

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](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

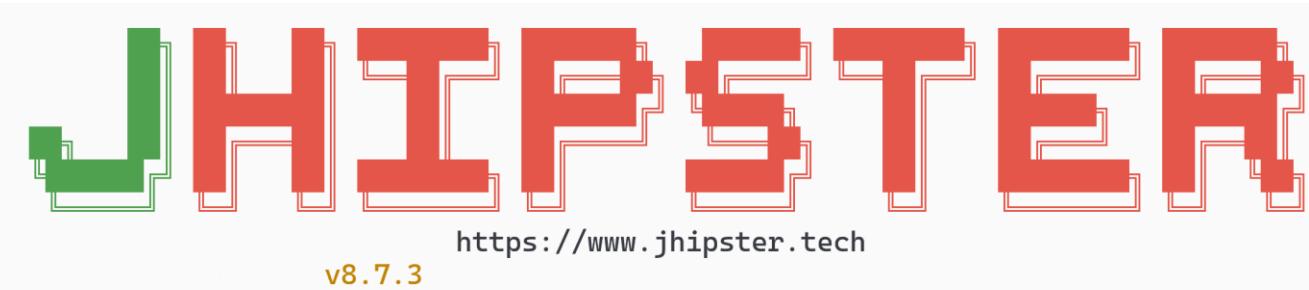
Step 1: re-generate ANOTHER Jhipster project with same name

```
# 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



```
(node:17324) [DEP0040] DeprecationWarning: The 'punycode' module is deprecated. Please use a userland alternative instead.  
(Use 'node --trace-deprecation ...' to show where the warning was created)
```

<https://www.jhipster.tech/creating-an-app/>

c:\web\demo-jhipster\tmp\demo-sessionAuth

WARNING! Your Node version is not LTS (Long Term Support), use it at your own risk! JHipster does not support non-LTS releases, so if you encounter a bug, please use a LTS version first.

? What is the base name of your application? (demoSessionAuth)

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

? What is the base name of your application? demo

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

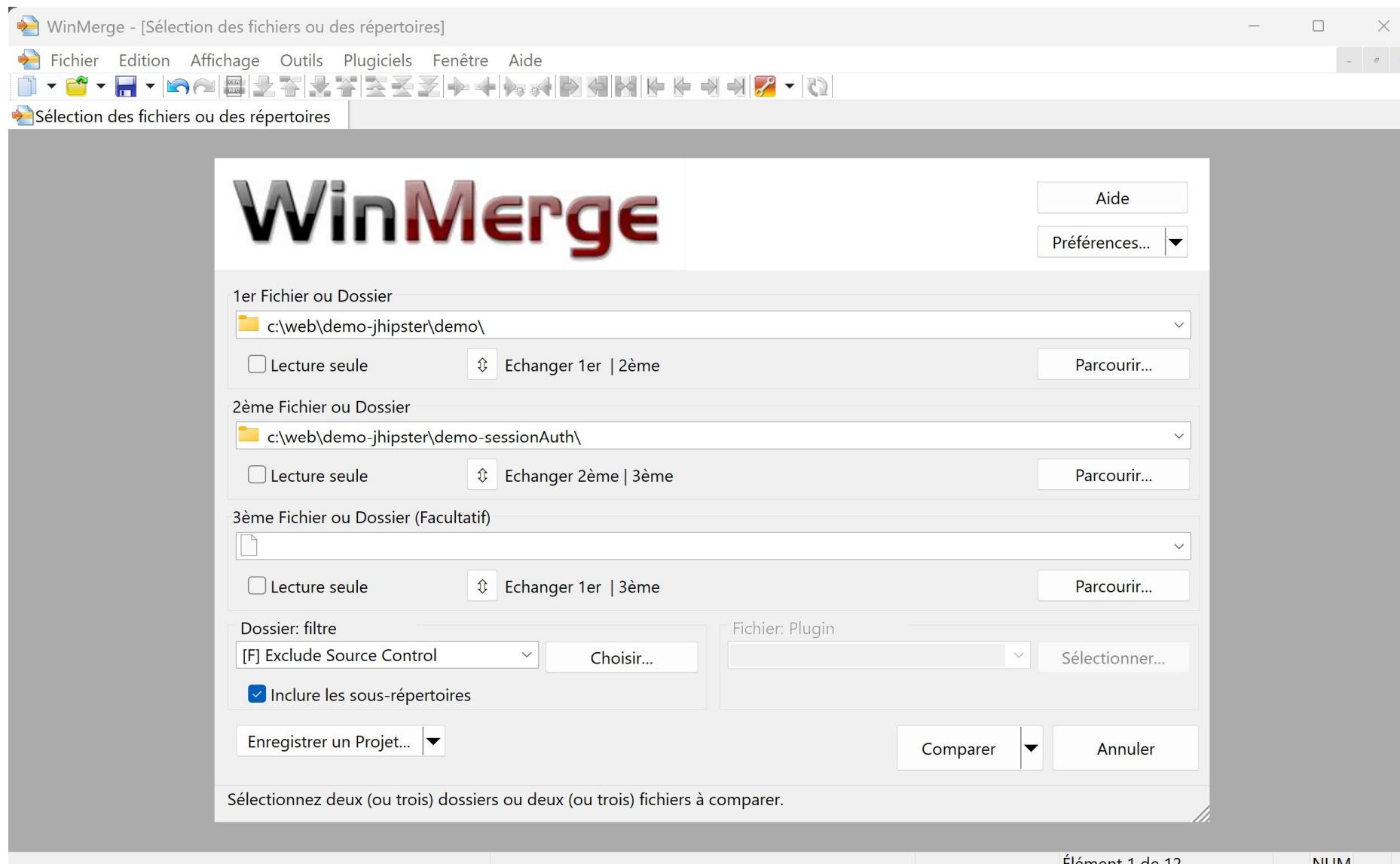
- ✓ What is the base name of your application? `demoSessionAuth`
- ✓ Which *type* of application would you like to create? `Monolithic application (recommended for simple projects)`
- ✓ What is your default Java package name? `com.mycompany.myapp`
- ✓ Would you like to use Maven or Gradle for building 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\	Répertoire	Résultat de la comparaison
Nom du fichier		
> .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	Les répertoires sont différents
`- repository	src\main\java\com\mycompany\myapp	Juste à Gauche : c:\web\demo-jhipster\demo
`- 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-sessionAuth\
Nom du fichier	Répertoire	Résultat de la comparaison
src	src	Les répertoires sont différents
main	src\main	Les répertoires sont différents
java	src\main\java	Les répertoires sont différents
com	src\main\java\com	Les répertoires sont différents
mycompany	src\main\java\com\mycompany	Les répertoires sont différents
myapp	src\main\java\com\mycompany\myapp	Les répertoires sont différents
config	src\main\java\com\mycompany\myapp\config	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-sessionAuth\
Nom du fichier	Répertoire	Résultat de la comparaison
> .husky		Les répertoires sont différents
src	src	Les répertoires sont différents
main	src\main	Les répertoires sont différents
java	src\main\java	Les répertoires sont différents
com	src\main\java\com	Les répertoires sont différents
mycompany	src\main\java\com\mycompany	Les répertoires sont différents
myapp	src\main\java\com\mycompany\myapp	Les répertoires sont différents
domain	src\main\java\com\mycompany\myapp\domain	Les répertoires sont différents
PersistentToken.java	src\main\java\com\mycompany\myapp\domain\PersistentToken.java	Juste à Droite : c:\web\demo-jhipster\de
repository	src\main\java\com\mycompany\myapp\repository	Les répertoires sont différents
PersistentTokenRepository.java	src\main\java\com\mycompany\myapp\repository\PersistentTokenRepository.java	Juste à Droite : c:\web\demo-jhipster\de
security	src\main\java\com\mycompany\myapp\security	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\sessions	Juste à Droite : c:\web\demo-jhipster\de
session.model.ts	src\main\webapp\app\account\sessions\session.model.ts	Juste à Droite : c:\web\demo-jhipster\de
sessions.component.html	src\main\webapp\app\account\sessions\sessions.component.html	Juste à Droite : c:\web\demo-jhipster\de
sessions.component.spec.ts	src\main\webapp\app\account\sessions\sessions.component.spec.ts	Juste à Droite : c:\web\demo-jhipster\de
sessions.component.ts	src\main\webapp\app\account\sessions\sessions.component.ts	Juste à Droite : c:\web\demo-jhipster\de
sessions.route.ts	src\main\webapp\app\account\sessions\sessions.route.ts	Juste à Droite : c:\web\demo-jhipster\de
sessions.service.ts	src\main\webapp\app\account\sessions\sessions.service.ts	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\auth-session.service.ts	Juste à Droite : c:\web\demo-jhipster\de

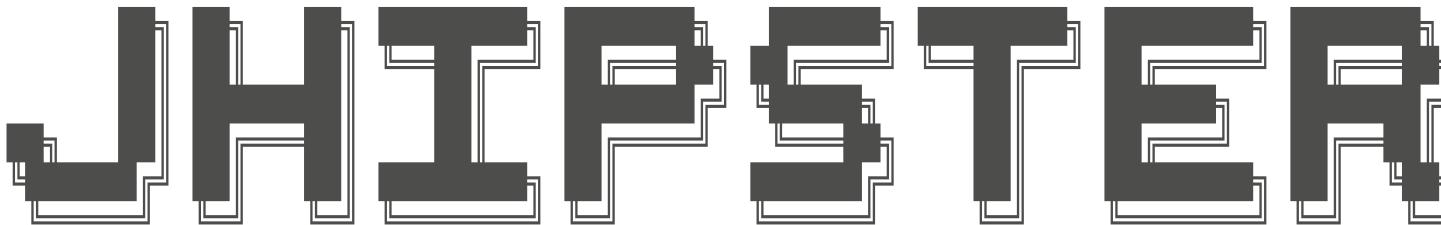
Different Files (maybe not for auth mode)

Nom du fichier	Répertoire	Résultat de la comparaison
src	src	Les répertoires sont différents
main	src\main	Les répertoires sont différents
docker	src\main\docker	Les répertoires sont différents
java	src\main\java	Les répertoires sont différents
com	src\main\java\com	Les répertoires sont différents
resources	src\main\resources	Les répertoires sont différents
config	src\main\resources\config	Les répertoires sont différents
liquibase	src\main\resources\config\liquibase	Les répertoires sont différents
tls	src\main\resources\config\tls	Les répertoires sont différents
application-dev.yml	src\main\resources\config\tls\application-dev.yml	Les fichiers de texte sont différents
application-prod.yml	src\main\resources\config\tls\application-prod.yml	Les fichiers de texte sont différents
application.yml	src\main\resources\config\tls\application.yml	Les fichiers de texte sont différents
.h2.server.properties	src\main\resources\tls\h2.server.properties	Les fichiers de texte sont différents
webapp	src\main\webapp	Les répertoires sont différents
app	src\main\webapp\app	Les répertoires sont différents
account	src\main\webapp\app\account	Les répertoires sont différents
account.route.ts	src\main\webapp\app\account\account.route.ts	Les fichiers de texte sont différents
core	src\main\webapp\app\core	Les répertoires sont différents
auth	src\main\webapp\app\core\auth	Les répertoires sont différents
interceptor	src\main\webapp\app\core\interceptor	Les répertoires sont différents
auth-expired.interceptor.ts	src\main\webapp\app\core\interceptor\auth-expired.interceptor.ts	Les fichiers de texte sont différents
index.ts	src\main\webapp\app\core\interceptor\index.ts	Les fichiers de texte sont différents
layouts	src\main\webapp\app\layouts	Les répertoires sont différents
navbar	src\main\webapp\app\layouts\navbar	Les répertoires sont différents
navbar.component.html	src\main\webapp\app\layouts\navbar\navbar.component.html	Les fichiers de texte sont différents
login	src\main\webapp\app\login	Les répertoires sont différents
login.service.ts	src\main\webapp\app\login\login.service.ts	Les fichiers de texte sont différents
swagger-ui	src\main\webapp\swagger-ui	Les répertoires sont différents
test	src\main\webapp\test	Les répertoires sont différents
.yo-rc.json		Les fichiers de texte sont différents
pom.xml		Les fichiers de texte sont différents
sonar-project.properties		Les fichiers de texte sont différents

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  
cbca1e849 -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

The screenshot shows a web browser window with the title "Welcome, Java Hipster!" and the URL "localhost:9000". The page has a dark header bar with a logo and navigation links for "Home", "Language", and "Account". A red ribbon banner on the left says "Development Demo vDEV". The main content features a cartoon character with orange hair and a bow tie, and the text "Welcome, Java Hipster! (Demo)". It includes instructions for logging in with default accounts ("Administrator" or "User") and a link to register a new account. Below this, there's a section for questions with links to the JHipster homepage, Stack Overflow, bug tracker, public chat room, and Twitter. At the bottom, there's a GitHub link and a footer note.

Welcome, Java Hipster!

localhost:9000

Home Language Account

Development Demo vDEV

Welcome, Java Hipster! (Demo)

This is your homepage

If you want to [sign in](#), you can try the default accounts:

- Administrator (login="admin" and password="admin")
- User (login="user" and password="user").

You don't have an account yet? [Register a new account](#)

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

Click sign-in => /login page

The screenshot shows a web browser window with the URL `localhost:9000/login`. The page title is "Sign in".

The page includes the following elements:

- Header:** A dark header bar with a blue ribbon icon and the text "Demo vDEV". It also features "Development" and "vDEV" text.
- Top Right:** Navigation icons for refresh, search, and other browser functions.
- Header Buttons:** "Home", "Language", and "Account" buttons.
- Form Fields:** "Username" field containing "admin" and "Password" field containing ".....".
- Checkboxes:** "Remember me" checkbox.
- Buttons:** A blue "Sign in" button.
- Callout Boxes:** Two yellow rectangular boxes at the bottom.
 - The top box contains the text "[Did you forget your password?](#)".
 - The bottom box contains the text "You don't have an account yet? [Register a new account](#)".

Home Page after logged-in

A screenshot of a web browser window displaying the JHipster homepage at localhost:9000. The browser's address bar shows the URL. The page has a dark header with a red diagonal banner on the left containing the text "Development Demo vDEV". On the right side of the header are links for Home, Entities, Administration, Language, and Account. Below the header, there is a large illustration of a woman with orange hair in braids, wearing a white shirt and a blue bow tie. The main content area features a large heading "Welcome, Java Hipster! (Demo)". Below it, a sub-heading says "This is your homepage". A green callout box contains the message "You are logged in as user 'admin'.". Further down, there is a section titled "If you have any question on JHipster:" followed by a bulleted list of links. At the bottom of the page, there is a footer with the text "This is your footer".

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](#)

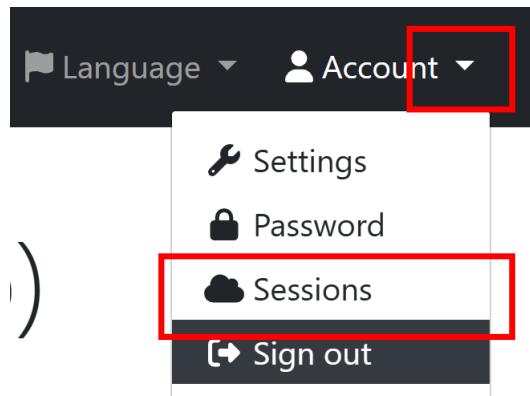
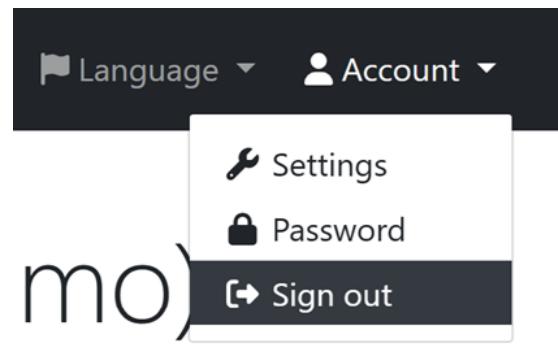
If you like JHipster, don't forget to give us a star on [GitHub](#)!

Different Menu for Account Sub Menu Item "Sessions"

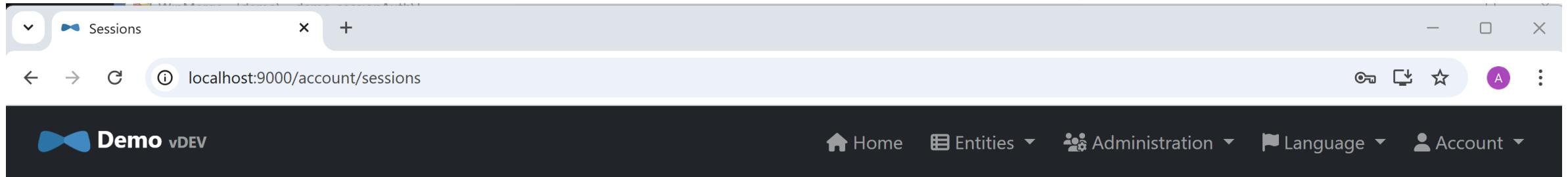
"JWT" Mode



"Http Session" Mode



Sessions View for Current User



Active sessions for [admin]

IP address

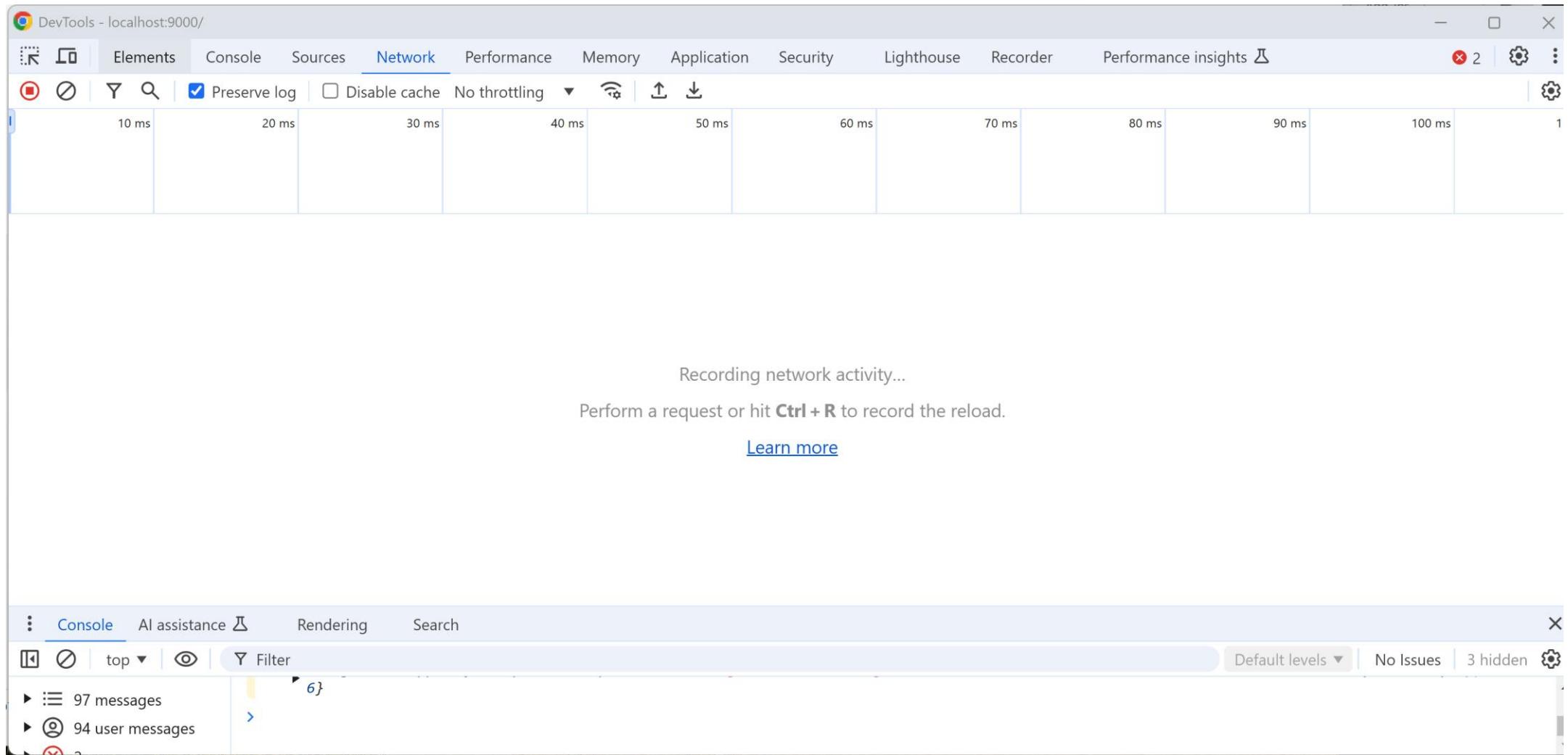
User Agent

Date

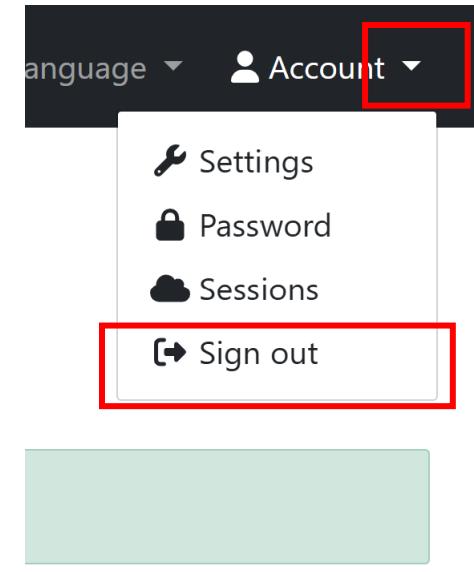
This is your footer

Debuging Security from "Http" viewpoint

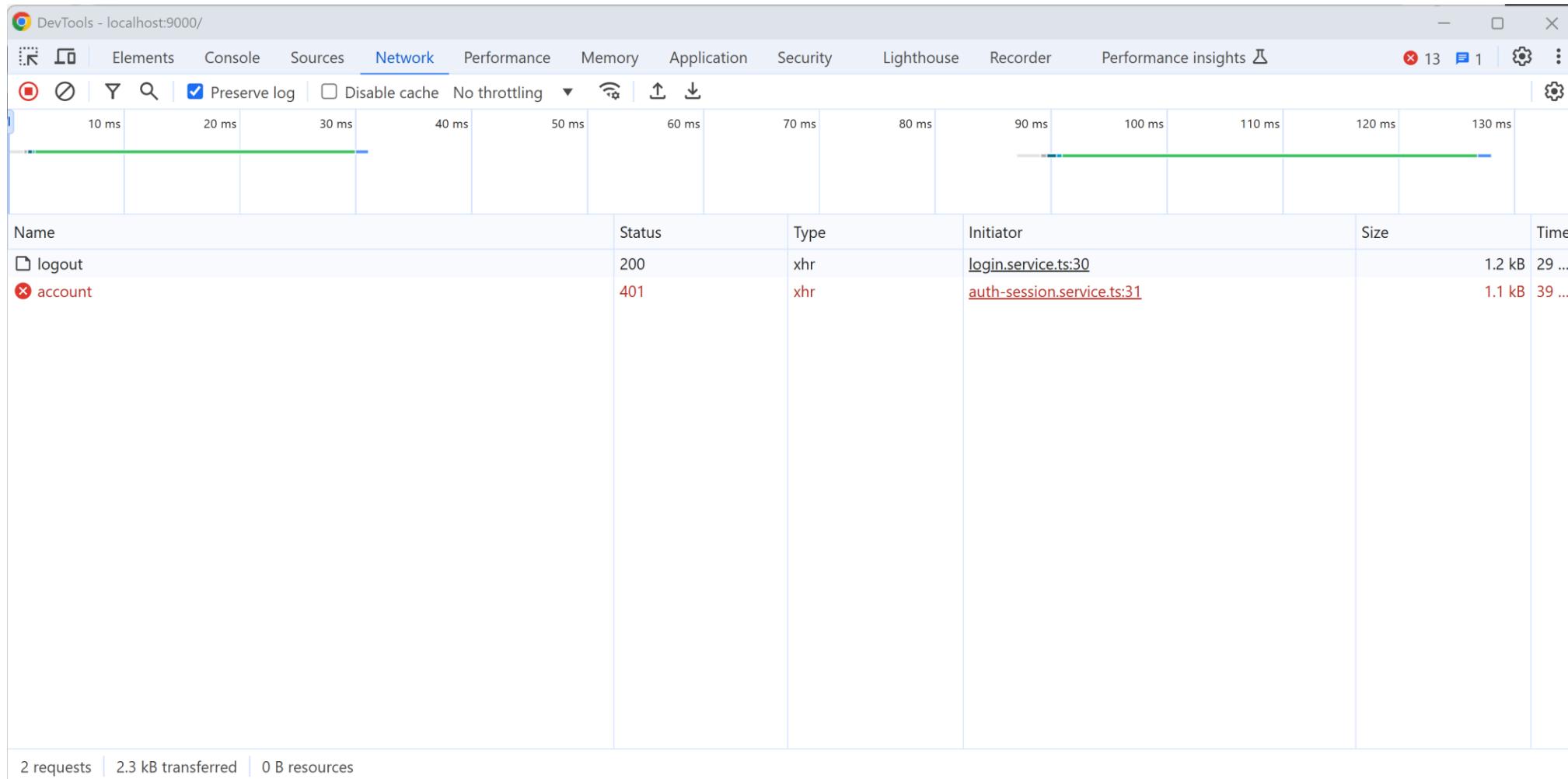
Chrome Dev Tools > Network



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



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



Http POST /api/logout

Name	X Headers	Payload	Preview	Response	Initiator	Timing	Cookies
logout	<p>▼ General</p> <p>Request URL: http://localhost:9000/api/logout</p> <p>Request Method: POST</p> <p>Status Code: 200 OK</p> <p>Remote Address: [::1]:9000</p> <p>Referrer Policy: strict-origin-when-cross-origin</p> <p>► Response Headers (20)</p> <p>▼ Request Headers</p> <p>Raw</p> <p>Accept: application/json, text/plain, */*</p> <p>Accept-Encoding: gzip, deflate, br, zstd</p> <p>Accept-Language: en,fr-FR;q=0.9,fr;q=0.8,en-FR;q=0.7,en-US;q=0.6</p> <p>Connection: keep-alive</p> <p>Content-Length: 2</p> <p>Content-Type: application/json</p> <p>Cookie: shellInABox=3:101010; username=localhost-8888="2 1:0 10:1729931560 23:username=localhost-8888 192:eyJ1c2VybmcFtZSI6ICJjODY1Y2JmMjdjMDA0MmU1OTIzMDRjM2FkMDlhZGI2MiIsICJuYW1lJogIkFub255bW91cyBFbGFyYSIsICJkaXNwbGF5X25hbWUiOiAiQW5vbnnbtb3VzIEVsYXJhliwgImluaXRpYWxzIjogIkFFliwgImNvbG9yljogbnVsbH0= f903b8b6109cb4119762524f861461c7e4a21a4342679e640fd598be87edb95d";</p>						
account							

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

Name	Headers	Payload	Preview	Response	Initiator	Timing	Cookies
logout							
account							
	Access-Control-Allow-Credentials: true Access-Control-Allow-Origin: http://localhost:9000 Access-Control-Expose-Headers: Link, X-Total-Count, X-demoApp-alert, X-demoApp-error, X-demoApp-params Cache-Control: no-cache, no-store, max-age=0, must-revalidate Connection: close Content-Length: 0 Content-Security-Policy: default-src 'self'; frame-src 'self' data; script-src 'self' 'unsafe-inline' 'unsafe-eval' https://storage.googleapis.com; style-src 'self' 'unsafe-inline'; img-src 'self' data; font-src 'self' data; Date: Wed, 20 Nov 2024 21:11:13 GMT Expires: 0 Permissions-Policy: camera=(), fullscreen=(self), geolocation=(), gyroscope=(), magnetometer=(), microphone=(), midi=(), payment=(), sync-xhr=() Pragma: no-cache Referrer-Policy: strict-origin-when-cross-origin Set-Cookie: XSRF-TOKEN=""; path=/; Max-Age=0; Expires=Thu, 01-Jan-1970 00:00:00 GMT Set-Cookie: remember-me=; path=/; Max-Age=0; Expires=Thu, 01-Jan-1970 00:00:00 GMT Strict-Transport-Security: max-age=31536000 ; includeSubDomains Vary: Origin, Access-Control-Request-Method, Access-Control-Request-Headers X-Content-Type-Options: nosniff X-Frame-Options: SAMEORIGIN						

2 requests | 2.3 kB transferred | 0 B resources

Sign-in => 3 Http Requests

The screenshot shows the Network tab in Google Chrome DevTools. The timeline at the top indicates a total duration of 6000 ms. Below the timeline, a table lists three network requests:

Name	Status	Type	Initiator	Size	Ti...
✖ account	401	xhr	login.component.ts:32	1.0 kB	10...
📁 authentication	200	xhr	login.component.ts:44	1.2 kB	15...
⌚ account	200	xhr	login.component.ts:44	1.5 kB	41 ...

At the bottom, the summary shows "3 requests | 3.7 kB transferred | 365 B resources".

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

The screenshot shows a browser developer tools Network tab with a single request listed. The request is for the URL `http://localhost:9000/api/account` using the `GET` method. The status code returned is `401 Unauthorized`. The response headers include `Access-Control-Allow-Origin: *`, `Cache-Control: no-cache, no-store, max-age=0, must-revalidate`, `Connection: close`, `Content-Length: 0`, `Content-Security-Policy: default-src 'self'; frame-src 'self' data; script-src 'self' 'unsafe-inline' 'unsafe-eval' https://storage.googleapis.com; style-src 'self' 'unsafe-inline'; img-src 'self' data; font-src 'self' data;`, `Date: Wed, 20 Nov 2024 21:17:58 GMT`, `Expires: 0`, `Permissions-Policy: camera=(), fullscreen=(self), geolocation=(), gyroscope=(), magnetometer=(), microphone=(), midi=(), payment=(), sync-xhr=()`, `Pragma: no-cache`, `Referrer-Policy: strict-origin-when-cross-origin`, `Strict-Transport-Security: max-age=31536000 ; includeSubDomains`, `Vary: Origin, Access-Control-Request-Method, Access-Control-Request-Headers`, and `X-Content-Type-Options: nosniff`.

Name	Headers	Preview	Response	Initiator	Timing	Cookies
x account	General	Request URL: http://localhost:9000/api/account	Request Method: GET	401 Unauthorized	[::1]:9000	strict-origin-when-cross-origin
authentication	Response Headers	Access-Control-Allow-Origin: *	Cache-Control: no-cache, no-store, max-age=0, must-revalidate	Connection: close	Content-Length: 0	Content-Security-Policy: default-src 'self'; frame-src 'self' data; script-src 'self' 'unsafe-inline' 'unsafe-eval' https://storage.googleapis.com; style-src 'self' 'unsafe-inline'; img-src 'self' data; font-src 'self' data;
account		Date: Wed, 20 Nov 2024 21:17:58 GMT	Expires: 0	Permissions-Policy: camera=(), fullscreen=(self), geolocation=(), gyroscope=(), magnetometer=(), microphone=(), midi=(), payment=(), sync-xhr=()	Pragma: no-cache	Referrer-Policy: strict-origin-when-cross-origin
		Strict-Transport-Security: max-age=31536000 ; includeSubDomains	Vary: Origin, Access-Control-Request-Method, Access-Control-Request-Headers	X-Content-Type-Options: nosniff		

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

Name	X	Headers	Payload	Preview	Response	Initiator	Timing	Cookies	
✖ account									
authentication									
⌚ account									
▼ General									
Request URL:				http://localhost:9000/api/authentication					
Request Method:				POST					
Status Code:				● 200 OK					
Remote Address:				[:1]:9000					
Referrer Policy:				strict-origin-when-cross-origin					
▶ Response Headers (20)									
▼ Request Headers		□							
		Raw							
Accept:				application/json, text/plain, */*					
Accept-Encoding:				gzip, deflate, br, zstd					
Accept-Language:				en,fr-FR;q=0.9,fr;q=0.8,en-FR;q=0.7,en-US;q=0.6					
Connection:				keep-alive					
Content-Length:				60					
Content-Type:				application/x-www-form-urlencoded					
Cookie:				shellInABox=3:101010; username=localhost-8888="2 1:0 10:1729931560 23:username=localhost-8888 192:eyJ1c2VybmcFtZSI6ICJiODY1Y2JmMjdjMDA0MmU1OTIzM0RjM2FkMDlhZG12MilsICJuYW1lljogIkFub255bW91cyBFbGFyYSIsICJkaXNwbGF5X25hbWUiOiAiQW5vbndlbt3VzIEVsYXJhliwglmluaXRpYWxzljoglkkFFliwglmNvbG9yljogbnVsbH0= f903b8b6109cb4119762524f861461c7e4a21a4342679e640fd598be87edb95d";_xsrf=2 dc7823e1 02bf7e8cc9e6846a2ae45bbce25a0d92 1729931560; XSRF-TOKEN=aa604feb-715f-4b36-9c7c-c9d38bb2a9a4					

Request [2/3] Payload : username & password

X Headers **Payload** Preview Response Initiator Timing Cookies

▼Form Data **view parsed**

username=admin&password=admin&remember-me=false&submit=Login

X Headers **Payload** Preview Response Initiator Timing Cookies

▼Form Data **view source** view URL-encoded

username: admin
password: admin
remember-me: false
submit: Login

```
graph TD; A[view source] --> B[view source]; C[view parsed] --> D[view parsed]
```

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

"JWT" Mode



"Http Session" Mode

Payload Request Body was in JSON

X	Headers	<u>Payload</u>	Preview	Response	Initiator	Timing	Cookies
▼ Request Payload view source							
▼ {username: "admin", password: "admin", rememberMe: false}							
password: "admin"							
rememberMe: false							
username: "admin"							

Payload Request in "URL-encoded"

X	Headers	<u>Payload</u>	Preview	Response	Initiator	Timing	Cookies
▼ Form Data view parsed							
username=admin&password=admin&remember-me=false&submit=Login							

X	Headers	<u>Payload</u>	Preview	Response	Initiator	Timing	Cookies
▼ Form Data view source view URL-encoded							
username: admin							
password: admin							
remember-me: false							
submit: Login							

Request [2/3] Response Body : EMPTY !

Request [2/3] : Response Header

Set-Cookie: "JSESSIONID=xxxx"

Name	Headers	Payload	Preview	Response	Initiator	Timing	Cookies
✖ account							
.authentication	▼ Response Headers						
⌚ account							
	Access-Control-Allow-Credentials:	true					
	Access-Control-Allow-Origin:	http://localhost:9000					
	Access-Control-Expose-Headers:	Link, X-Total-Count, X-demoApp-alert, X-demoApp-error, X-demoApp-params					
	Cache-Control:	no-cache, no-store, max-age=0, must-revalidate					
	Connection:	close					
	Content-Length:	0					
	Content-Security-Policy:	default-src 'self'; frame-src 'self' data;; script-src 'self' 'unsafe-inline' 'unsafe-eval' https://storage.googleapis.com; style-src 'self' 'unsafe-inline'; img-src 'self' data;; font-src 'self' data;					
	Date:	Wed, 20 Nov 2024 21:18:03 GMT					
	Expires:	0					
	Permissions-Policy:	camera=(), fullscreen=(self), geolocation=(), gyroscope=(), magnetometer=(), microphone=() (), midi=(), payment=(), sync-xhr=()					
	Pragma:	no-cache					
	Referrer-Policy:	strict-origin-when-cross-origin					
	Set-Cookie:	JSESSIONID=ub_W21OY6ZI3F7gtlk_zKvDymp4MjYZEA9rlInf9; path=/; HttpOnly					
	Set-Cookie:	XSRF-TOKEN=""; path=/; Max-Age=0; Expires=Thu, 01-Jan-1970 00:00:00 GMT					
	Strict-Transport-Security:	max-age=31536000 ; includeSubDomains					
	Vary:	Origin, Access-Control-Request-Method, Access-Control-Request-Headers					
	X-Content-Type-Options:	nosniff					
	X-Frame-Options:	SAMEORIGIN					

Request [2/3] Difference with JWT Mode

"JWT" Mode



"Http Session" Mode

X	Headers	Payload	Preview	Response	Initiator	Timing	Cookies
---	---------	---------	---------	----------	-----------	--------	---------

```
1 {  
2   "id_token" : "eyJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJhZG1pbjIsImV4cC  
3 }
```

Response Body was in JSON,
containing the JWT Token

X	Headers	Payload	Preview	Response	Initiator	Timing	Cookies
---	---------	---------	---------	----------	-----------	--------	---------

1

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

Name	Headers	Preview	Response	Initiator	Timing	Cookies										
✖ account																
☐ authentication																
⌚ account	<p>▼ General</p> <p>Request URL: http://localhost:9000/api/account</p> <p>Request Method: GET</p> <p>Status Code: 200 OK</p> <p>Remote Address: [:1]:9000</p> <p>Referrer Policy: strict-origin-when-cross-origin</p> <p>► Response Headers (19)</p> <p>▼ Request Headers</p> <table><thead><tr><th>Raw</th></tr></thead><tbody><tr><td>Accept: application/json, text/plain, */*</td></tr><tr><td>Accept-Encoding: gzip, deflate, br, zstd</td></tr><tr><td>Accept-Language: en,fr-FR;q=0.9,fr;q=0.8,en-FR;q=0.7,en-US;q=0.6</td></tr><tr><td>Connection: keep-alive</td></tr><tr><td>Cookie: shellInABox=3:101010; username=localhost-8888="2 1:0 10:1729931560 23:username=localhost-8888 192:eyJc2VybMftZSI6ICJiODY1Y2JmMjdjMDA0MmU1OTIzMDrjM2FkMDlhZGl2MilsICJuYW1lljoglKfub255bW91cyBFbGFyYSIsICJkaXNwbGF5X25hbWUiOiAiQW5vbndlbt3VzIEVsYXJhliwgImluaXRpYWxzljoglKFFliwglmNvbG9yljogbnVsbH0= f903b8b6109cb4119762524f861461c7e4a21a4342679e640fd598be87edb95d";_xsrf=2 dc7823e1 02bf7e8cc9e6846a2ae45bbce25a0d92 1729931560;JSESSIONID=ub_W21OY6Zl3F7gtlk_zKvDymp4MjYZEA9rlINF9</td></tr><tr><td>Host: localhost:9000</td></tr><tr><td>Referer: http://localhost:9000/login</td></tr><tr><td>Sec-Ch-Ua: "Google Chrome";v="131", "Chromium";v="131", "Not_A Brand";v="24"</td></tr><tr><td>Sec-Ch-Ua-Mobile: ?0</td></tr></tbody></table>	Raw	Accept: application/json, text/plain, */*	Accept-Encoding: gzip, deflate, br, zstd	Accept-Language: en,fr-FR;q=0.9,fr;q=0.8,en-FR;q=0.7,en-US;q=0.6	Connection: keep-alive	Cookie: shellInABox=3:101010; username=localhost-8888="2 1:0 10:1729931560 23:username=localhost-8888 192:eyJc2VybMftZSI6ICJiODY1Y2JmMjdjMDA0MmU1OTIzMDrjM2FkMDlhZGl2MilsICJuYW1lljoglKfub255bW91cyBFbGFyYSIsICJkaXNwbGF5X25hbWUiOiAiQW5vbndlbt3VzIEVsYXJhliwgImluaXRpYWxzljoglKFFliwglmNvbG9yljogbnVsbH0= f903b8b6109cb4119762524f861461c7e4a21a4342679e640fd598be87edb95d";_xsrf=2 dc7823e1 02bf7e8cc9e6846a2ae45bbce25a0d92 1729931560;JSESSIONID=ub_W21OY6Zl3F7gtlk_zKvDymp4MjYZEA9rlINF9	Host: localhost:9000	Referer: http://localhost:9000/login	Sec-Ch-Ua: "Google Chrome";v="131", "Chromium";v="131", "Not_A Brand";v="24"	Sec-Ch-Ua-Mobile: ?0					
Raw																
Accept: application/json, text/plain, */*																
Accept-Encoding: gzip, deflate, br, zstd																
Accept-Language: en,fr-FR;q=0.9,fr;q=0.8,en-FR;q=0.7,en-US;q=0.6																
Connection: keep-alive																
Cookie: shellInABox=3:101010; username=localhost-8888="2 1:0 10:1729931560 23:username=localhost-8888 192:eyJc2VybMftZSI6ICJiODY1Y2JmMjdjMDA0MmU1OTIzMDrjM2FkMDlhZGl2MilsICJuYW1lljoglKfub255bW91cyBFbGFyYSIsICJkaXNwbGF5X25hbWUiOiAiQW5vbndlbt3VzIEVsYXJhliwgImluaXRpYWxzljoglKFFliwglmNvbG9yljogbnVsbH0= f903b8b6109cb4119762524f861461c7e4a21a4342679e640fd598be87edb95d";_xsrf=2 dc7823e1 02bf7e8cc9e6846a2ae45bbce25a0d92 1729931560;JSESSIONID=ub_W21OY6Zl3F7gtlk_zKvDymp4MjYZEA9rlINF9																
Host: localhost:9000																
Referer: http://localhost:9000/login																
Sec-Ch-Ua: "Google Chrome";v="131", "Chromium";v="131", "Not_A Brand";v="24"																
Sec-Ch-Ua-Mobile: ?0																
3 requests	3.7 kB transferred	365 B resources														

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

Name	X Headers	Preview	Response	Initiator	Timing	Cookies
✖ account						
☐ authentication						
⌚ account						
	▼ General					
	Request URL:	http://localhost:9000/api/account				
	Request Method:	GET				
	Status Code:	● 200 OK				
	Remote Address:	[::1]:9000				
	Referrer Policy:	strict-origin-when-cross-origin				
	► Response Headers (19)					
	▼ Request Headers					
	Accept:	application/json, text/plain, */*				
	Accept-Encoding:	gzip, deflate, br, zstd				
	Accept-Language:	en,fr-FR;q=0.9,fr;q=0.8,en-FR;q=0.7,en-US;q=0.6				
	Connection:	keep-alive				
	Cookie:	<pre>shellInABox=3:101010; username-localhost-8888="2 1:0 10:1729931560 23:username-localhost-8888 192:eyJ1c2VybmFtZSI6ICJiODY1Y2JmMjdjMDA0MmU1OTIzMjRjM2FkMDlhZGI2MilsICJuYW1lIjogIkFub255bW91cyBFbGFyYSIsICJkaXNwbGF5X25hbWUiOiAiQW5vbnlb3VzIEVsYXJhliwgImluaXRpYWxzljogIkFFliwgImNvbG9yljogbnVsbH0= f903b8b6109cb4119762524f861461c7e4a21a4342679e640fd598be87edb95d";xsrf=2 dc7823e1 02bf7e8cc9e6846a2ae45bbce25a0d92 1729931560;JSESSIONID=ub_W21OY6ZI3F7gtlk_zKvDymp4MjYZEA9rlINf9</pre>				
	Host:	localhost:9000				
	Referer:	http://localhost:9000/login				
	Sec-Ch-Ua:	"Google Chrome";v="131", "Chromium";v="131", "Not_A Brand";v="24"				
	Sec-Ch-Ua-Mobile:	?				

3 requests | 3.7 kB transferred | 365 B resources

Difference Authentication with JWT Mode

"JWT" Mode ↔ "Http Session" Mode

Authorization: Bearer
eyJhbGciOiJIUzUxMiJ9.eyJzdWlOIjhZG1pbilsImV4cCI6MTcz
U0VSLiwiaWF0IjoxNzMyMDQ3OTA0fQ.44dwV-tYt34U5CpaS
OGQKph1BprwpWYt5whgKsTCCilNi4lvacdqM2eRTJ30NsA

Cookie: shellInABox=3:101010; username=localhost-8888="2|1:0|10:1729931560
8888|192:eyJ1c2VybmFtZSI6ICJiODY1Y2JmMjdjMDA0MmU1OTIzM**DRjM**
W1IjogIkFub255bW91cyBFbGFyYSIsICJkaXNwbGF5X25hbWUiOiAiQW5s
mluaXRpYWxzIjogIkFFliwgImNvbG9yljogbnVsbH0=|f903b8b6109cb411s
1a4342679e640fd598be87edb95d";
_xsrf=2|dc7823e1|02bf7e8cc9e6846a2ae45bbce25a0d92|1729931560;
JSESSIONID=ub_W21OY6ZI3F7gtIk_zKvDymp4MjYZEA9rlINf9

the JWT secret is passed via Http Header
"Authorization": "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

The screenshot shows the Network tab in Google Chrome DevTools for the URL `localhost:9000/`. The response for the request to `/api/account` is selected. The response body is displayed in a code editor-like interface with line numbers:

```
1  {
2    "id" : 1,
3    "login" : "admin",
4    "firstName" : "Administrator",
5    "lastName" : "Administrator",
6    "email" : "admin@localhost",
7    "imageUrl" : "",
8    "activated" : true,
9    "langKey" : "en",
10   "createdBy" : "system",
11   "createdDate" : null,
12   "lastModifiedBy" : "system",
13   "lastModifiedDate" : null,
14   "authorities" : [ "ROLE_USER", "ROLE_ADMIN" ]
15 }
```

At the bottom of the DevTools interface, the statistics are:

3 requests | 3.8 kB transferred | 602 B resources

NO Difference for Authorization with JWT Mode

"JWT" Mode  "Http Session" Mode

NO difference

in both cases, ROLE_* are retrieved 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 UserDetailsService

```
© SecurityConfiguration.java × : 2 ▾ ▾  
38     public class SecurityConfiguration {  
58         public SecurityFilterChain filterChain(HttpSecurity http, MvcRequestMatcher.Builder mvc) throws Exception {  
108             .exceptionHandling( ExceptionHandlingConfigurer<HttpSecurity> exceptionHandling ->  
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 SEvers 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

```
TS auth-session.service.ts × :  
1  > import ... ✓  
8  
9  @Injectable({ providedIn: 'root' }) Show usages  
10 export class AuthServiceProvider {  
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 !

```
ts index.ts ×
1  > import ...
6
7  export const httpInterceptorProviders :({ provide: InjectionToken<readonly
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 ];
```

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 ];
```

```
ts index.ts ✘
1 > import ...
2
3
4
5
6
7 export const httpInterceptorProviders :({ provide: InjectionToken<readonly
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 ];
```

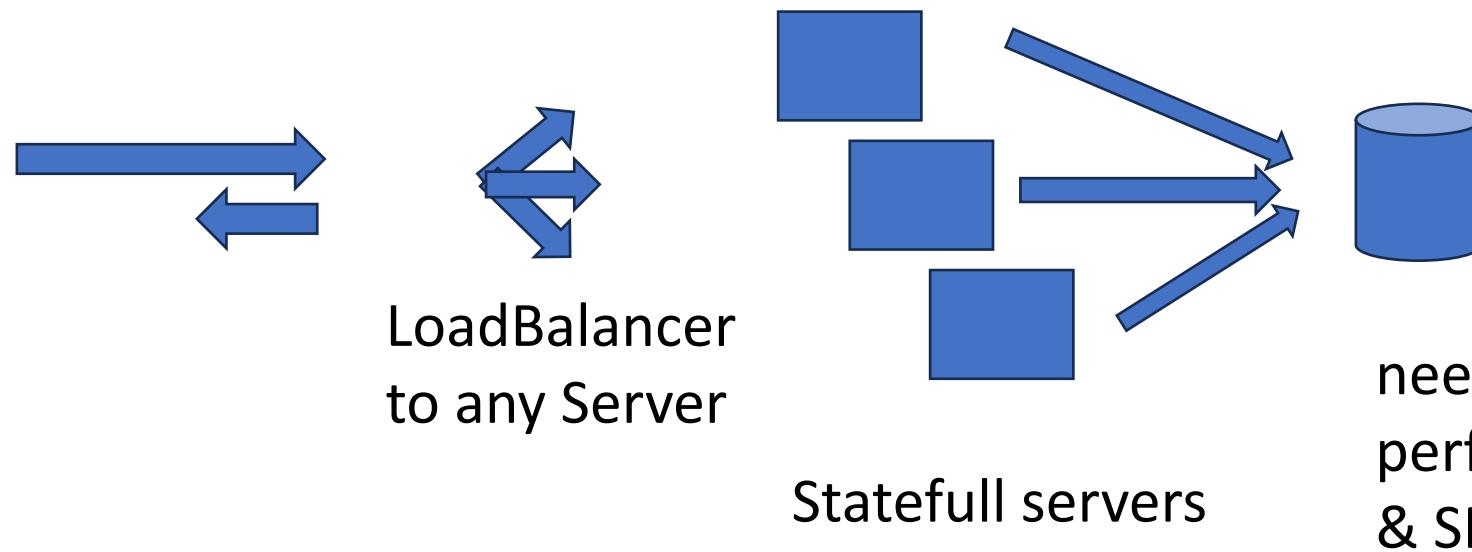
Authorisations

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

Please see previous detailed document on JWT

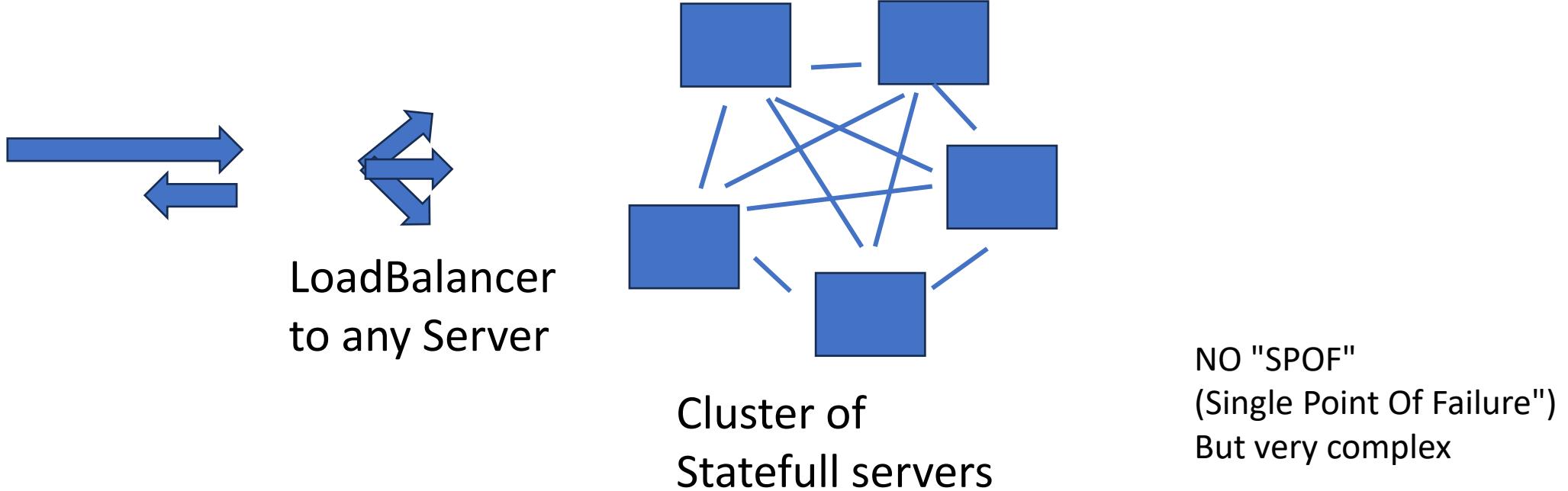
Network & Infrastructure Deployments Options

Servers are Statefull - Need Centralised Database



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

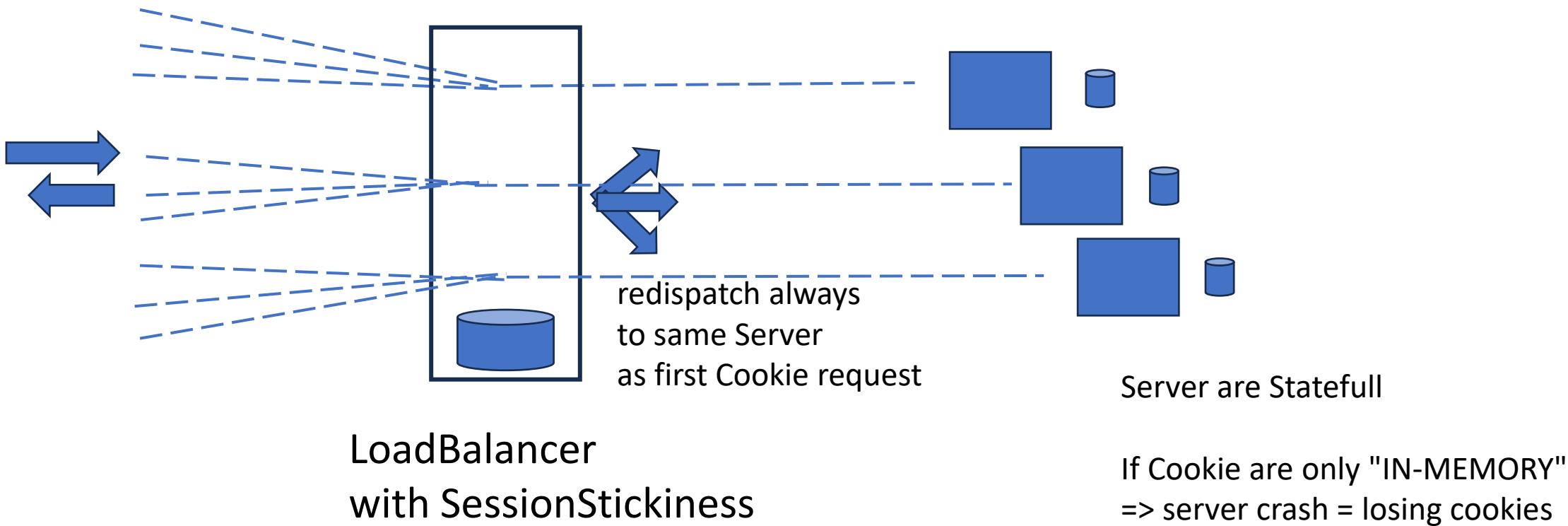
Server Connected in Cluster "Distributed Cache" (=idem embedded database)



LoadBalancer with "Session Stickiness"

Statefull Server

(local DB or in-memory not persisted)



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