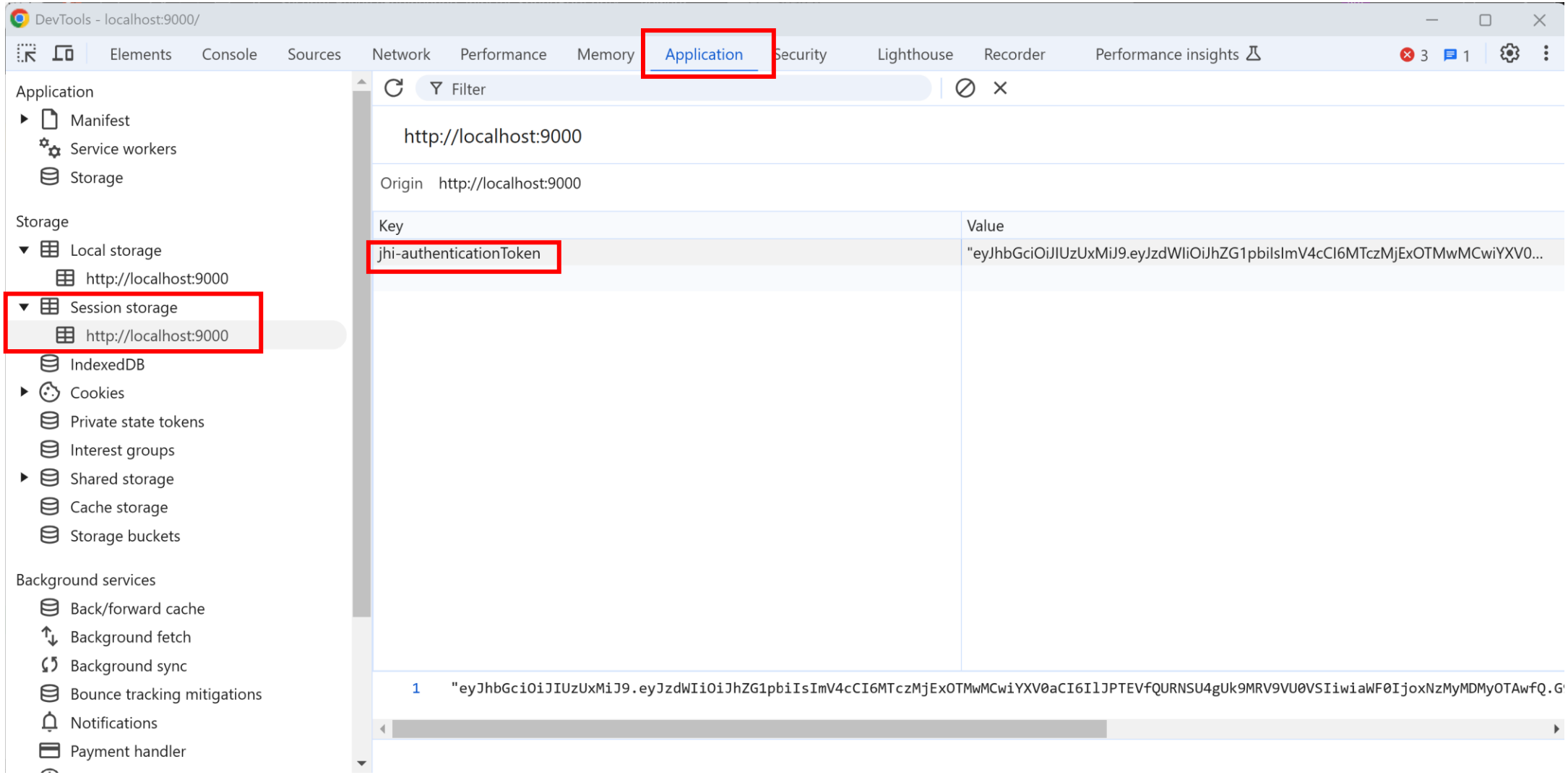
Stored JWT Available for later Http Calls

state-storage.service.ts

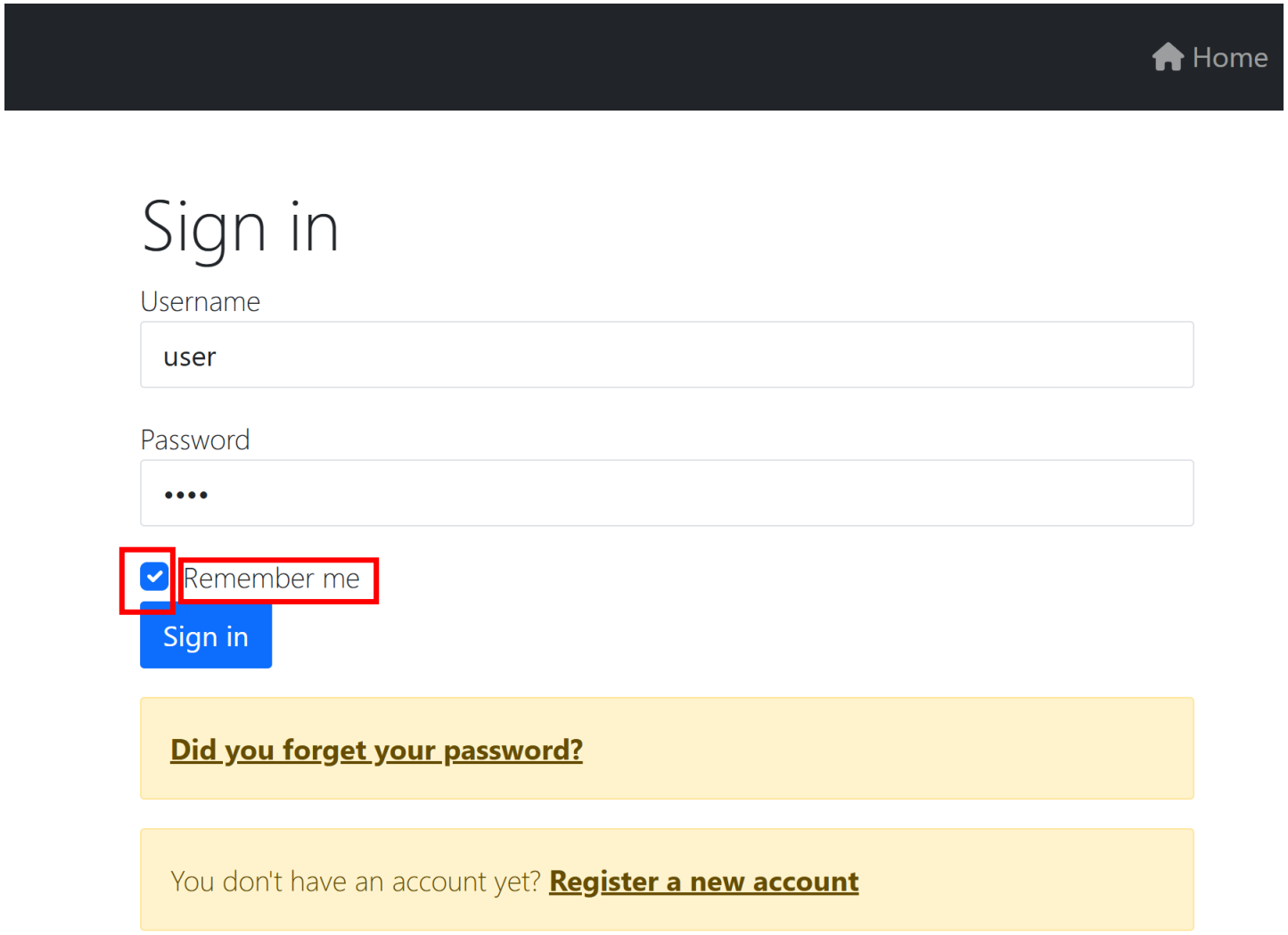
```
4   export class StateStorageService {
31
32    getAuthenticationToken(): string | null {  Show usages
33      const authenticationToken : string | null  = localStorage.getItem(this.authenticationKey) ?? sessionStorage.getItem(this.authenticationKey);
34      return authenticationToken ? (JSON.parse(authenticationToken) as string | null) : authenticationToken;
35    }
```

The code retries both local and session Storage,
then reparse from JSON to get JWT Bearer Token value

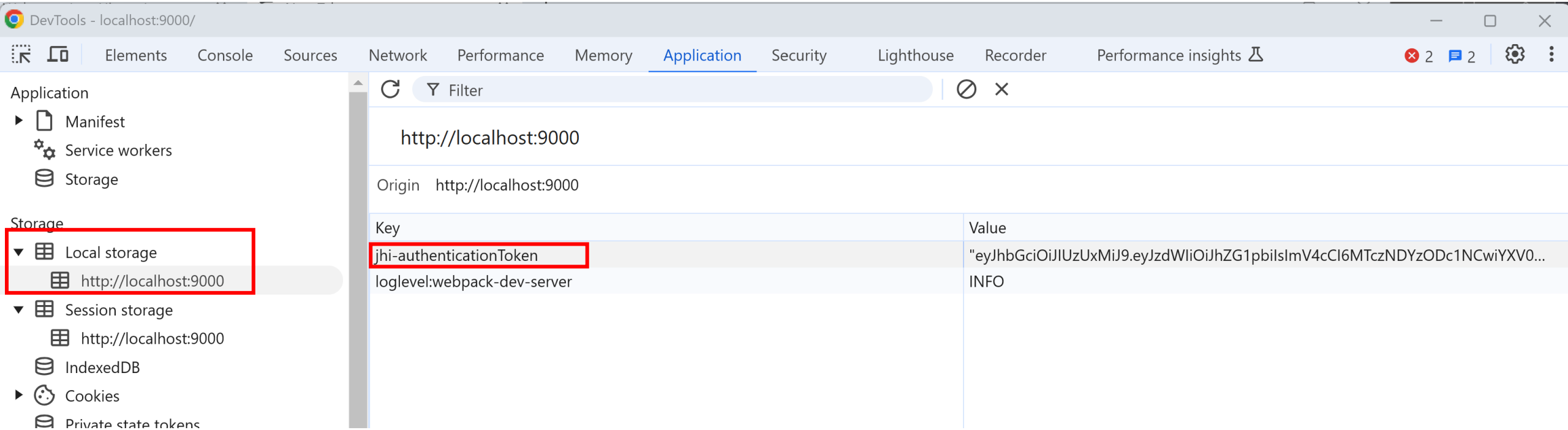
Viewing SessionStorage with Chrome Devtools



Sign-in with "rememberMe" option

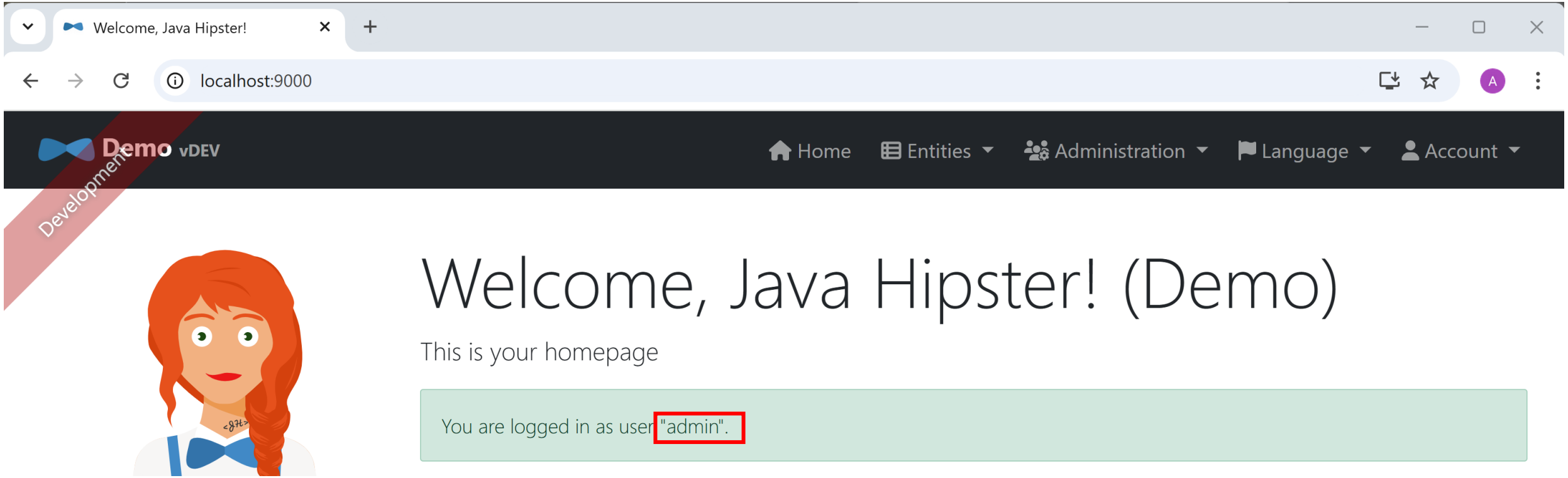


rememberMe => JWT in LocalStorage



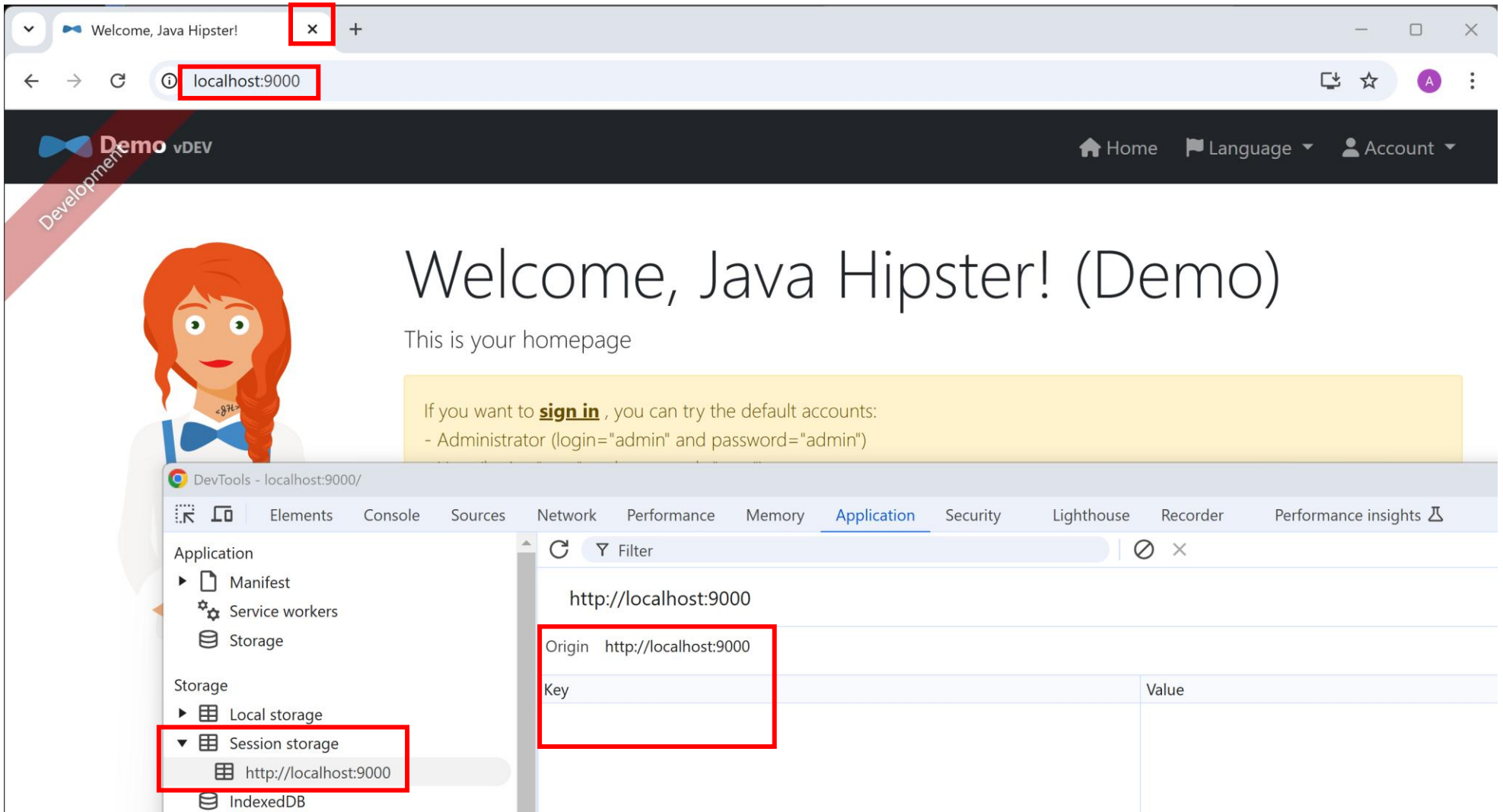
LocalStorage :
resist to a Close Browser Tab + Re-Open
(!= SessionStorage, purged)

Closed + re-Opened Tab => still "logged-in" (even resist to Quit Browser application, relaunch it, reopen Tab)

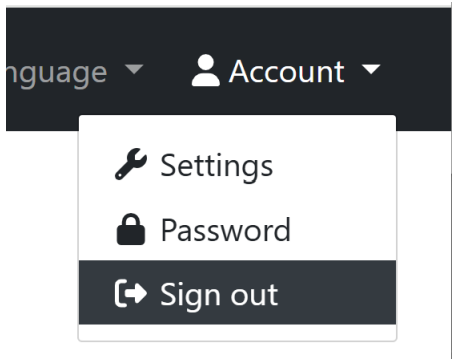


SessionStorage => auto purge on Close Tab

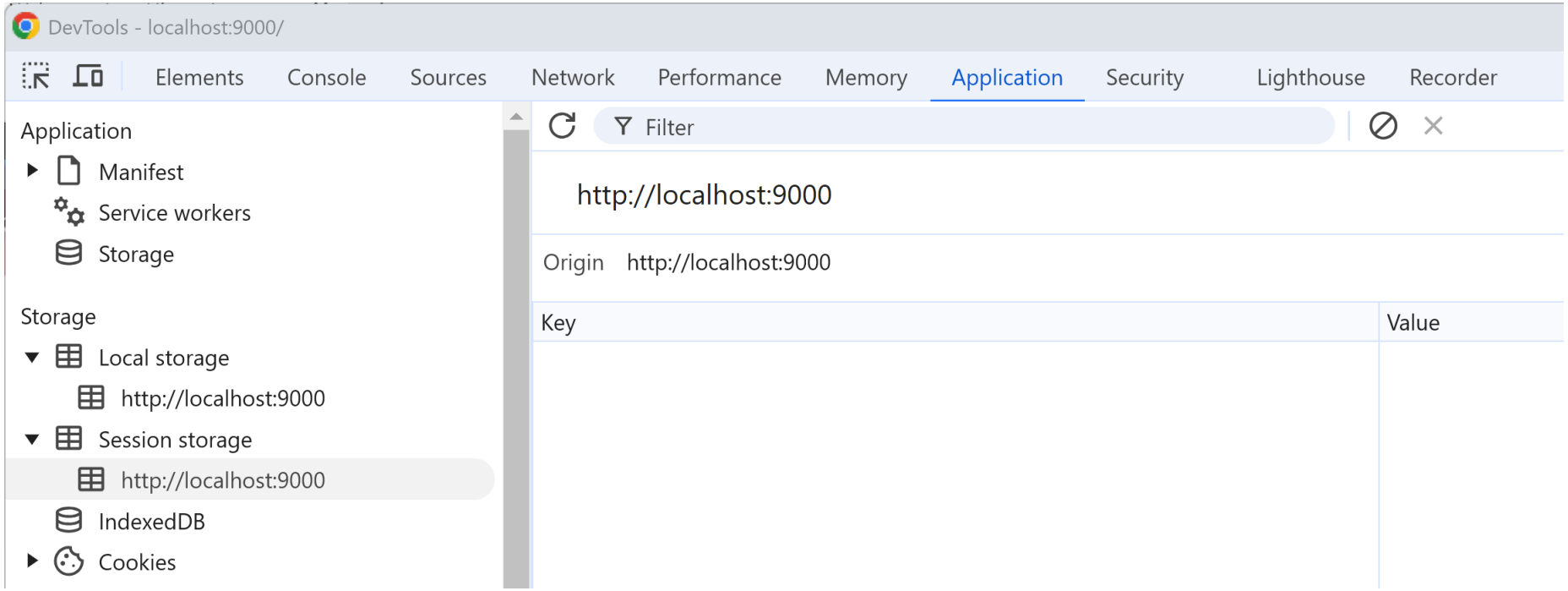
Close Tab => even Chrome Dev Tools panel is Closed
Re-Open new tab with same URL => then re-open Chrome Dev Tools => SessionStorage was cleared



On Sign-Out
=> All Security information must be cleared!
(Local + Session)

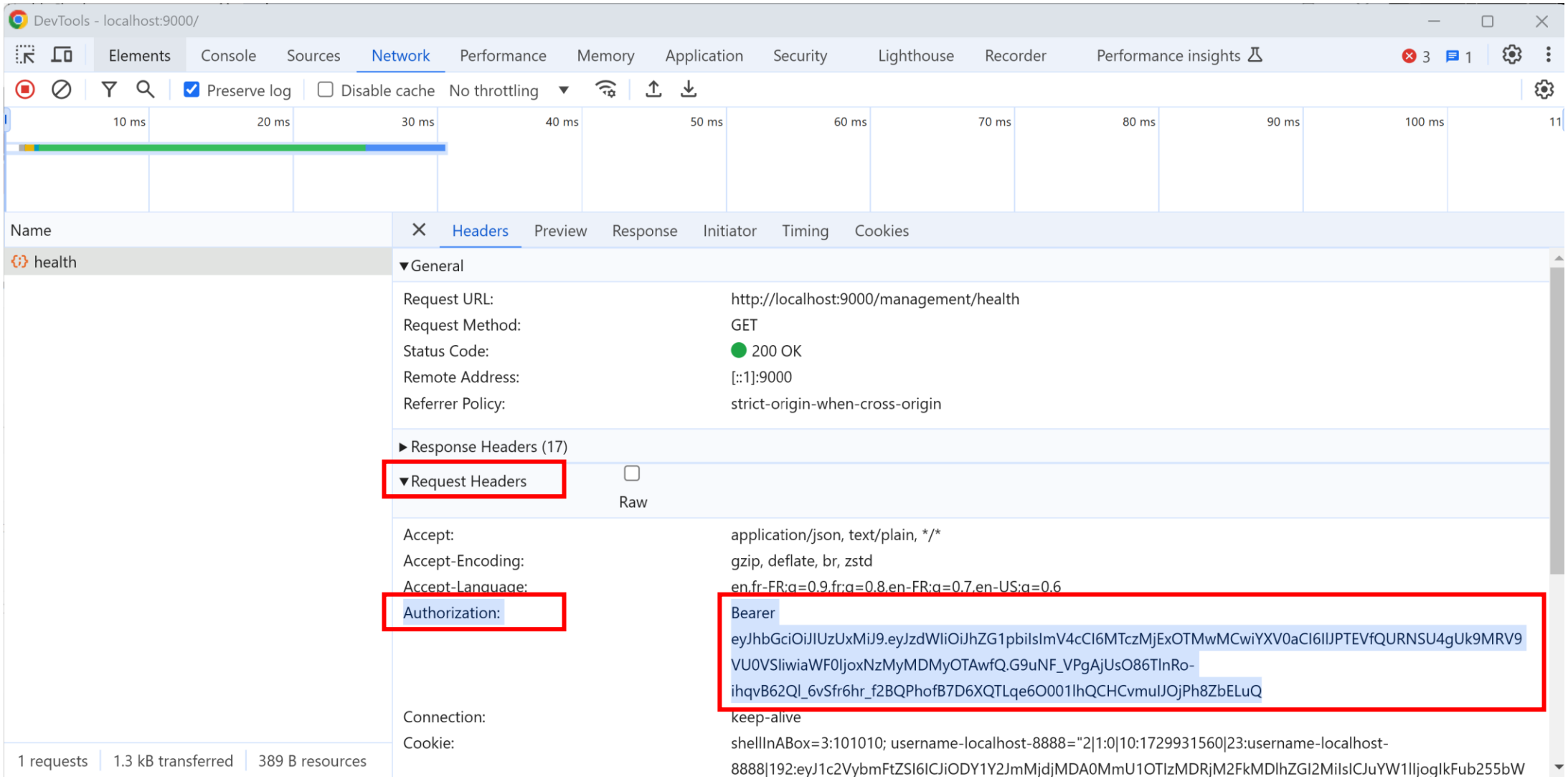



```
37    clearAuthenticationToken(): void {  Show usages
38      sessionStorage.removeItem(this.authenticationKey);
39      localStorage.removeItem(this.authenticationKey);
40    }
```



Passing JWT back in Http Rest API Calls

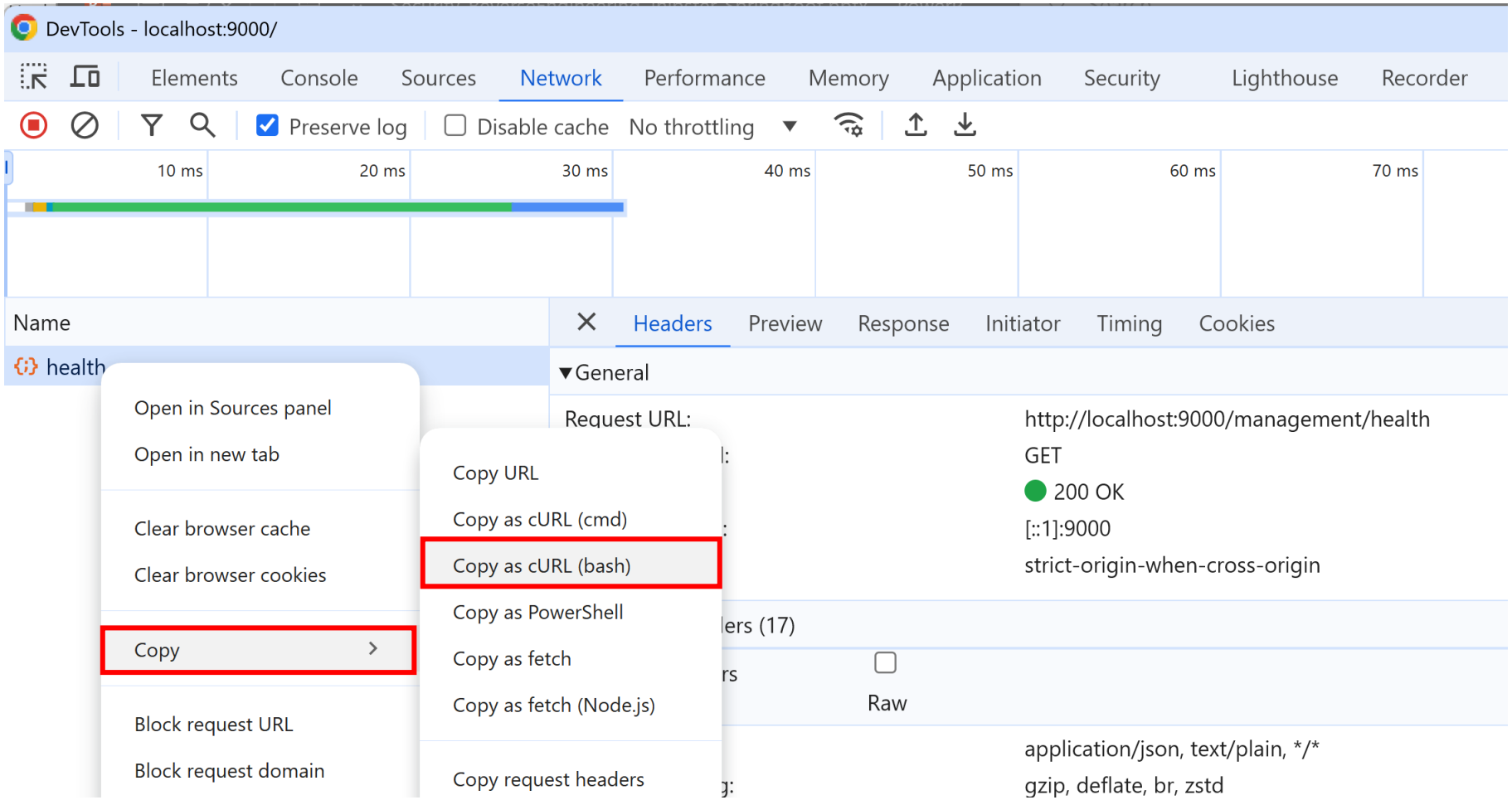
Debugging Authenticated Rest API Calls



Header "Authorization: Bearer ${token}" ...

All Rest API calls are Authenticated using Http Header "Authorization", containing a Bearer token

Minimalist Check using Curl



Full Curl Command ... can simplify

```
curl 'http://localhost:9000/management/health' \
  -H 'Accept: application/json, text/plain, */*' \
  -H 'Accept-Language: en,fr-FR;q=0.9,fr;q=0.8,en-FR;q=0.7,en-US;q=0.6' \
  -H 'Authorization: Bearer eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJhZG1pbiIsImV4cCI6MTczMjExOTMwMCwiYXV0aCI6IlJPTEVfQURNS
  -H 'Connection: keep-alive' \
  -H 'Cookie: shellInABox=3:101010; username-localhost-8888="2|1:0|10:1729931560|23:username-localhost-8888|192:eyJ1c2
  -H 'Referer: http://localhost:9000/admin/health' \
  -H 'Sec-Fetch-Dest: empty' \
  -H 'Sec-Fetch-Mode: cors' \
  -H 'Sec-Fetch-Site: same-origin' \
  -H 'User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/131.0.0.0 Safari
  -H 'sec-ch-ua: "Google Chrome";v="131", "Chromium";v="131", "Not_A Brand";v="24"' \
  -H 'sec-ch-ua-mobile: ?0' \
  -H 'sec-ch-ua-platform: "Windows"'
```

Check Minimalist Curl Command

token="..."
curl 'http://localhost:9000/management/health' **-H "Authorization: Bearer ${token}"**

```
arnaud@DesktopArnaud /cygdrive/c/arn/devPerso
$ token="eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJhZG1pbiIsImV4cCI6MTczMjExOTMwMCwiYXV0aCI6IlJPTEVfQURNSU
TlnRo-ihqvB62Ql_6vSfr6hr_f2BQPhofB7D6XQTLqe6O001lhQCHCvmuIJOjPh8ZbELuQ"

arnaud@DesktopArnaud /cygdrive/c/arn/devPerso
$ curl 'http://localhost:9000/management/health' -H "Authorization: Bearer ${token}"
{"status":"UP","components":{"db":{"status":"UP","details":{"database":"H2","validationQuery":"i
1022601719808,"free":168535191552,"threshold":10485760,"path":"C:\\web\\demo-jhipster\\demo\\.",
tatus":"UP"},"readinessState":{"status":"UP"}},"groups":["liveness","readiness"]}
```

HttpClient Code in Angular :
NO extra Code for Header !

health.service.ts

```
1   import ...
7
8   @Injectable({ providedIn: 'root' })  Show usages
9   export class HealthService {
10    private readonly http : HttpClient  = inject(HttpClient);
11    private readonly applicationConfigService : ApplicationConfigService  = inject(ApplicationConfigService);
12
13    checkHealth(): Observable<Health> {  Show usages
14      return this.http.get<Health>(this.applicationConfigService.getEndpointFor('management/health'));
15    }
```

HttpClient ... using Http Interceptors

app.config.ts

```
45
46  export const appConfig: ApplicationConfig = {  Show usages
47    providers: [
48      provideRouter(routes, ...routerFeatures),
49      importProvidersFrom(BrowserModule),
50      // Set this to true to enable service worker (PWA)
51      importProvidersFrom(ServiceWorkerModule.register('ngsw-worker.js', { enabled: false })),
52      importProvidersFrom(TranslationModule),
53      provideHttpClient(withInterceptorsFromDi()),
54      Title,
55      { provide: LOCALE_ID, useValue: 'en' },
56      { provide: NgbDateAdapter, useClass: NgbDateDayjsAdapter },
57      httpInterceptorProviders,
58      { provide: TitleStrategy, useClass: AppPageTitleStrategy },
59      // jhipster-needle-angular-add-module JHipster will add new module here
60    ],
61  };
```

List of Dependency Injected HttpInterceptors

Interceptor for adding Http Header

other Interceptors, for renewing, showing errors, showing notifications..

index.ts

```ts
import { HTTP_INTERCEPTORS } from '@angular/common/http';

import { AuthInterceptor } from 'app/core/interceptor/auth.interceptor';
import { AuthExpiredInterceptor } from 'app/core/interceptor/auth-expired.interceptor';
import { ErrorHandlerInterceptor } from 'app/core/interceptor/error-handler.interceptor';
import { NotificationInterceptor } from 'app/core/interceptor/notification.interceptor';

export const httpInterceptorProviders : ({ provide: InjectionToken<readonly HttpI... = [  Show usages
  {
    provide: HTTP_INTERCEPTORS,
    useClass: AuthInterceptor,
    multi: true,
  },
  {
    provide: HTTP_INTERCEPTORS,
    useClass: AuthExpiredInterceptor,
    multi: true,
  },
  {
    provide: HTTP_INTERCEPTORS,
    useClass: ErrorHandlerInterceptor,
    multi: true,
  },
  {
    provide: HTTP_INTERCEPTORS,
    useClass: NotificationInterceptor,
    multi: true,
  },
];
```

AuthInterceptor

do not add token
for http urls
other than backend
Api server

add Bearer
JWT Token
for backend calls

auth.interceptor.ts

```
1   import ...
7
8   @Injectable()  Show usages
9   export class AuthInterceptor implements HttpInterceptor {
10    private readonly stateStorageService : StateStorageService  = inject(StateStorageService);
11    private readonly applicationConfigService : ApplicationConfigService  = inject(ApplicationConfigService);
12
13    intercept(request: HttpRequest<any>, next: HttpHandler): Observable<HttpEvent<any>> {  no usages
14      const serverApiUrl : string  = this.applicationConfigService.getEndpointFor('');
15      if (!request.url || (request.url.startsWith('http') && !(serverApiUrl && request.url.startsWith(serverApiU
16        return next.handle(request);
17      }
18
19      const token: string | null = this.stateStorageService.getAuthenticationToken();
20      if (token) {
21        request = request.clone({
22          setHeaders: {
23            Authorization: `Bearer ${token}`,
24          },
25        });
26      }
27      return next.handle(request);
28    }
```

Displaying Angular Features Conditionaly on Authenticated / Authorized

@if (account()) { .. } @else { .. }

home.component.html

```
1    <div class="row">
6      <div class="col-md-9">
11       <div>
12         @if (account() !== null) {
13           <div class="alert alert-success">
14             @if (account(); as accountRef) {
15               <span id="home-logged-message" jhiTranslate="home.logged.message" [translateVa
16                 >You are logged in as user "{{ accountRef.login }}".</span
17               >
18             }
19           </div>
20         } @else {
21           <div class="alert alert-warning">
22             <span jhiTranslate="global.messages.info.authenticated.prefix">If you want to </
23             <a class="alert-link" (click)="login()" jhiTranslate="global.messages.info.authe
24               sign in</a>
25             <span jhiTranslate="global.messages.info.authenticated.suffix"
26               >, you can try the default accounts:<br />- Administrator (login="admin" and p
27               (login="user" and password="user").</span
28             >
29           </div>
```

account() signal computed from accountService.getAuthenticationState()

home.component.ts

```
14    styleUrl: './home.component.scss',
15    imports: [SharedModule, RouterModule],
16  })
17  export default class HomeComponent implements OnInit, OnDestroy {
18    account : WritableSignal<Account | null> = signal<Account | null>(null);
19
20    private readonly destroy$ : Subject<void> = new Subject<void>();
21
22    private readonly accountService : AccountService = inject(AccountService);
23    private readonly router : Router = inject(Router);
24
25    ngOnInit(): void {  Show usages
26      this.accountService
27        .getAuthenticationState()
28        .pipe(takeUntil(this.destroy$))
29        .subscribe(account : Account | null => this.account.set(account));
30    }
```

AccountService.authenticationState
... computed from http GET /api/account reponse

see next
where authenticate()
method is called

only location where
authenticationState is "set"

account.service.ts

```
1   import ...
11
12  @Injectable({ providedIn: 'root' })  Show usages
13  export class AccountService {
14    private readonly userIdentity = signal<Account | null>(null);
15    private readonly authenticationState = new ReplaySubject<Account | null>(1);
16    private accountCache$?: Observable<Account> | null;

28    authenticate(identity: Account | null): void {  Show usages
29      this.userIdentity.set(identity);
30      this.authenticationState.next(this.userIdentity());
31      if (!identity) {
32        this.accountCache$ = null;
33      }
34    }
```

Caching Detailed Account Info of current User (Bearer of JWT)

... this code is caching
the fetch reponse
of "http GET /api/account"

= get detailed firstname,
lastname, email, etc...
for the current authenticated
(JWT) user

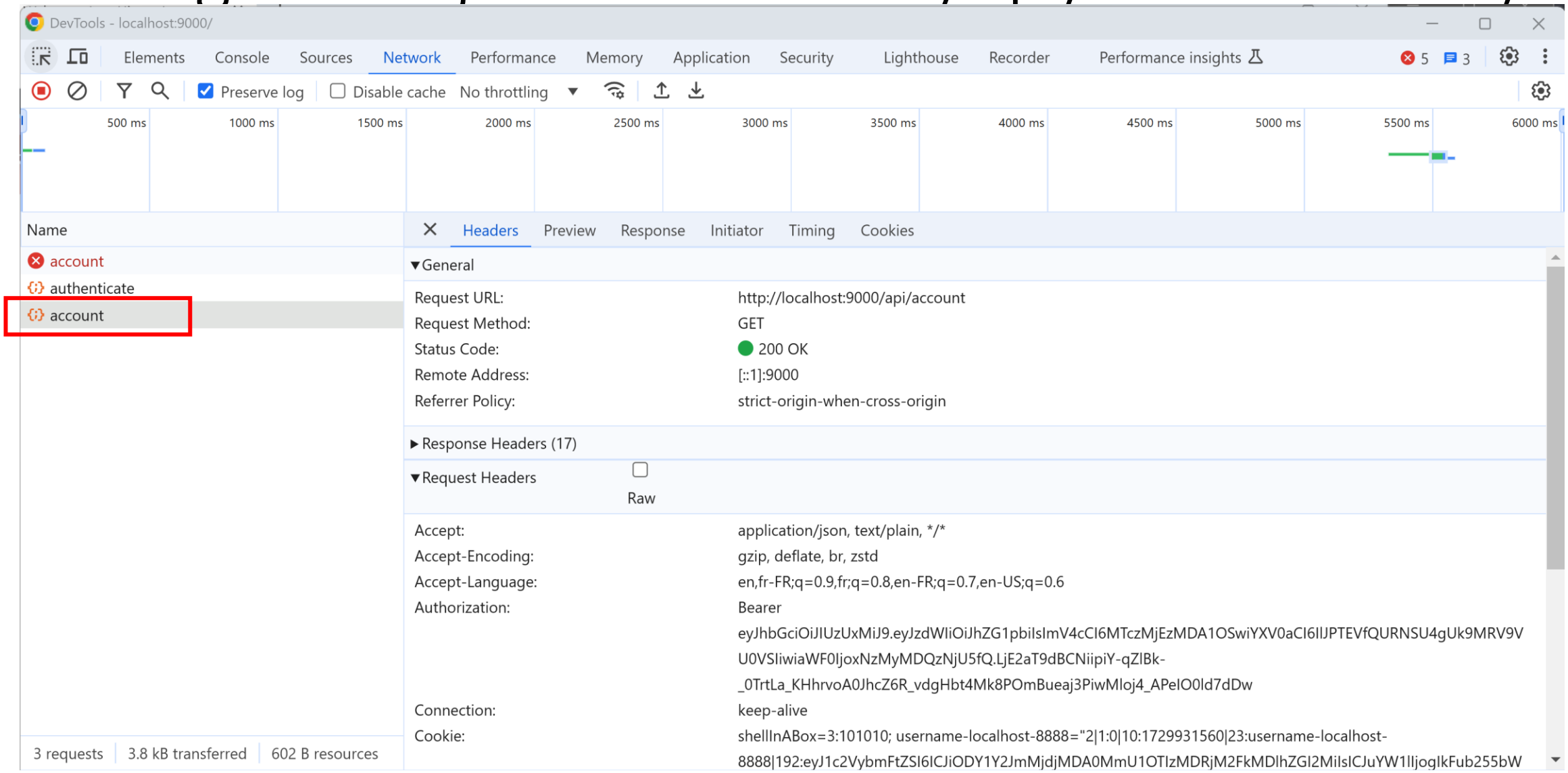
```
51    identity(force?: boolean): Observable<Account | null> {
52      if (!this.accountCache$ || force) {
53        this.accountCache$ = this.fetch().pipe(
54          tap((account: Account) : void  => {
55            this.authenticate(account);
```

... code ommited

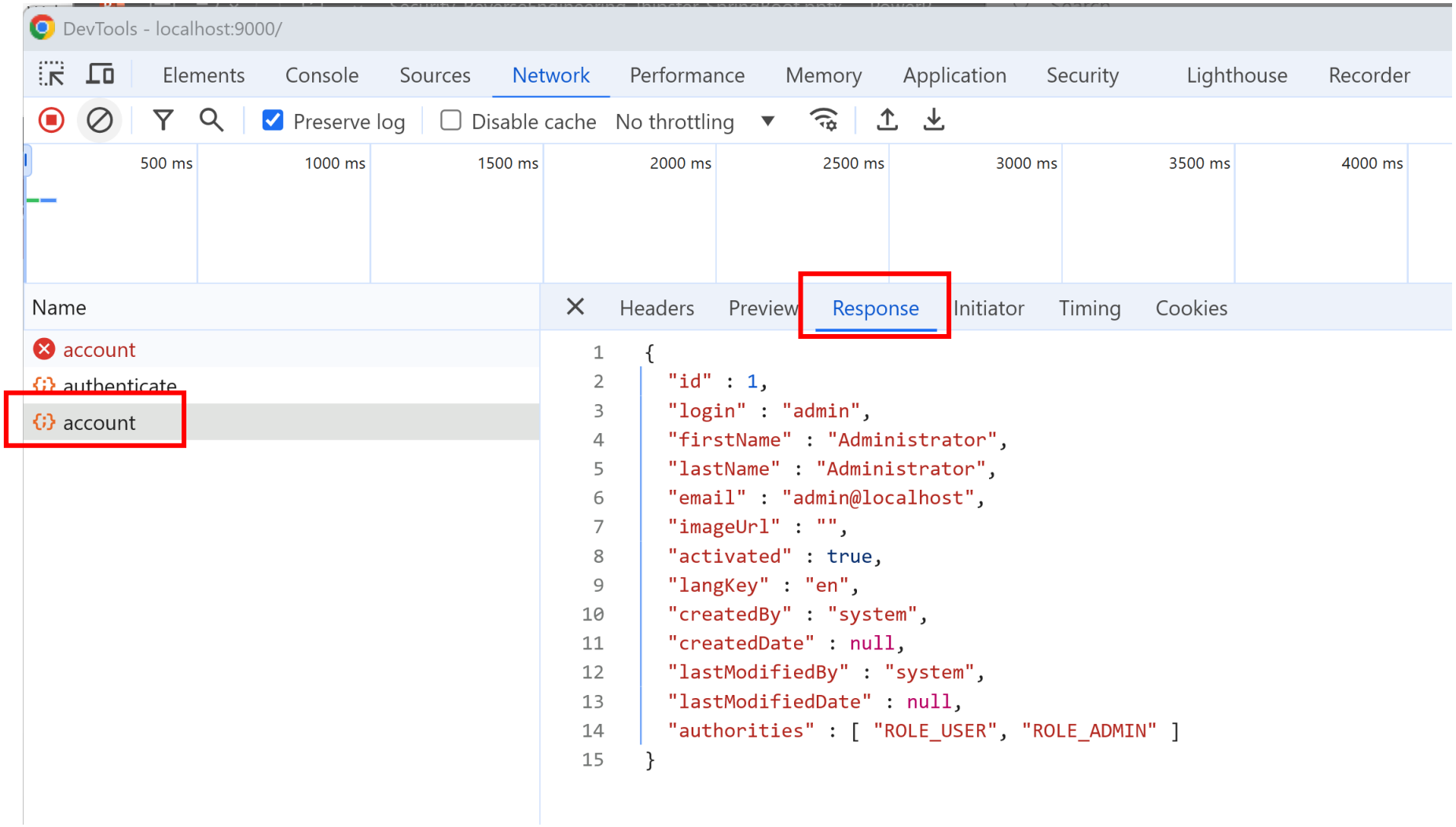
```
65          }),
66          shareReplay(),
67        );
68      }
69      return this.accountCache$.pipe(catchError(() : Observable<null>  => of(null)));
70    }
```

```
80    private fetch(): Observable<Account> {  Show usages
81      return this.http.get<Account>(this.applicationConfigService.getEndpointFor('api/account'));
82    }
```

http GET /api/account
(caching called just after POST /api/authenticate)



Response of http GET /api/account : detailed Information



Notice "authorities" :
[ROLE_ADMIN, ROLE_USER, ..]

```
13    "lastModifiedDate" : null,
14    "authorities" : [ "ROLE_USER", "ROLE_ADMIN" ]
15  }
```

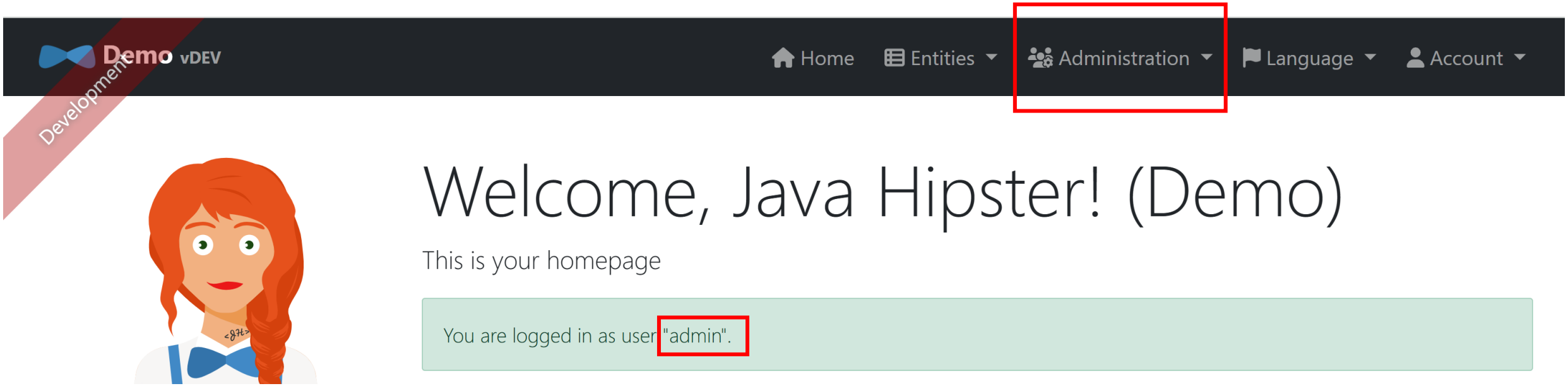
"**authorities**" = synonym of "**Roles**" or "**Authorizations**"
= listed of "**granted things**"

redundant with info already in JWT Token,
but the "Authorization" code works independently of the "Authentication".
Could plug authentication with OAuth2, PIN card, Face recognition, etc,
=> the code for "authorities" is separated

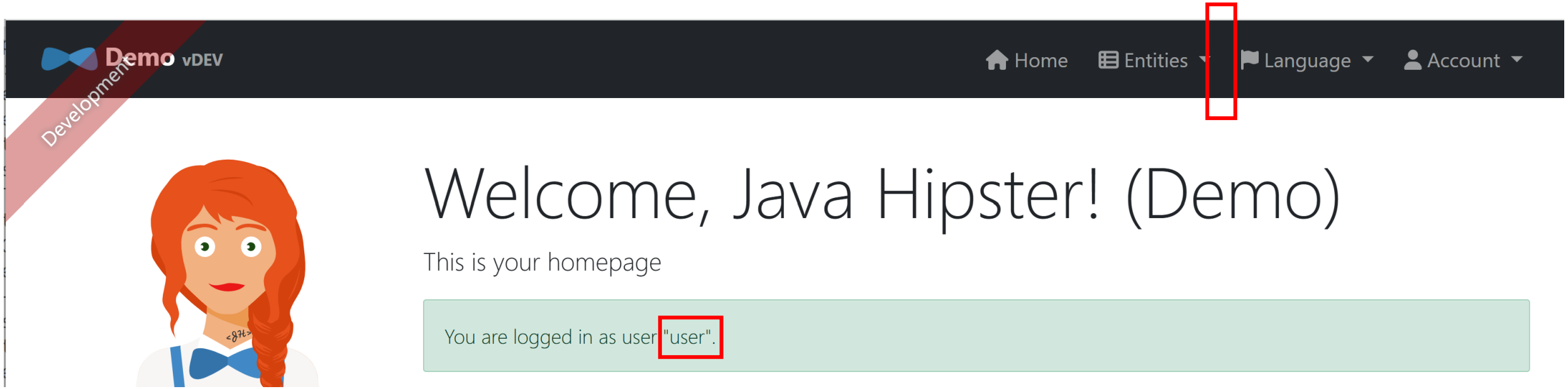
hasAnyAuthority :
check if authorized for a Role

```
40    hasAnyAuthority(authorities: string[] | string): boolean {  Show usages
41      const userIdentity : Account | null  = this.userIdentity();
42      if (!userIdentity) {
43        return false;
44      }
45      if (!Array.isArray(authorities)) {
46        authorities = [authorities];
47      }
48      return userIdentity.authorities.some((authority: string) : boolean  => authorities.includes(authority));
49    }
```

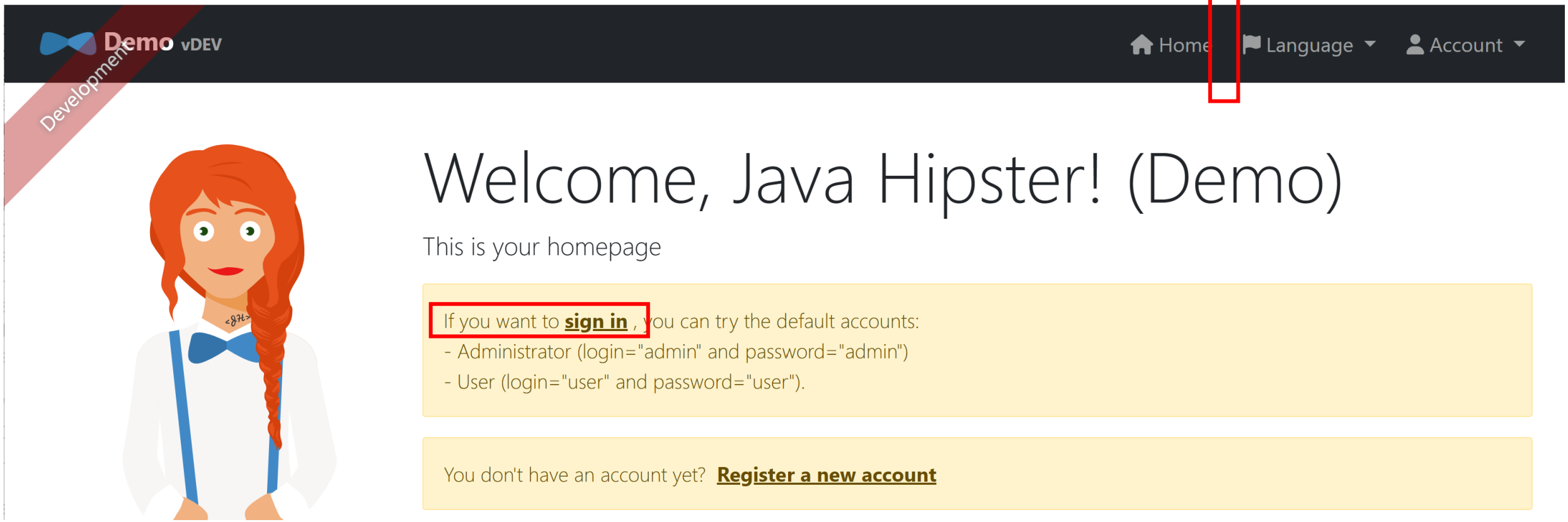
Showing Menu "Administration" when having ROLE_ADMIN



Hiding Menu "Administration" when logged-in, but NOT as an admin



Hiding Menu "Administration" when not logged-in



Showing/Hiding Features based on Role

The menu "Administration" is visible only if user has ROLE_ADMIN

navbar.component.html

```
1    <nav data-cy="navbar" class="navbar navbar-dark navbar-expand-md bg-dark">
2      <div class="container-fluid">
20       <div class="navbar-collapse collapse" id="navbarResponsive" [ngbCollapse]="isNa
21         <ul class="navbar-nav ms-auto">
32           <li
144          </li>
145        }
146        <li
147          *jhiHasAnyAuthority="'ROLE_ADMIN'"
148          ngbDropdown
149          class="nav-item dropdown pointer"
150          display="dynamic"
151          routerLinkActive="active"
152          [routerLinkActiveOptions]="{ exact: true }"
153        >
154          <a class="nav-link dropdown-toggle" ngbDropdownToggle href="javascript:vo
155            <span>
156              <fa-icon icon="users-cog"></fa-icon>
157              <span jhiTranslate="global.menu.admin.main">Administration</span>
158            </span>
159          </a>
160          <ul class="dropdown-menu" ngbDropdownMenu aria-labelledby="admin-menu">
161            <li>
162              <a
163                class="dropdown-item"
164                routerLink="/authority"
```

Code for Directive *jhiHasAnyAuthority=".."

compute if current user account has role

if has role, then
display the element
else
hide the element

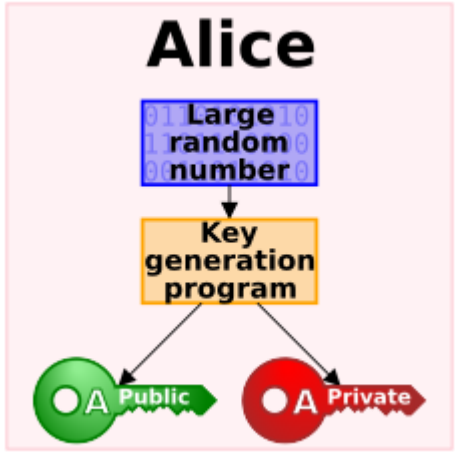
has-any-authority.directive.ts

```
16  @Directive({  Show usages
17    standalone: true,
18    selector: '[jhiHasAnyAuthority]',
19  })
20  export default class HasAnyAuthorityDirective {
21    public authorities : InputSignal<string | string[]>  = input<string | string[]>([], { alias: 'jhiHasAnyAuthority'
22
23    private readonly templateRef : TemplateRef<any>  = inject(TemplateRef<any>);
24    private readonly viewContainerRef : ViewContainerRef  = inject(ViewContainerRef);
25
26    constructor() {  no usages
27      const accountService : AccountService  = inject(AccountService);
28      const currentAccount : Signal<Account | null>  = accountService.trackCurrentAccount();
29      const hasPermission : Signal<boolean | undefined>  = computed(
30          () : boolean | undefined  => currentAccount()?.authorities &&
31                                       accountService.hasAnyAuthority(this.authorities()));
32
33      effect(
34        () : void  => {
35          if (hasPermission()) {
36            this.viewContainerRef.createEmbeddedView(this.templateRef);
37          } else {
38            this.viewContainerRef.clear();
39          }
```

Network & Infrastructure Deployments Options

Reminder: Public/Private Asymetric Cryptography

https://en.wikipedia.org/wiki/Public-key_cryptography



- Only the server can sign with the **Private Key**

- Everybody can check the signature with the **Public Key**

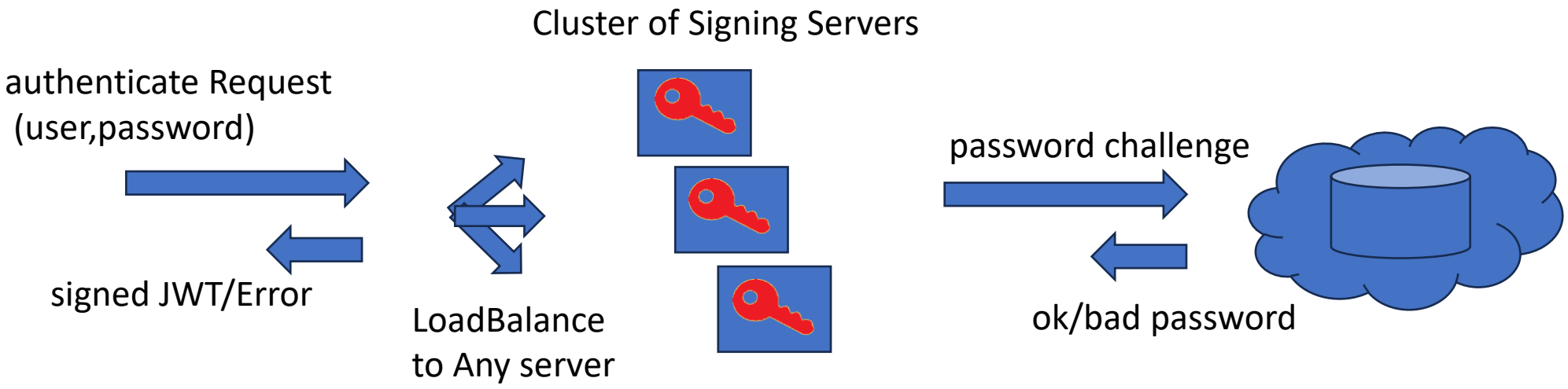
Private Key should be stored carefully on servers and protected.

Public Key can be broadcasted.

Signing a JWT Token
= need the Private Cryptographic Key (+ cpu)

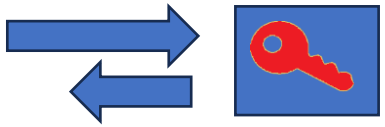
The private Key can be deployed as a secured configuration file
OR distributed in a secured cluster over Https
(chicken and egg pb: need another secret to ensure trust)

For JWT, the server is "STATELESS" : need not saving anything in Database
For checking Passwords, servers still need to connect to a LDAP realm, or Database



Checking JWT
= need only Public Cryptographic Key (+ cpu)

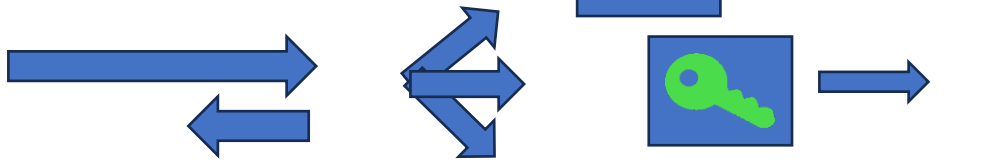
Servers for Signing
!= WebApp



The WebApp servers are "STATELESS"
The public Key can be distributed / broadcasted easily

Cluster of WebApp Servers (not signing)

Both roles: WebApp
+ capable of Signing

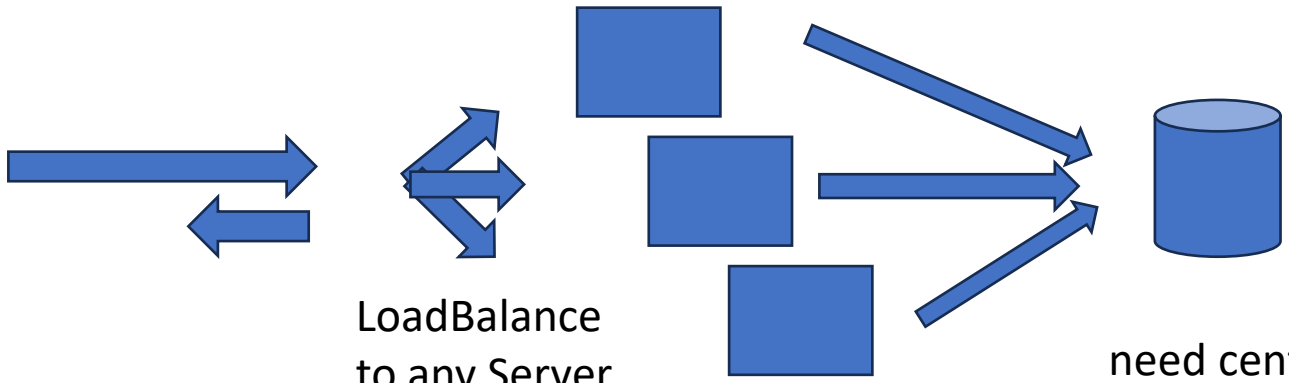


LoadBalance
to any Server

JWT .. easy to deploy Stateless Servers

See Next for "HTTP Session" using COOKIE => much more complex

need either "Database" to store all cookies
OR LoadBalancer supporting "Sessions Stickiness"



Conclusion

JHipster is ULTRA HIGH quality generated code

No compromise on Security

(pretty difficult to read, but interresting to learn new things done right)

There's more to it
- CRUD screen generated for your model
- can plug other frontend langages (React, VueJs)
- can plug different security mode: OAuth2 or old-style "COOKIE jsessionId"

Next Steps

- **Debugging the OAuth2 security mode**

  the "login" page is not hosted in your webapp,
  it is redirected to another trusted webapp,
  for example "FranceConnect", "Google", "github", etc
  then redirecting back to your website with a token

- **Debugging old-style COOKIE "jsessionId" security mode**

Questions

arnaud.nauwynck@gmail.com