Security Hands-On 1

Esilv 2025

arnaud.nauwynck@gmail.com

TD Outline

TD Goals = debug yourself the basic Http authentication scheme =

1/ install "Java JDK"
2/ install IDE : "IntelliJ IDEA"
3/ setup your app, using spring "initializr" click+download
4/ run your app
5/ open http://localhost:8080
  check app is a secured http webapp, with login/password dialog
6/ inspect network requests
7/ test Http requests

TD Outline

TD Goals = debug yourself the basic Http authentication scheme =

1/ install "Java JDK"
2/ install IDE : "IntelliJ IDEA"
3/ setup your app, using spring "initializr" click+download
4/ run your app
5/ open http://localhost:8080
check app is a secured http webapp, with login/password dialog
6/ inspect network requests
7/ test Http requests

Pre-Requisite 1 : Install Java JDK

JDK is Open-Source, but pre-built binaries may be licenced if downloading from oracle.com

on linux => "apt-get install default-jdk"

on Mac => "brew install openjdk"

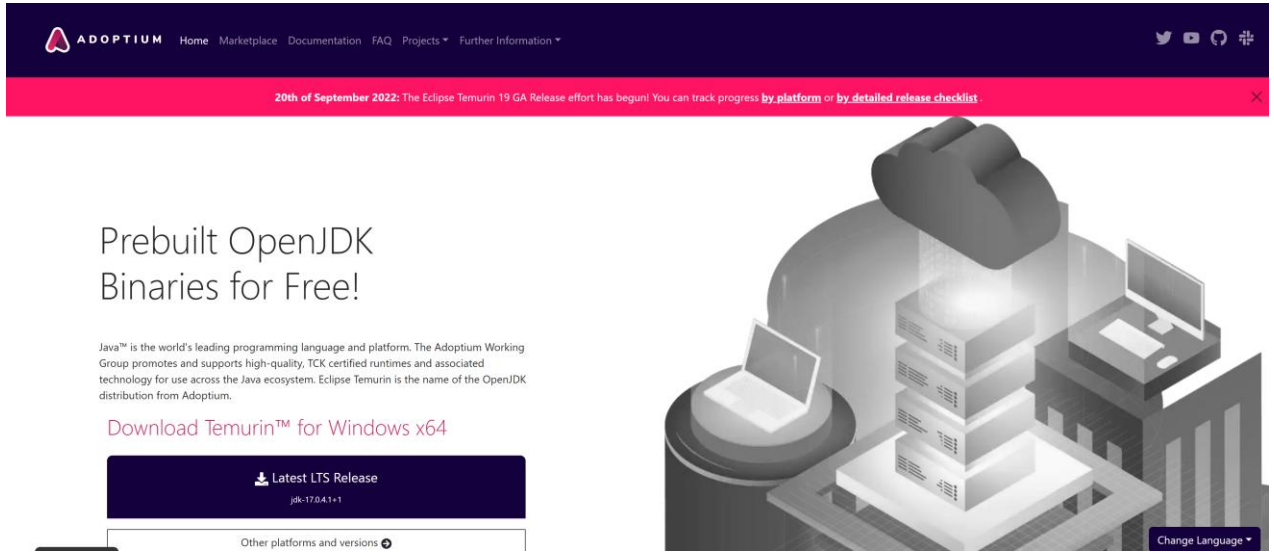
on Windows => download adoptium temurin .msi installer
+ execute (see next slides)

Step 1 : Install Java JDK, download (~170 Mo)

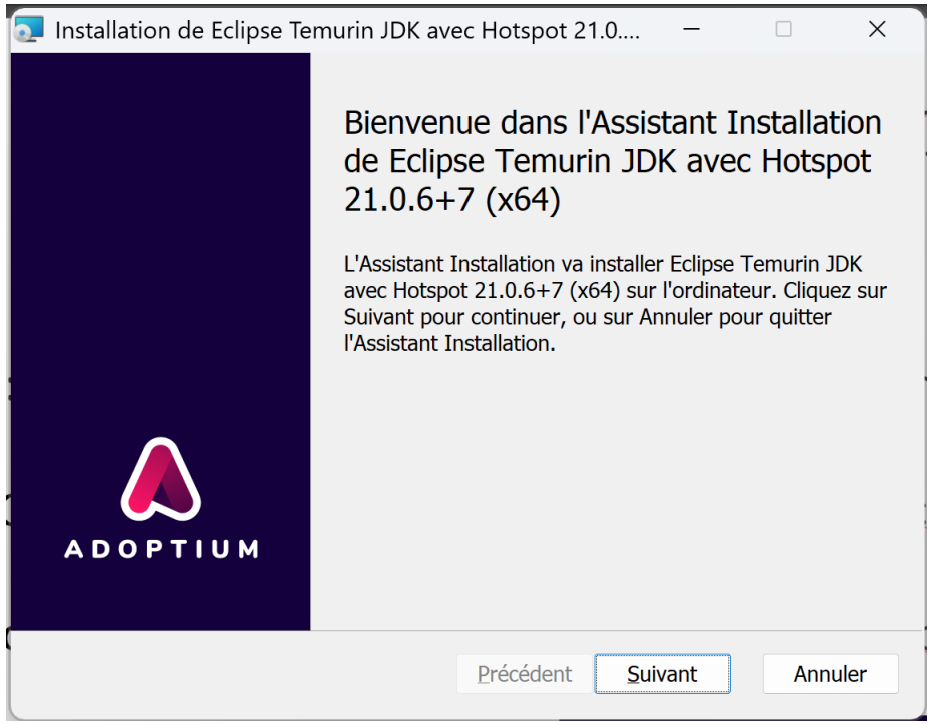
JDK is Open-Source, but pre-built binaries may be licenced if downloading from oracle.com

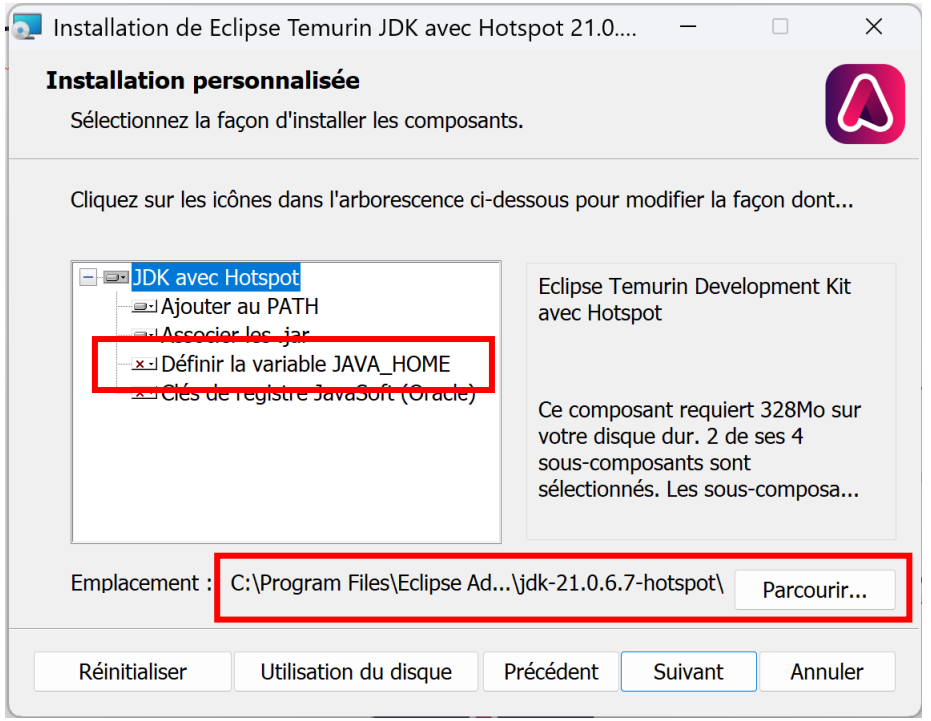
Download from « adoptium » (previously « adoptOpenJdk »)

https://adoptium.net/



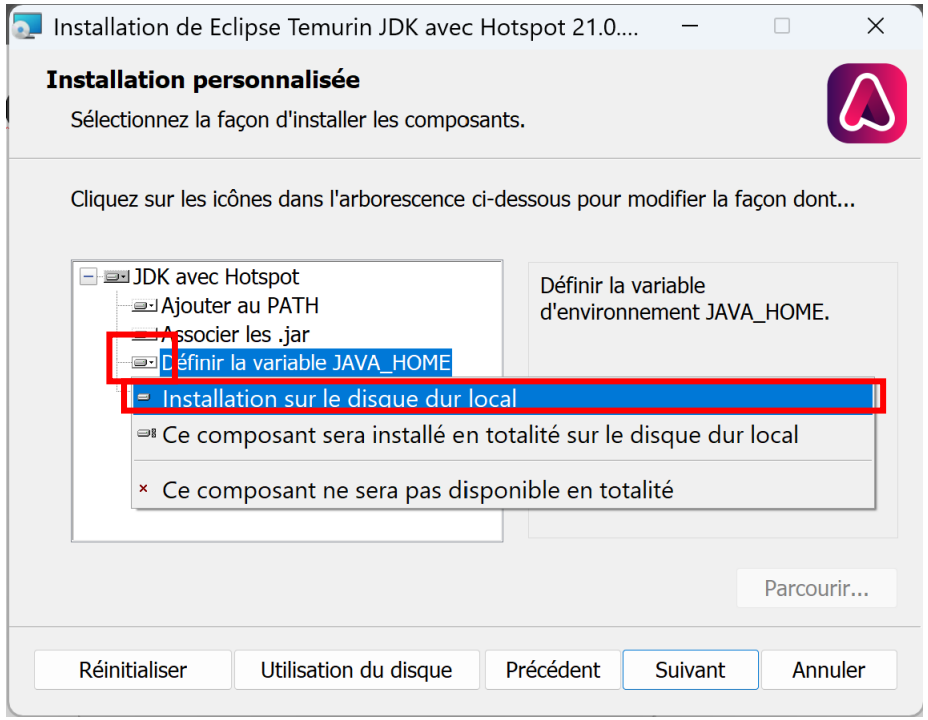
Installing ...

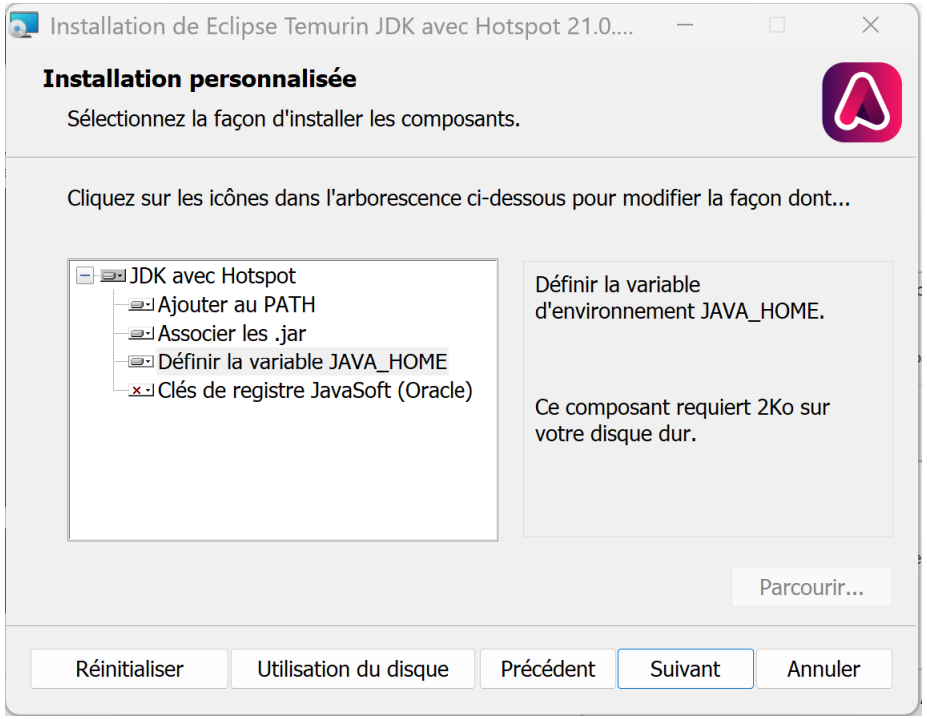




You may change defaults ... see next

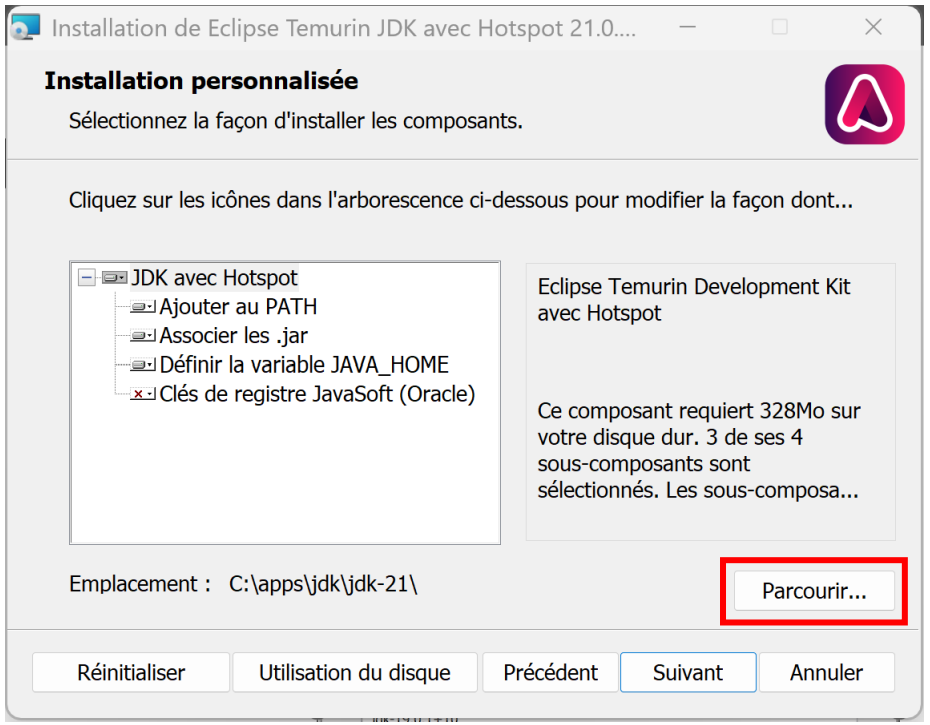
Optional :
defining env variable JAVA_HOME

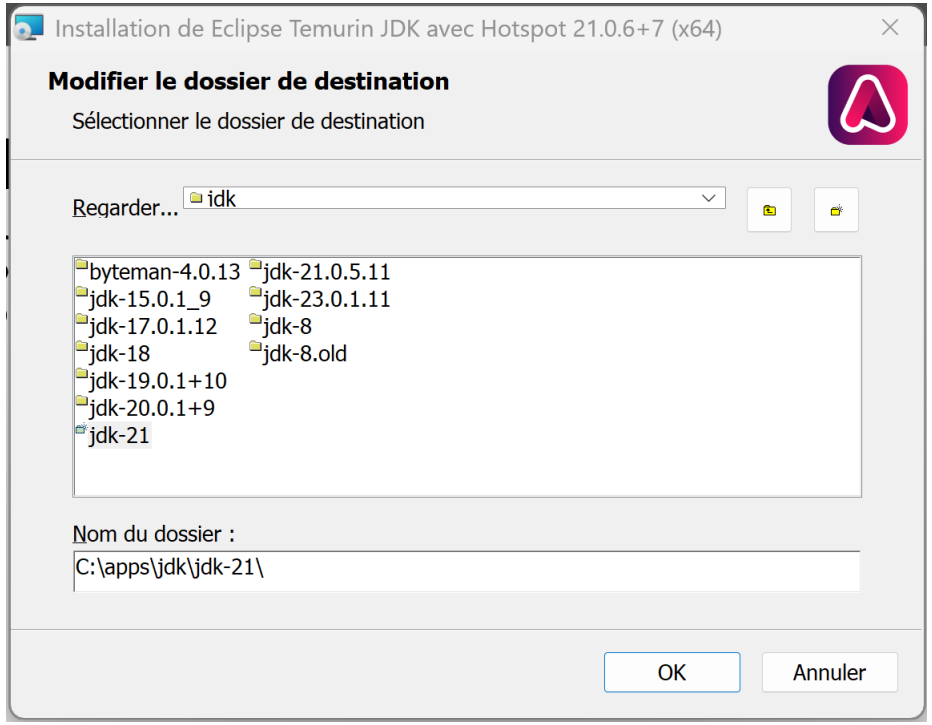




Optional

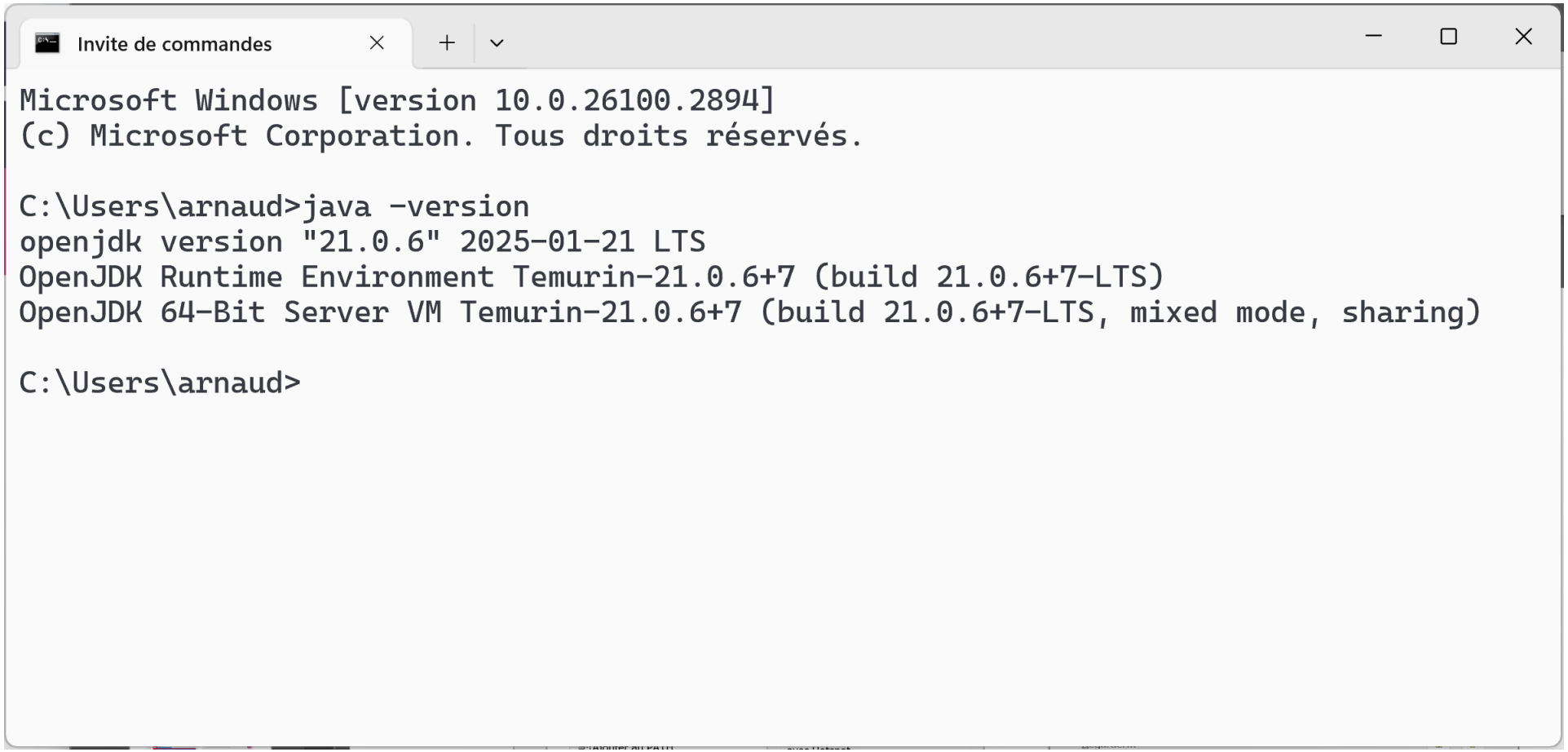
Changing directory "c:\apps\jdk"
(avoid long names, and space in dir name
so avoid "Program & Files\...")





Check Java installation OK

C:> java -version



TD Outline

TD Goals = debug yourself the basic Http authentication scheme =

1/ install "Java JDK"
2/ install IDE : "IntelliJ IDEA"
3/ setup your app, using spring "initializr" click+download
4/ run your app
5/ open http://localhost:8080
  check app is a secured http webapp, with login/password dialog
6/ inspect network requests
7/ test Http requests

Pre-Requisite 2 : Install IDE "IntelliJ IDEA"

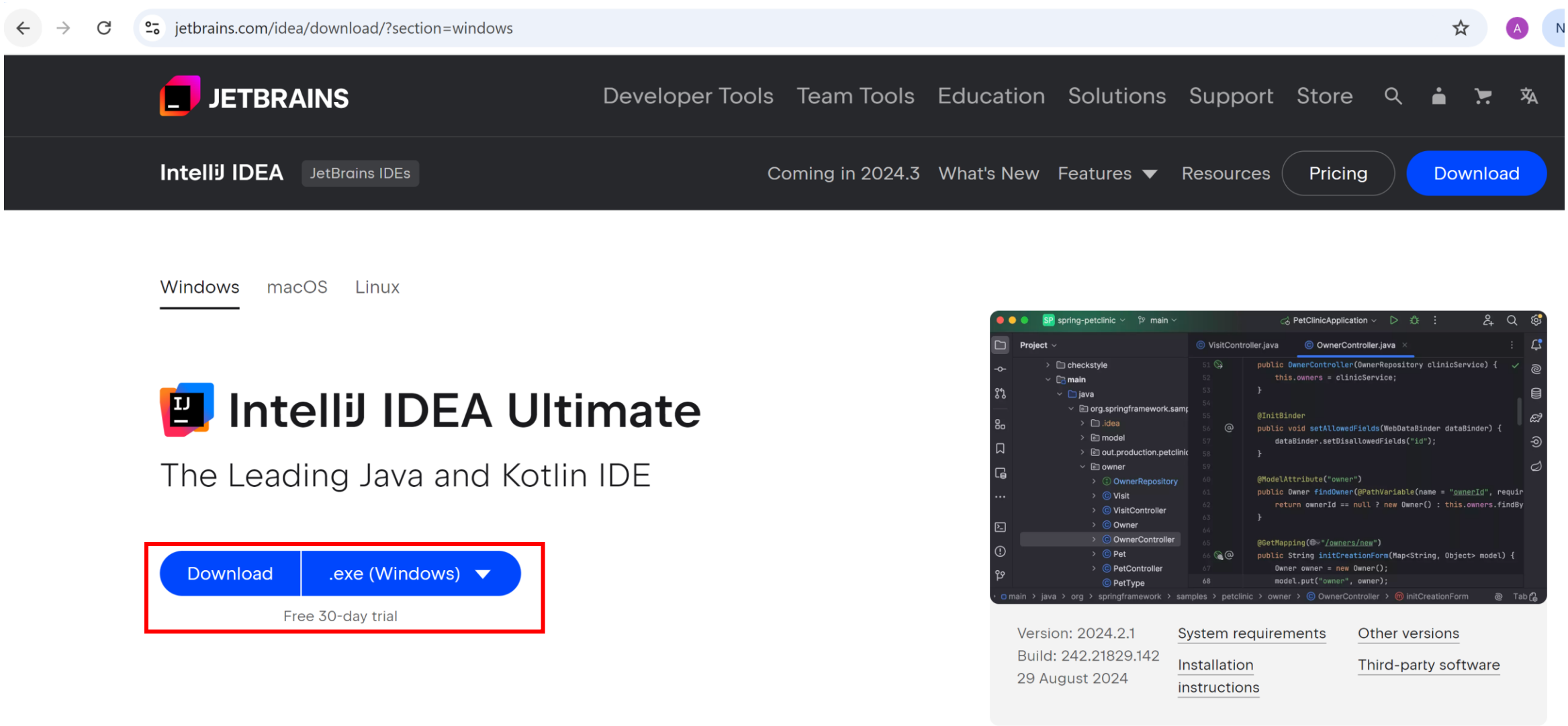
IntelliJ IDEA is the recommended IDE to use

it is the mostly used IDE in the world

100% ready and easy to use for java+maven+http+web+...

Download IntelliJ IDEA

https://www.jetbrains.com/idea/download



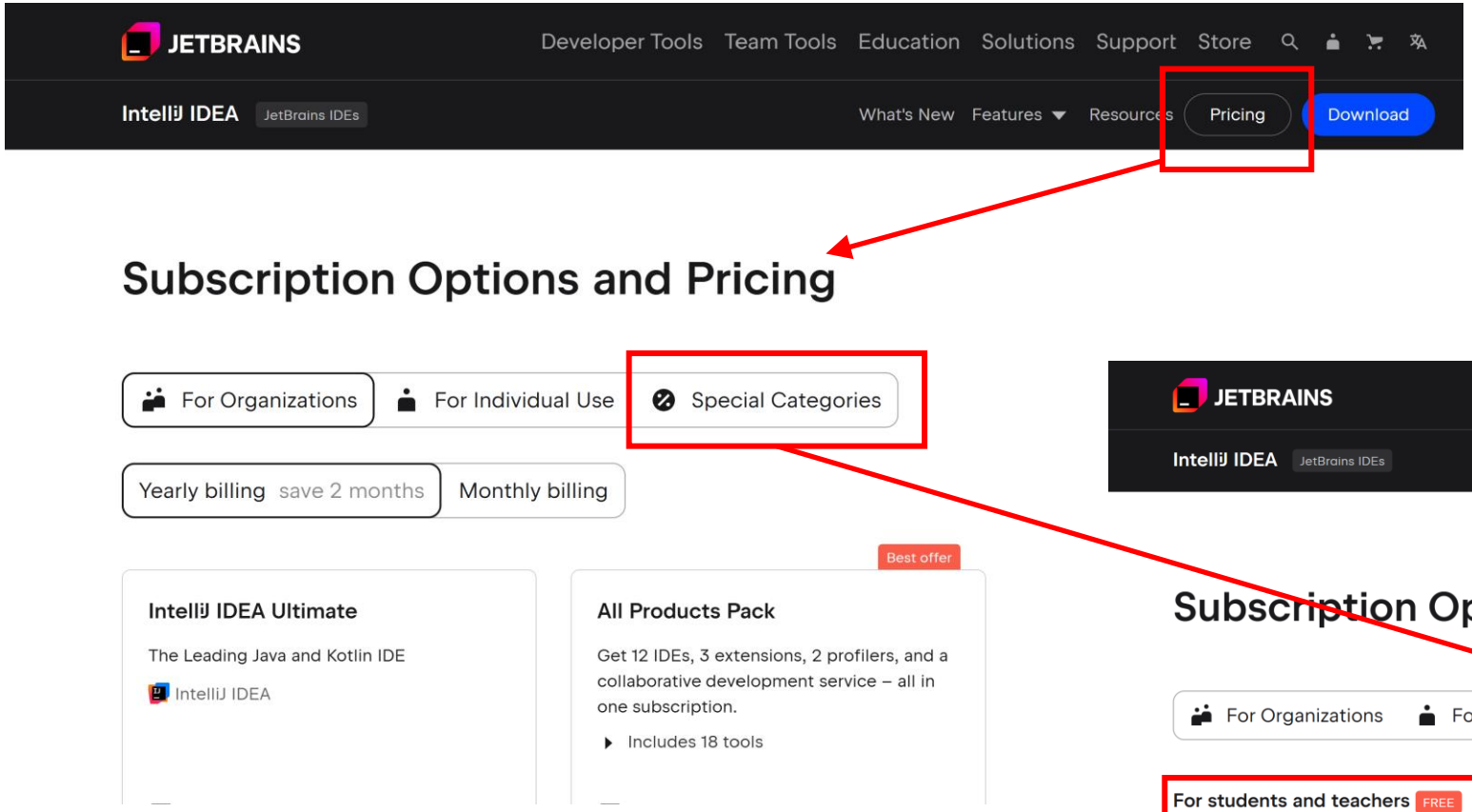
Free 30-days trial -OR- register free 1 year

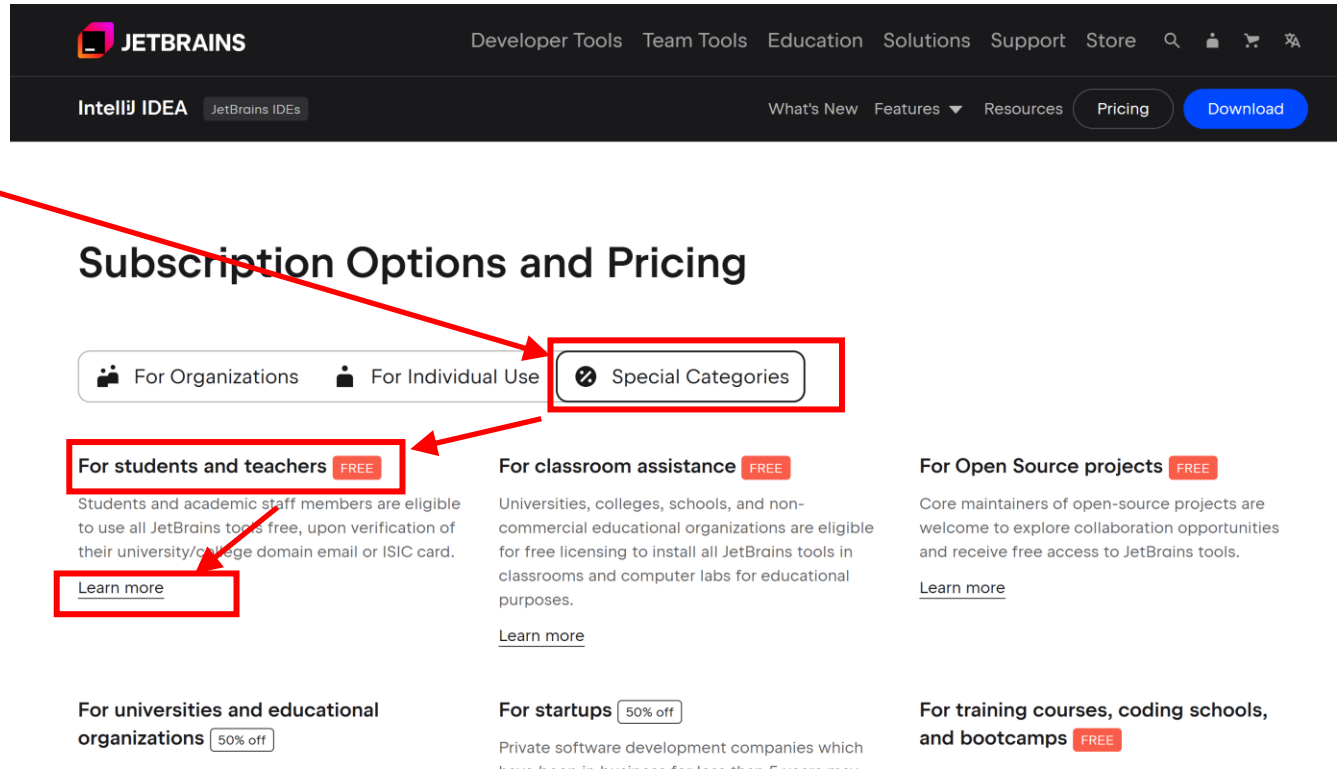
version "Community" is FREE (but supports only java)

version "Ultimate" is much more complete
... and is FREE for students
... is paid by most companies for all developpers who ask

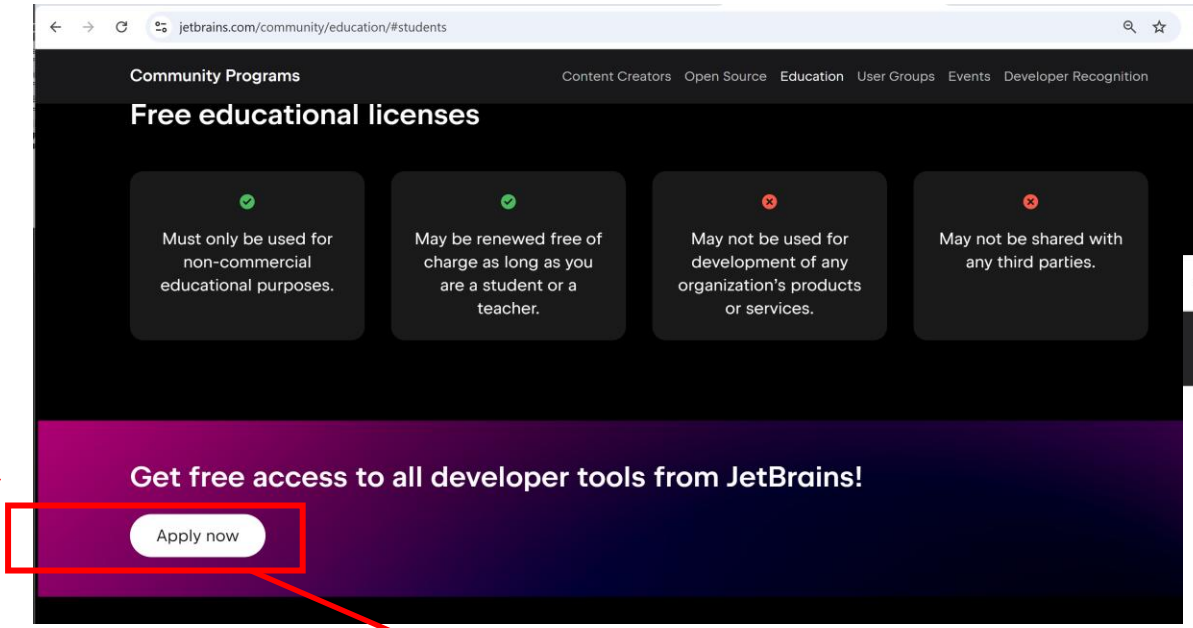
For doing today TD, you can just use "30-day trial"
if you have time (or later at home), you can fill the registration page with your "@esilv" email (see next slides)

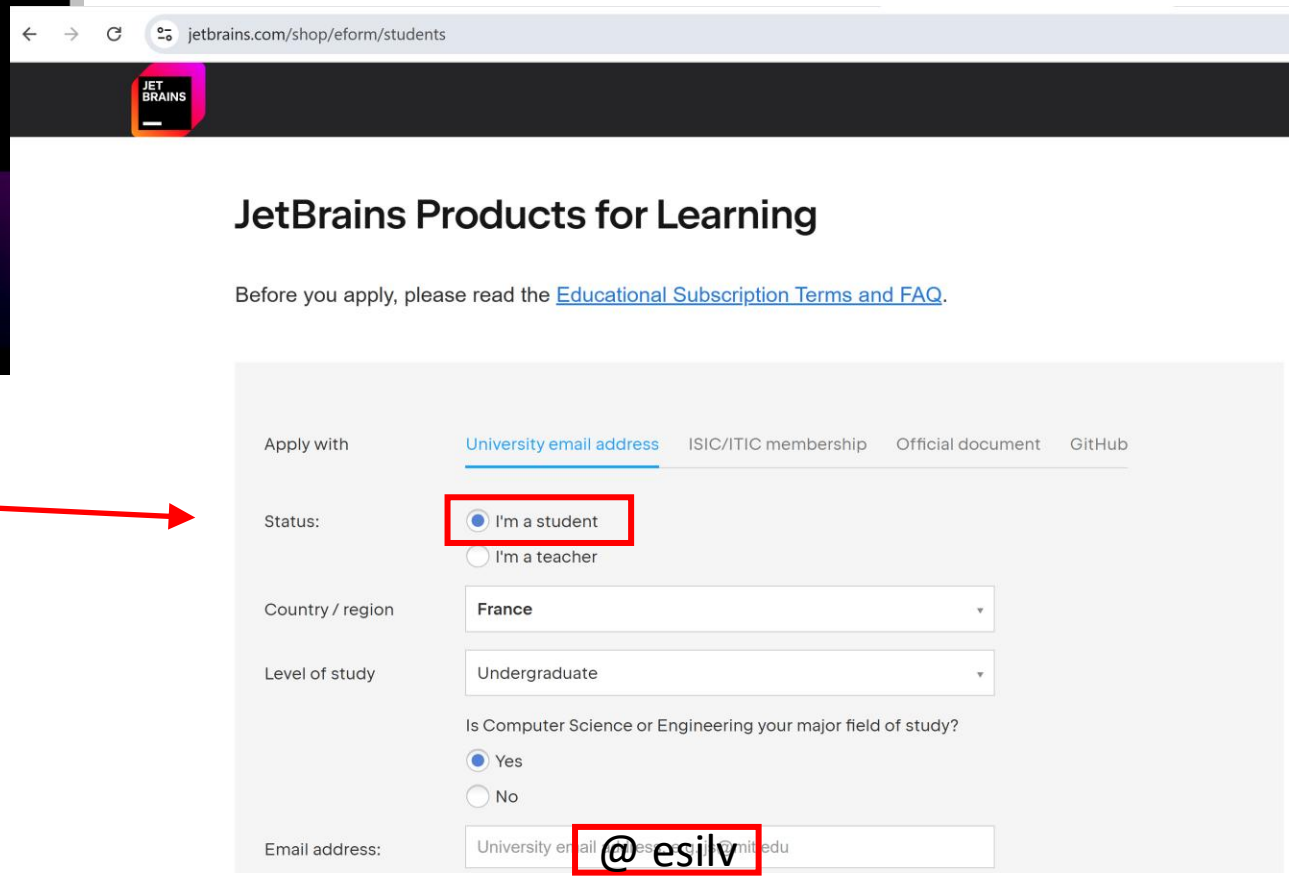
Register IntelliJ IDEA "Ultimate" as Student





Steps …


jetbrains.com/community/education/#students
Community Programs
Content Creators
Open Source
Education
User Groups
Events
Developer Recognition
Free educational licenses
Must only be used for non-commercial educational purposes.
May be renewed free of charge as long as you are a student or a teacher.
May not be used for development of any organization's products or services.
May not be shared with any third parties.
Get free access to all developer tools from JetBrains!
Apply now

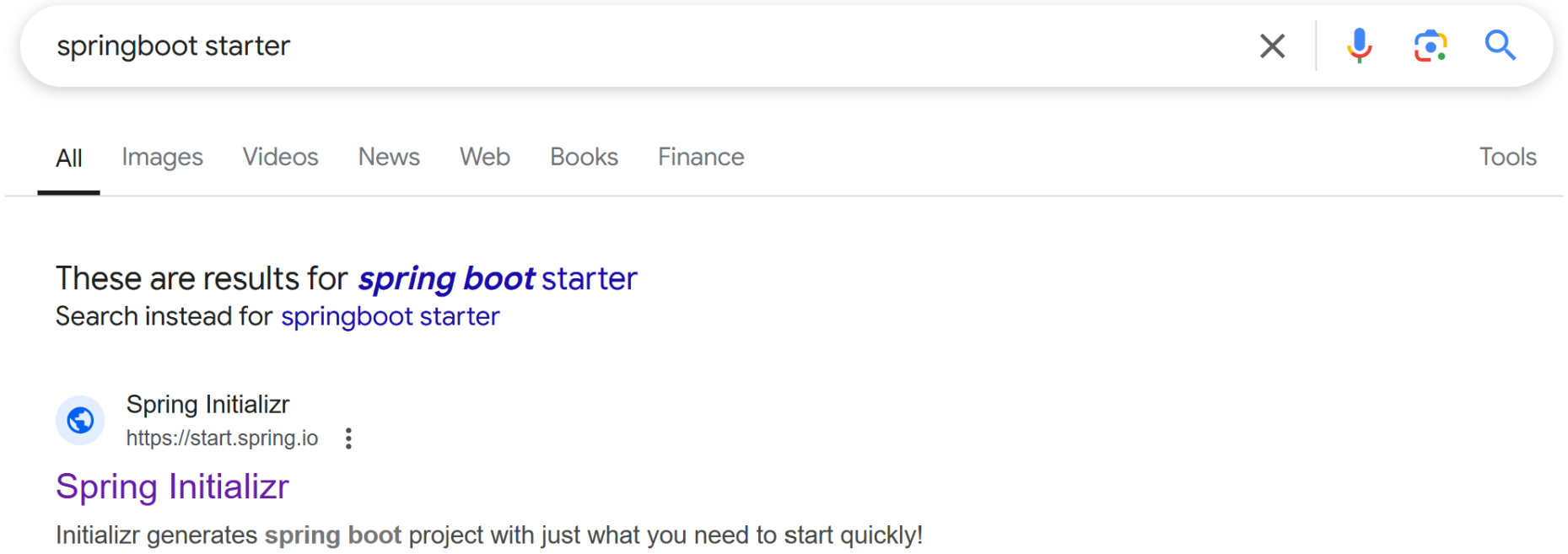

jetbrains.com/shop/eform/students
JET BRAINS
JetBrains Products for Learning
Before you apply, please read the Educational Subscription Terms and FAQ.
Apply with
University email address
ISIC/ITIC membership
Official document
GitHub
Status:
I'm a student
I'm a teacher
Country / region
France
Level of study
Undergraduate
Is Computer Science or Engineering your major field of study?
Yes
No
Email address:
@esilv

TD Outline

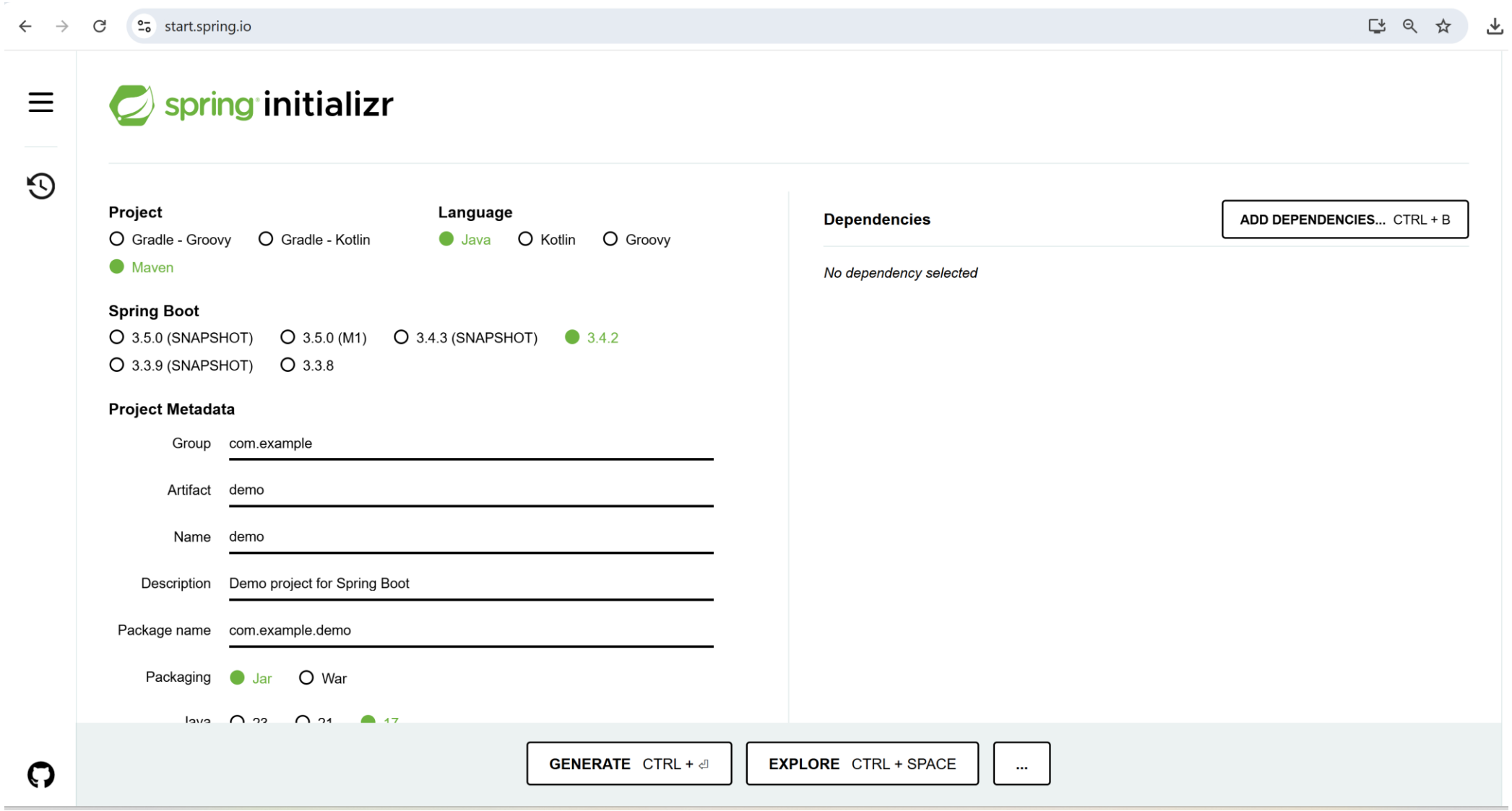
TD Goals = debug yourself the basic Http authentication scheme =

1/ install "Java JDK"
2/ install IDE : "IntelliJ IDEA"
3/ setup your app, using spring "initializr" click+download
4/ run your app
5/ open http://localhost:8080
check app is a secured http webapp, with login/password dialog
6/ inspect network requests
7/ test Http requests

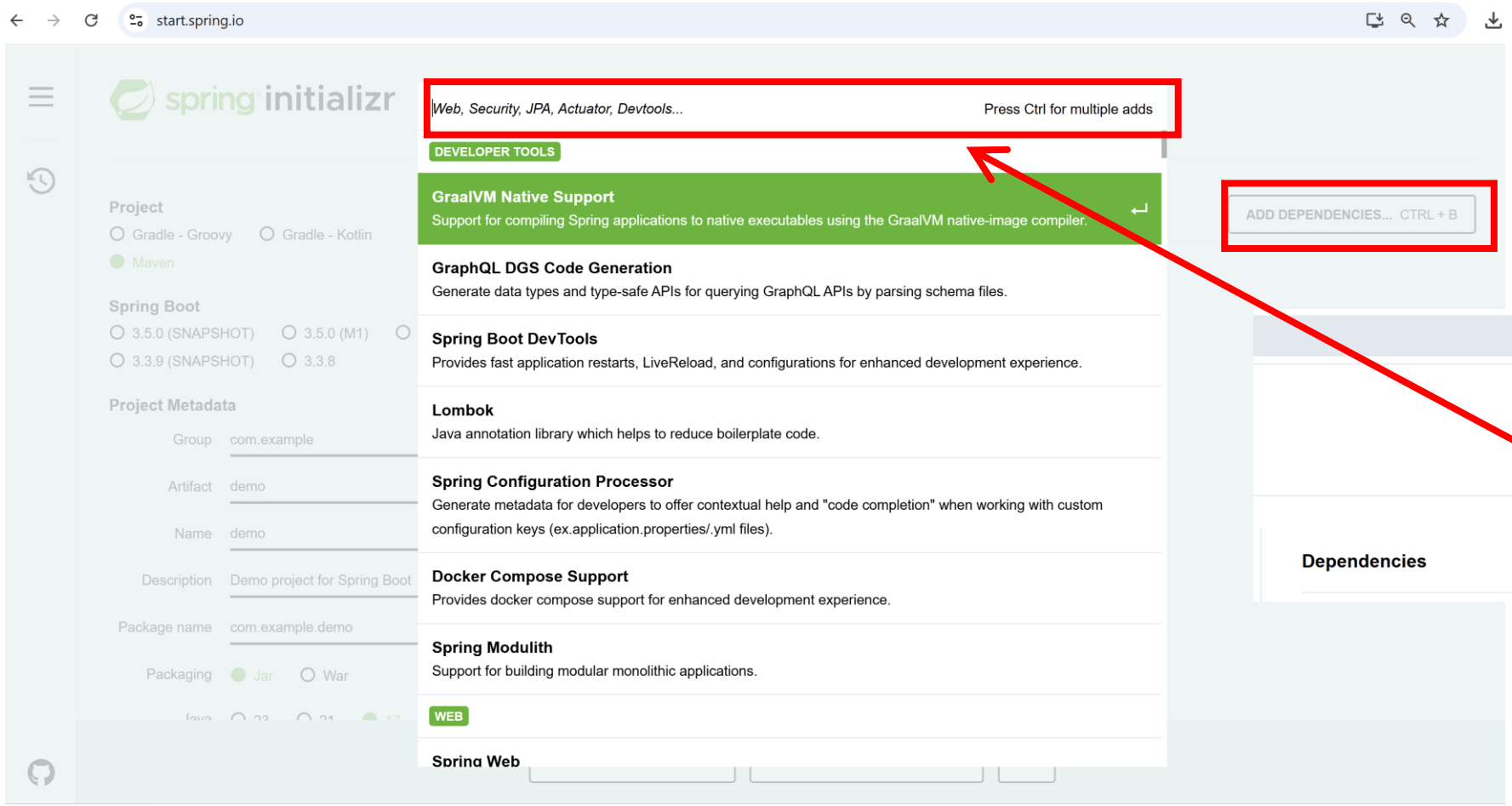
Using springboot starter aka "Spring Initializr" https://start.spring.io

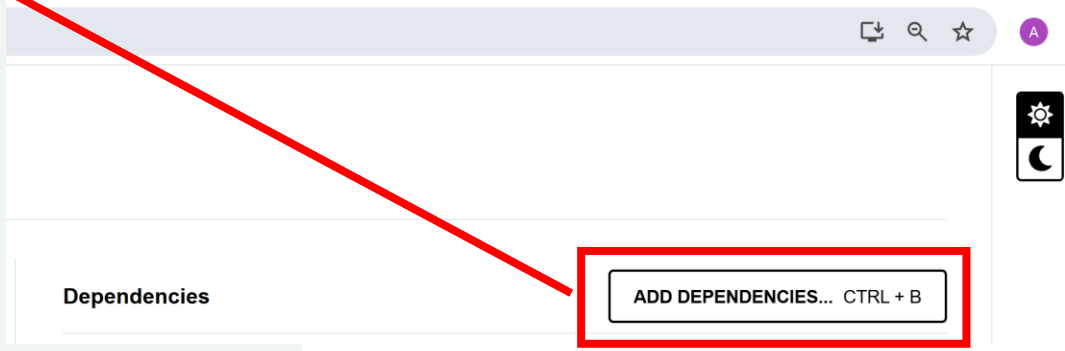


Open Springboot Initializr https://start.spring.io

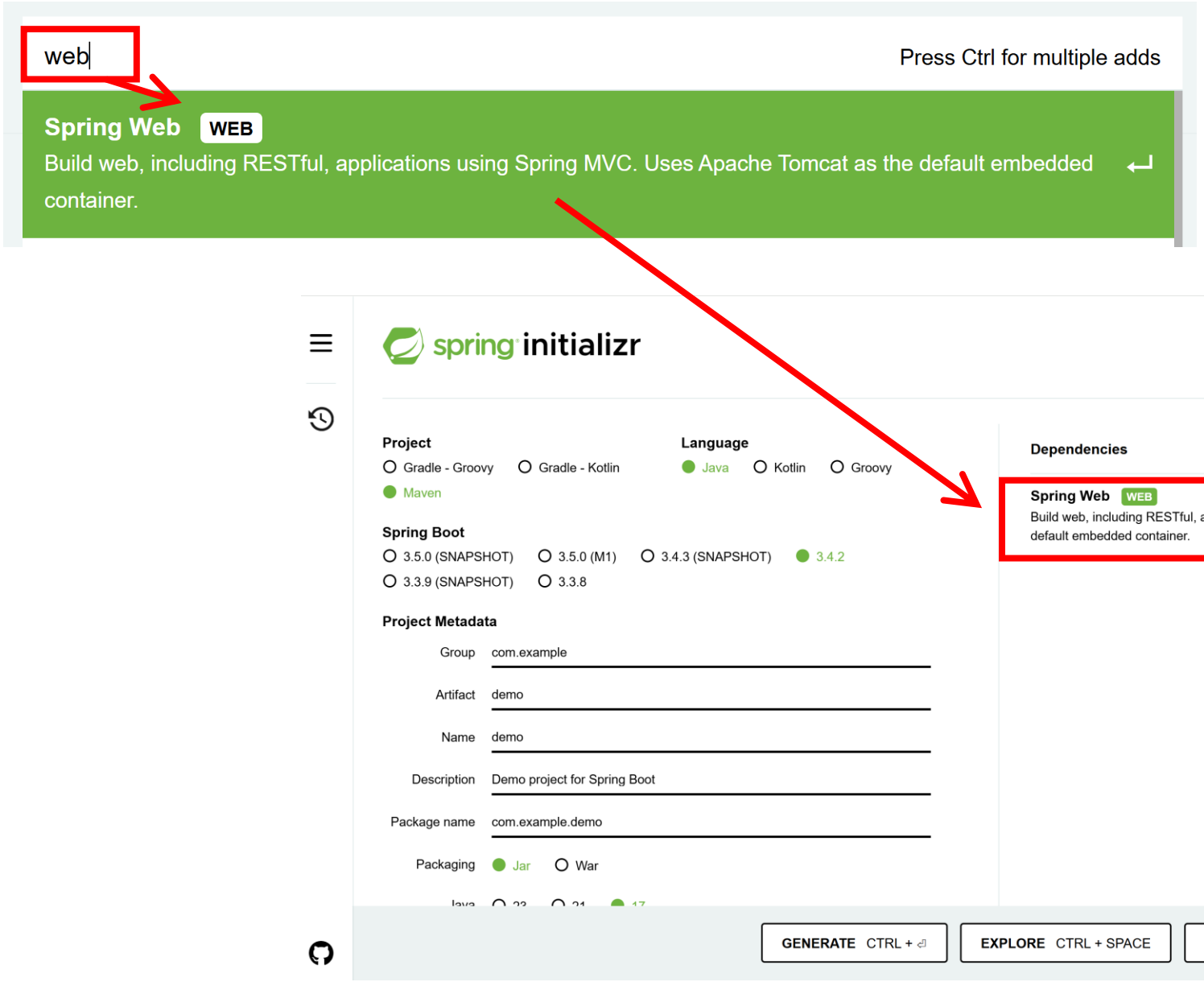


Click "ADD DEPENDENCIES..."

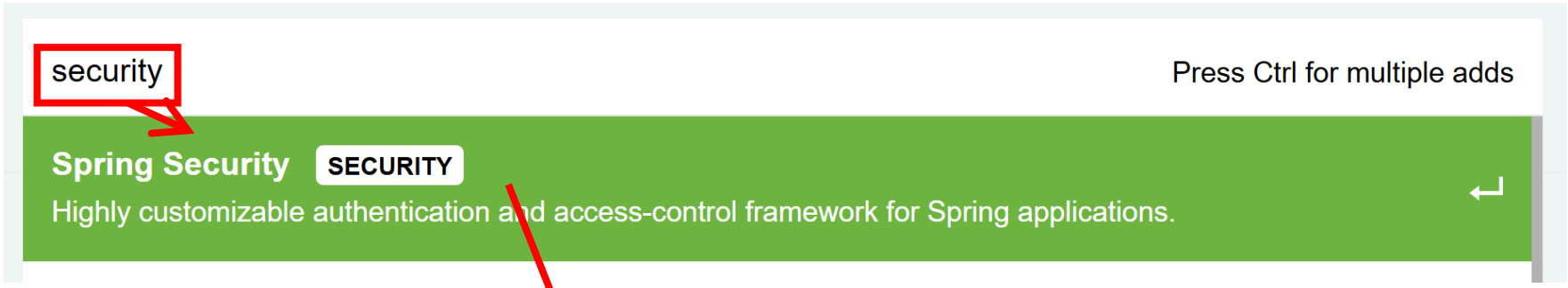


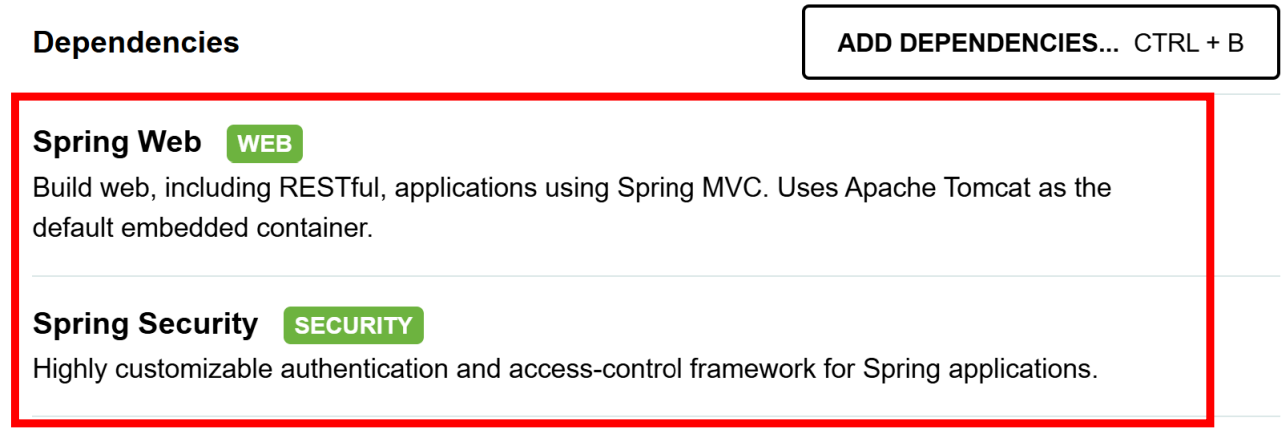


Add "web"

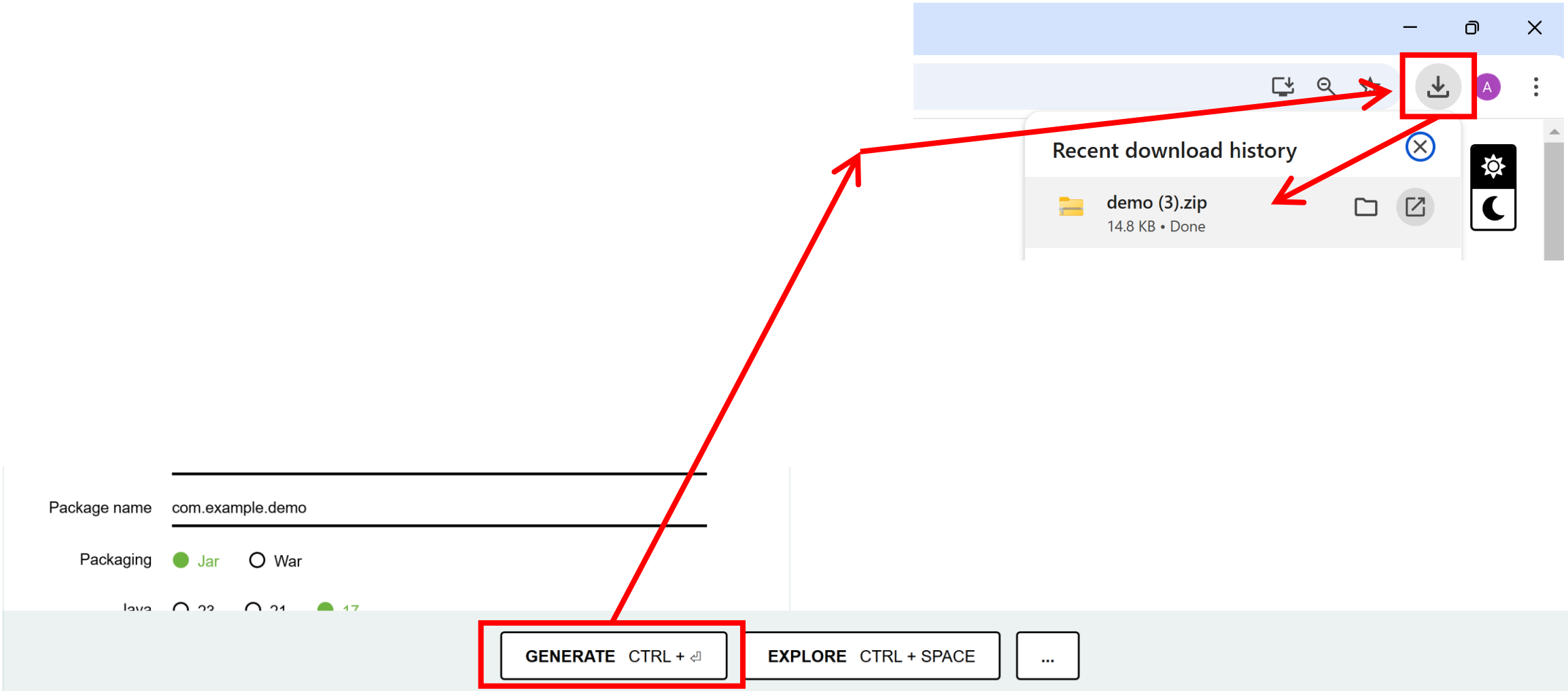

web
Press Ctrl for multiple adds
Spring Web
WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.
spring initializr
Project
Gradle - Groovy
Gradle - Kotlin
Maven
Language
Java
Kotlin
Groovy
Spring Boot
3.5.0 (SNAPSHOT)
3.5.0 (M1)
3.4.3 (SNAPSHOT)
3.4.2
3.3.9 (SNAPSHOT)
3.3.8
Project Metadata
Group
com.example
Artifact
demo
Name
demo
Description
Demo project for Spring Boot
Package name
com.example.demo
Packaging
Jar
War
Dependencies
ADD DEPENDENCIES... CTRL + B
Spring Web
WEB
Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.
GENERATE CTRL + ⏎
EXPLORE CTRL + SPACE
...

Then add "security"

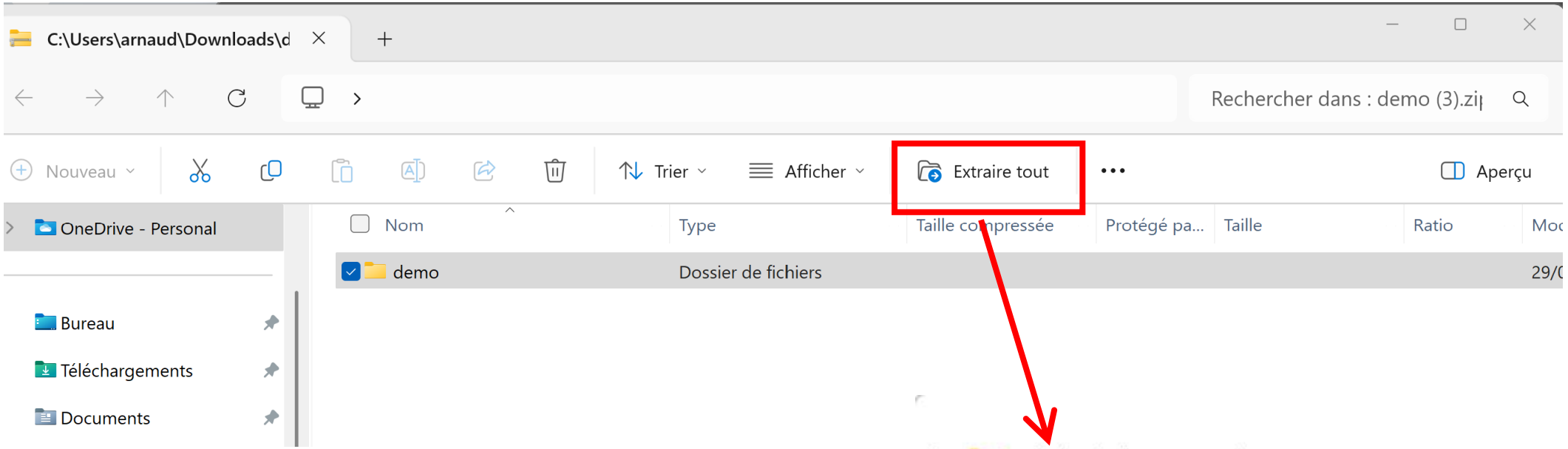


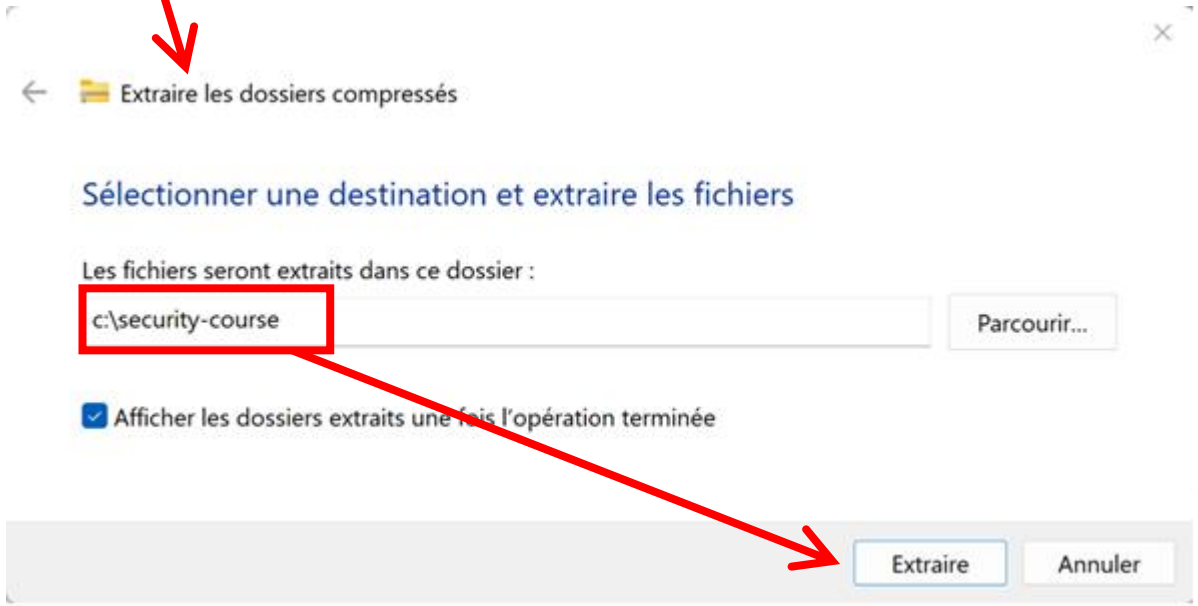


Then Click "GENERATE", open downloaded zip

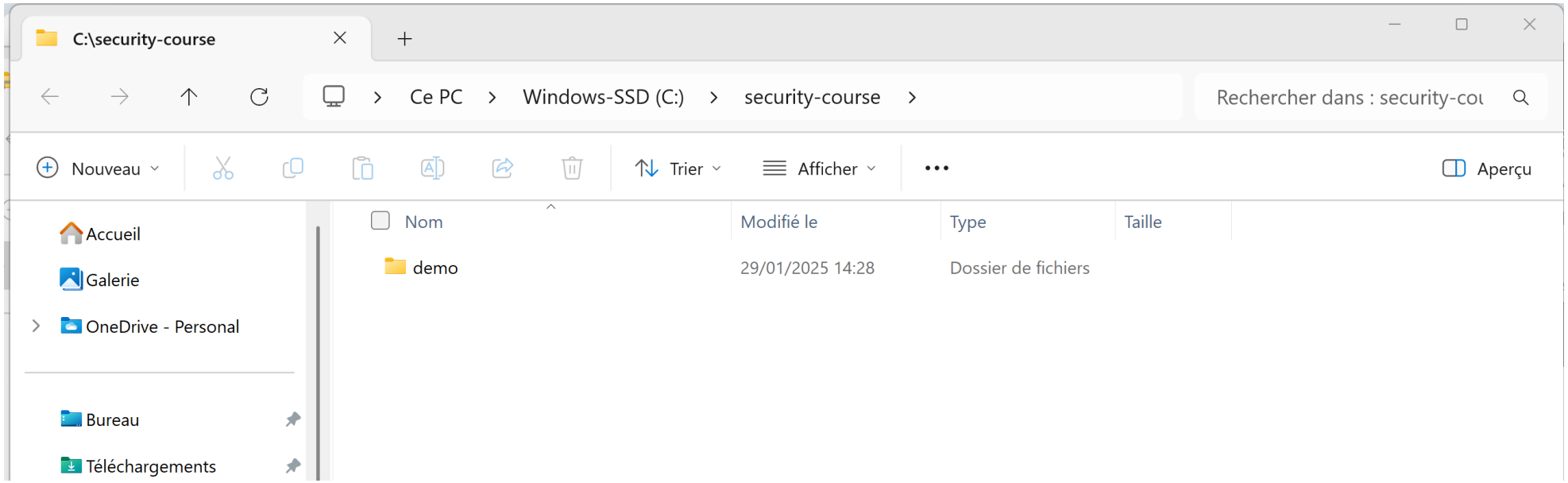


Extract All in a local dir (example in c:\security-course)





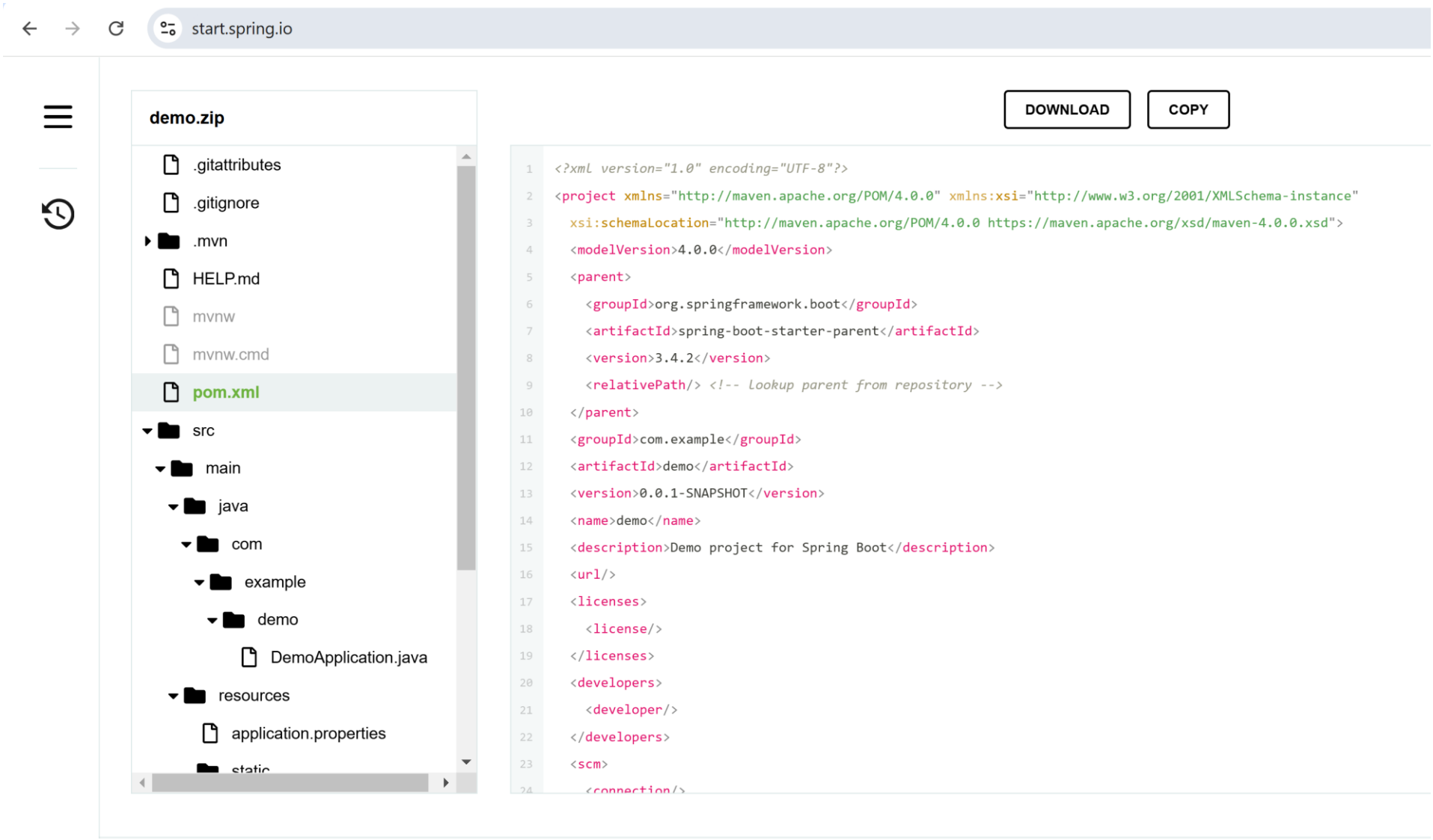
Explore Dir & Files



Basically 2 interresting files:

"pom.xml" => maven description of the java project
"src/main/java/com/example/demo/DemoApplication.java" => the main program (= the web app)

Explore Dir & Files from initializr "EXPLORE"



pom.xml contains basic information
+ 2 springboot dependencies: security, web

pom.xml (demo)

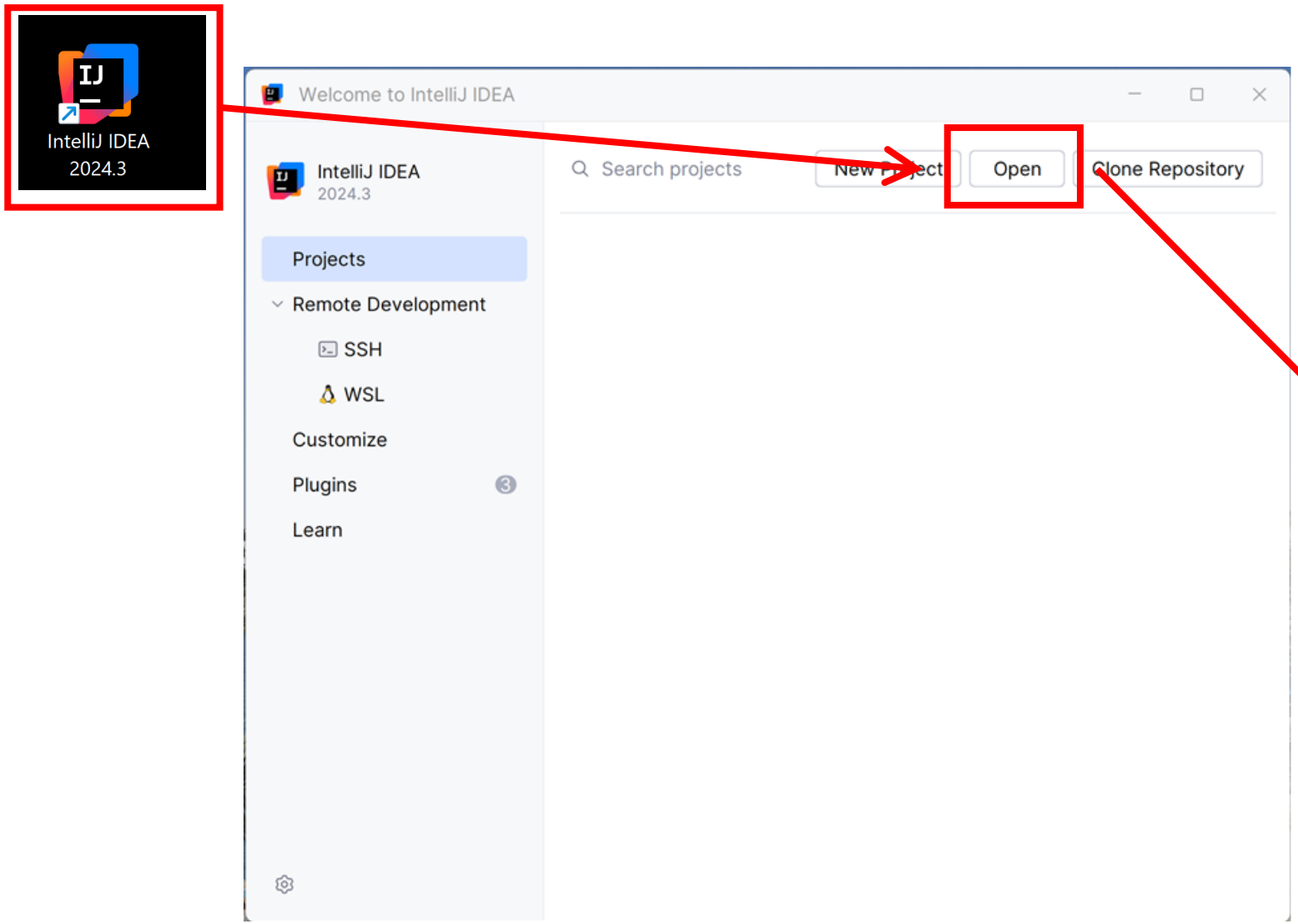
```
2   <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-inst
23      <scm>
28      </scm>
29      <properties>
30          <java.version>17</java.version>
31      </properties>
32      <dependencies> Add Starters...
33          <dependency>
34              <groupId>org.springframework.boot</groupId>
35              <artifactId>spring-boot-starter-security</artifactId>
36          </dependency>
37          <dependency>
38              <groupId>org.springframework.boot</groupId>
39              <artifactId>spring-boot-starter-web</artifactId>
40          </dependency>
41
42          <dependency>
```

TD Outline

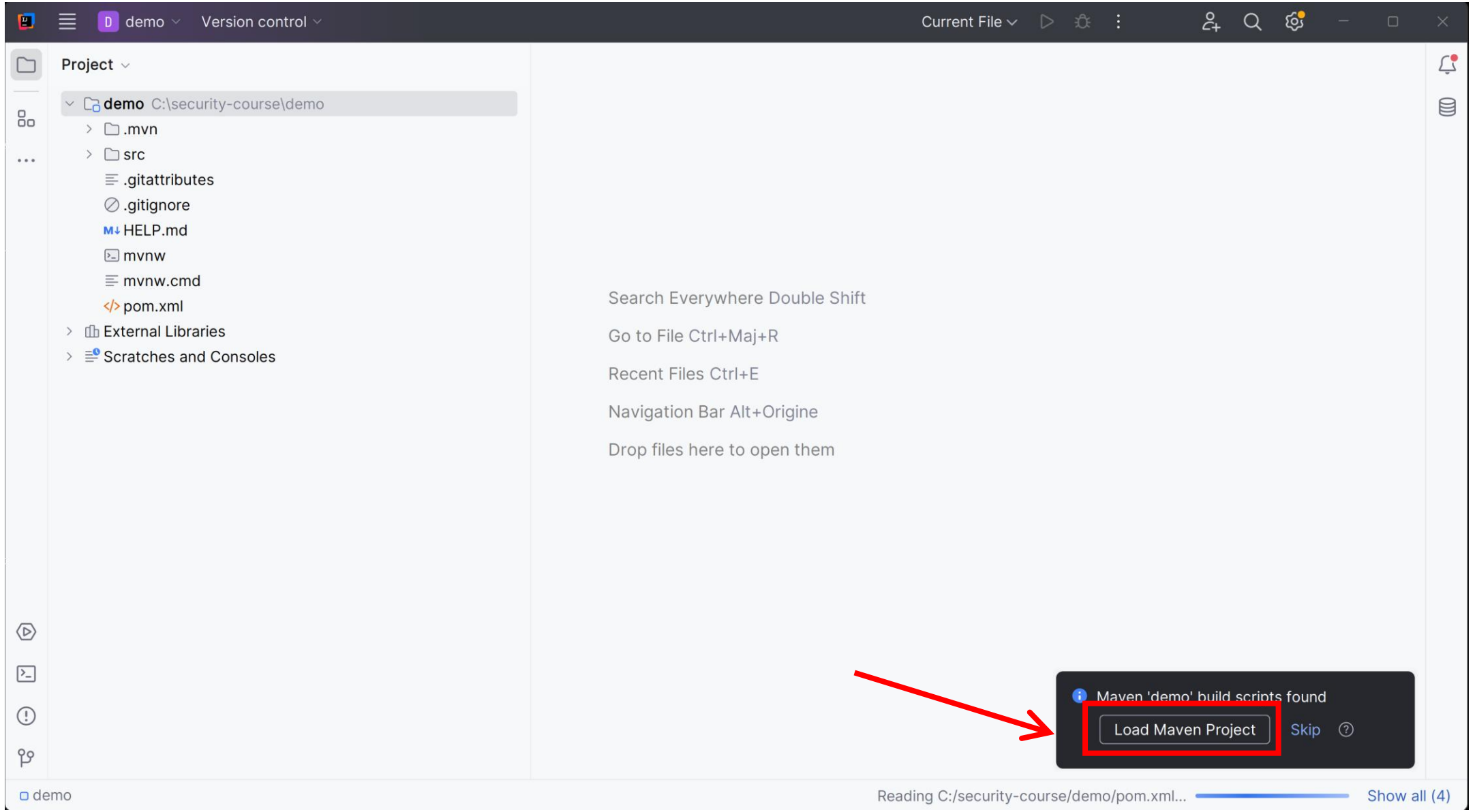
TD Goals = debug yourself the basic Http authentication scheme =

1/ install "Java JDK"
2/ install IDE : "IntelliJ IDEA"
3/ setup your app, using spring "initializr" click+download
4/ run your app
5/ open http://localhost:8080
  check app is a secured http webapp, with login/password dialog
6/ inspect network requests
7/ test Http requests

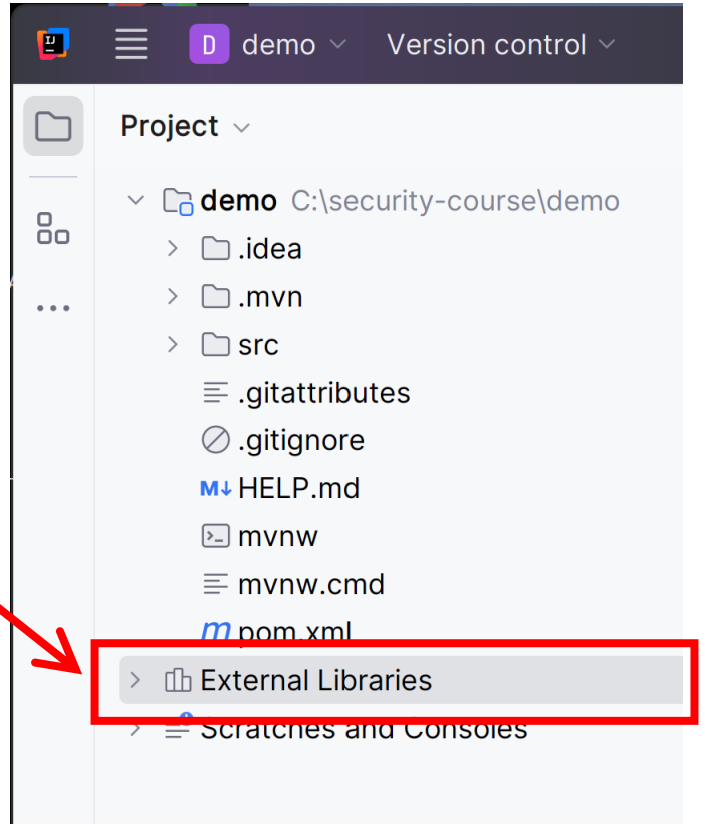
Launch your IDEA IntelliJ Open Project



Click default "Load Maven Project"

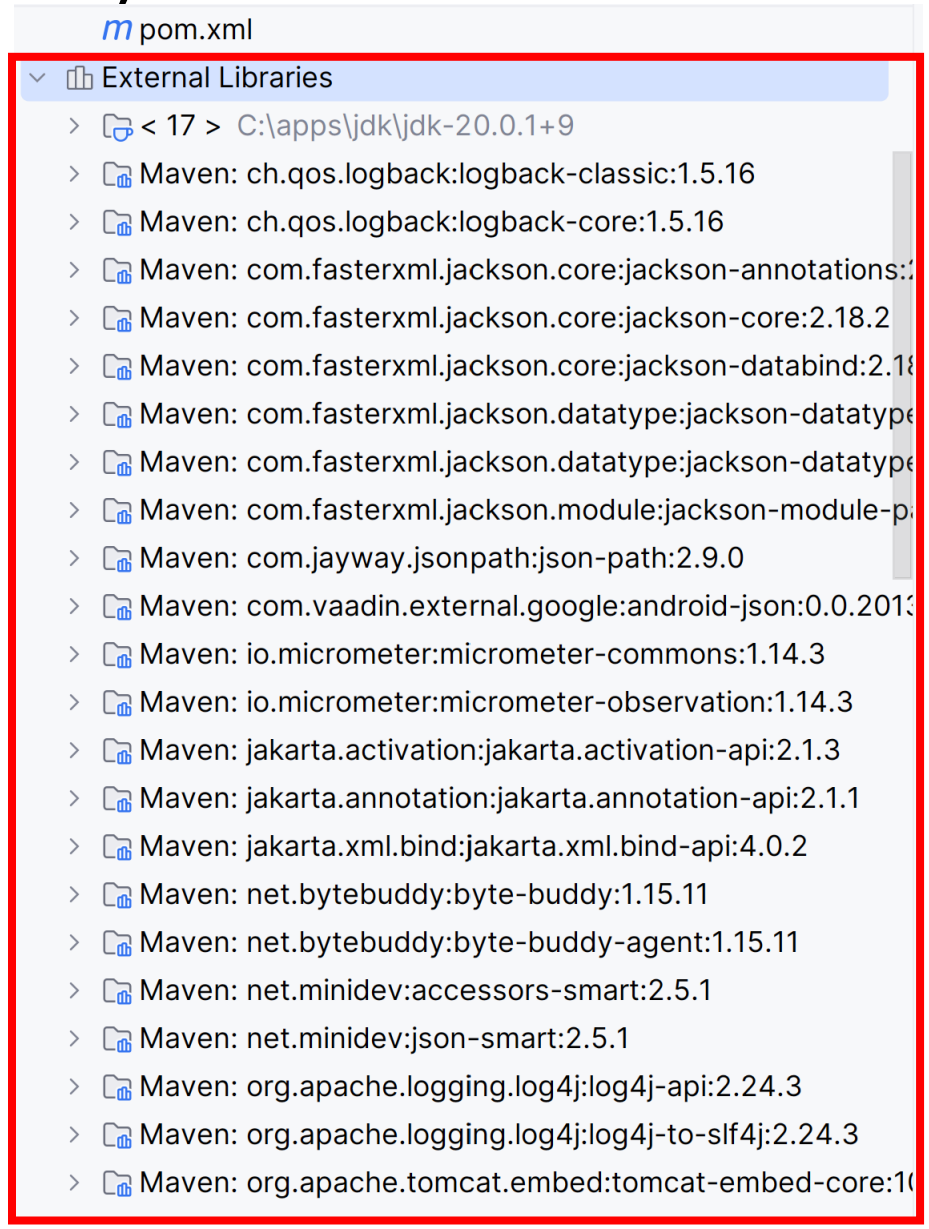


Wait few minutes (downloading dependencies ...)

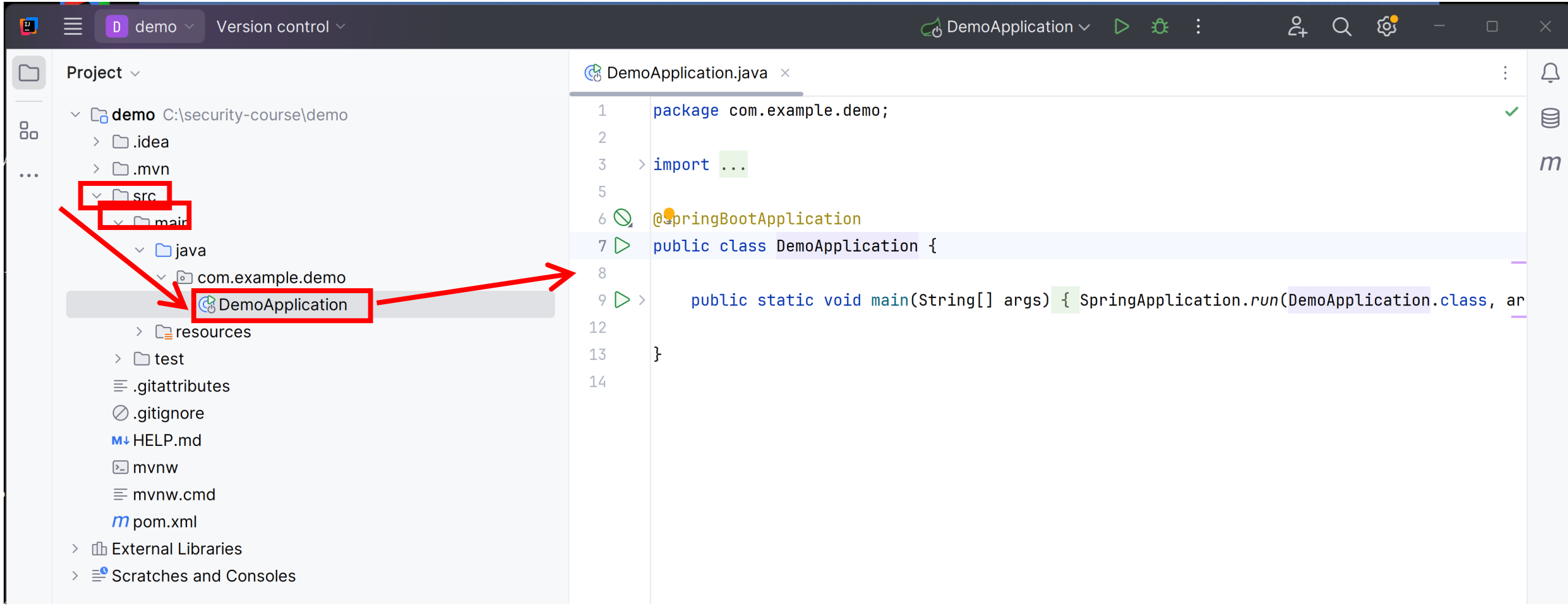


when finished
All dependencies
are automatically
downloaded

spring -> tomcat -> ...

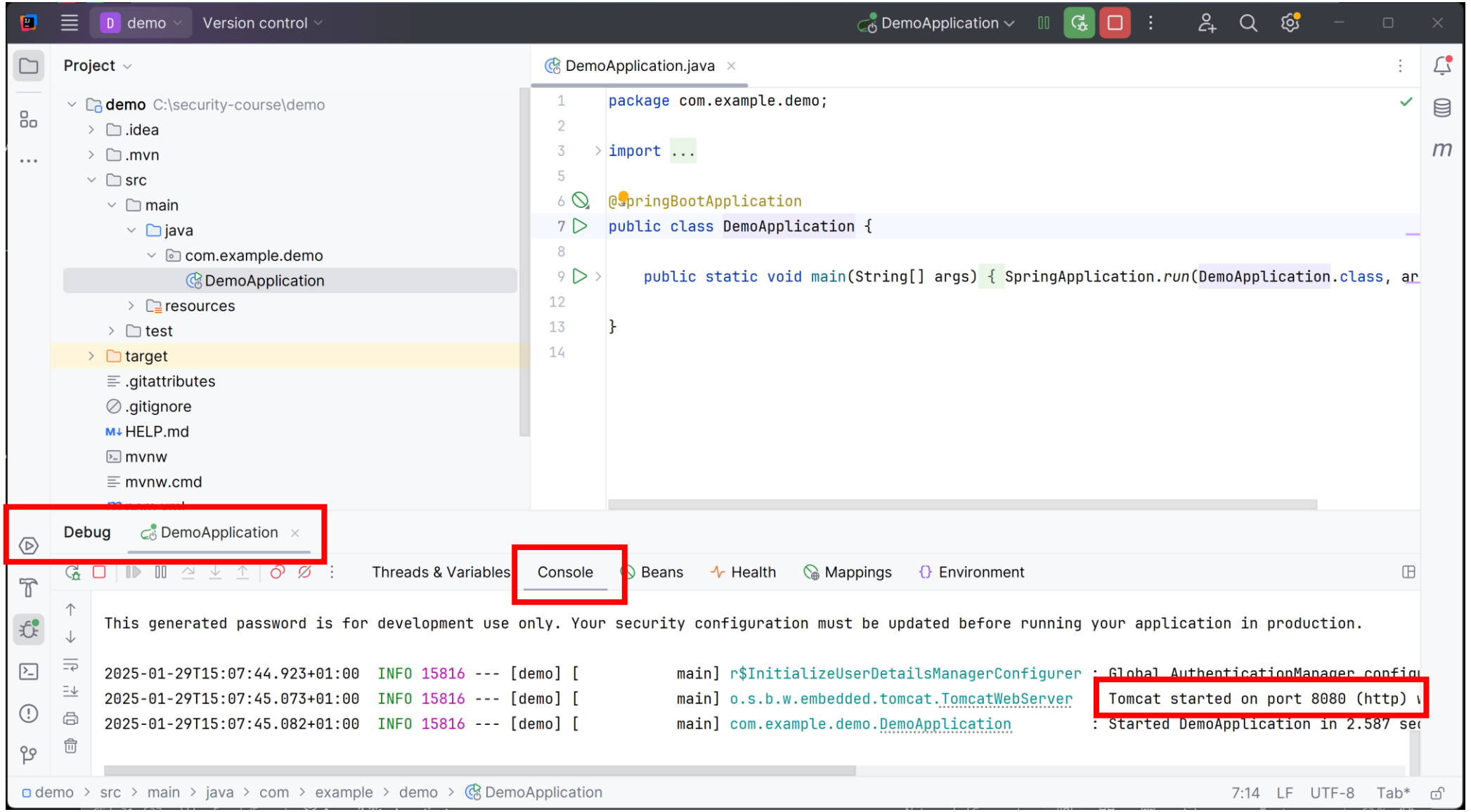


Open the main class "DemoApplication" in src/main/java/com.example.demo/

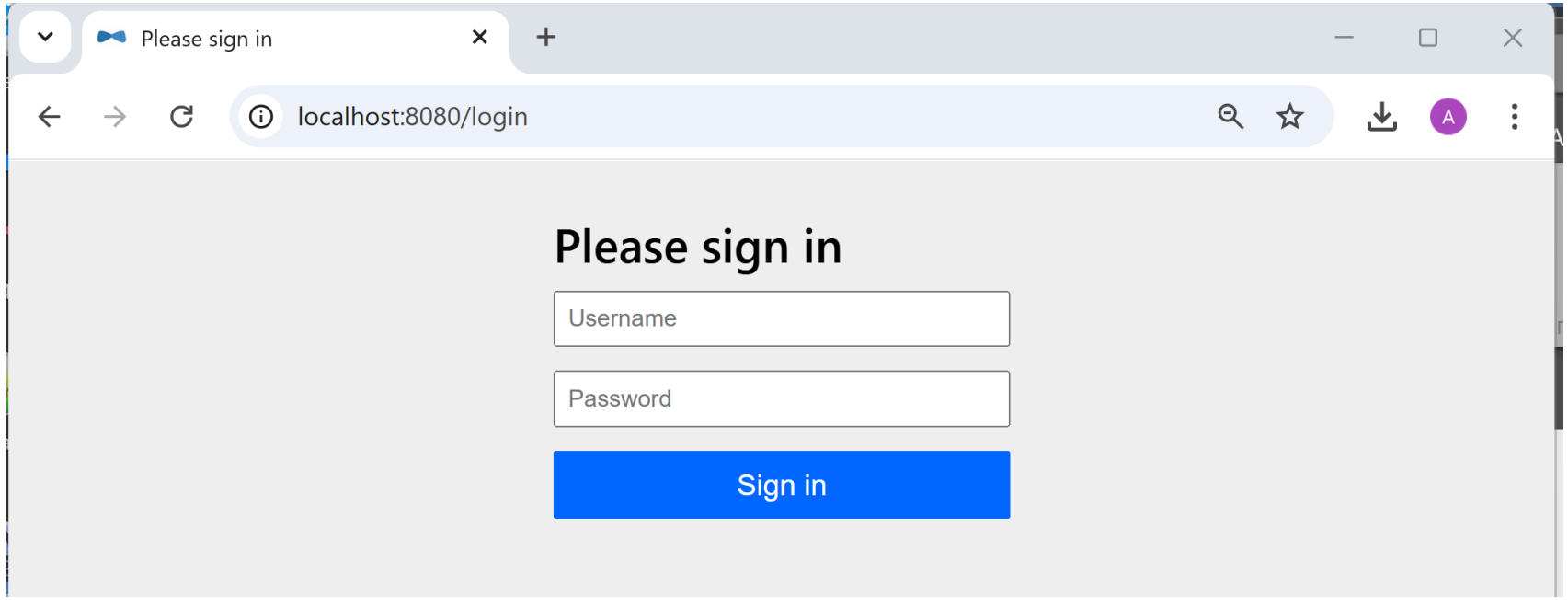


Click on "Run" Debug button

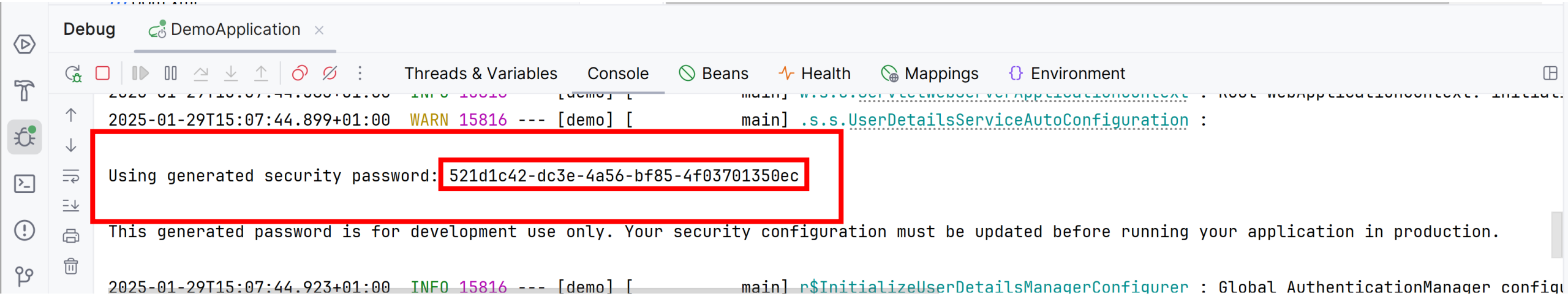
See Debug "Console" logs...
Tomcat started on port 8080 (http)



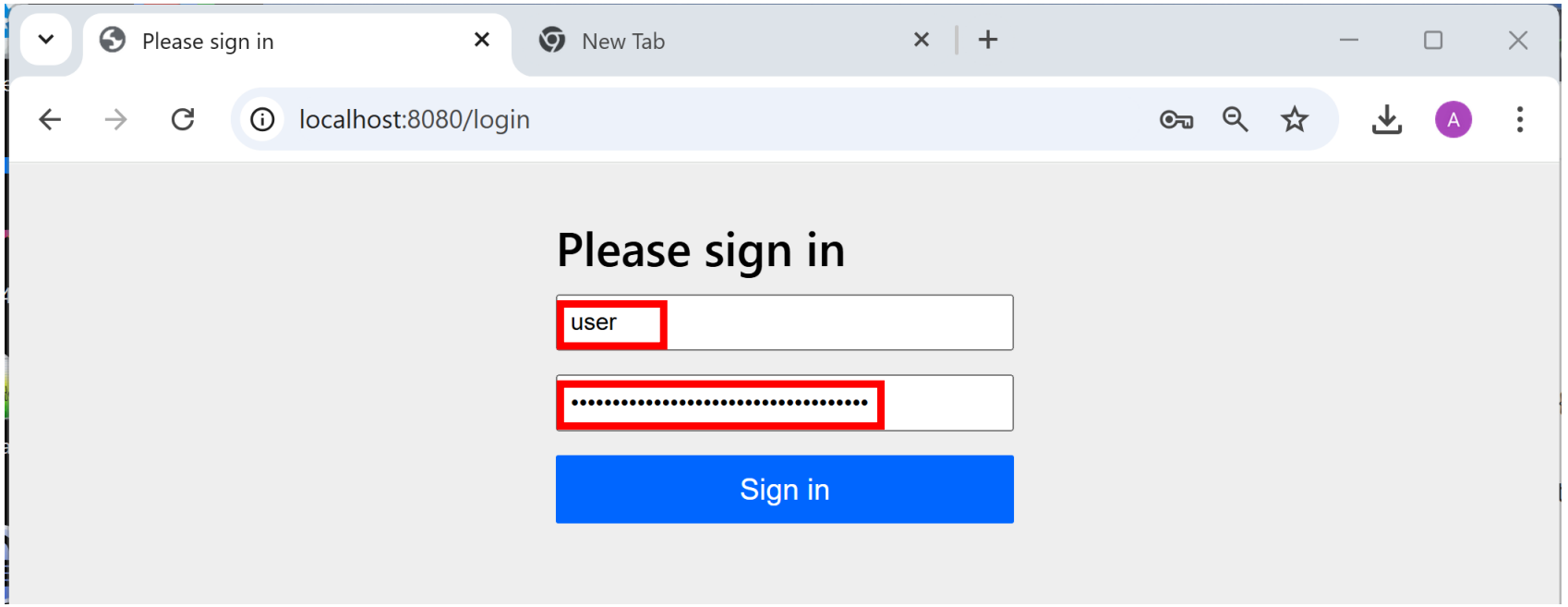
Open your Web browser (Chrome) http://localhost:8080



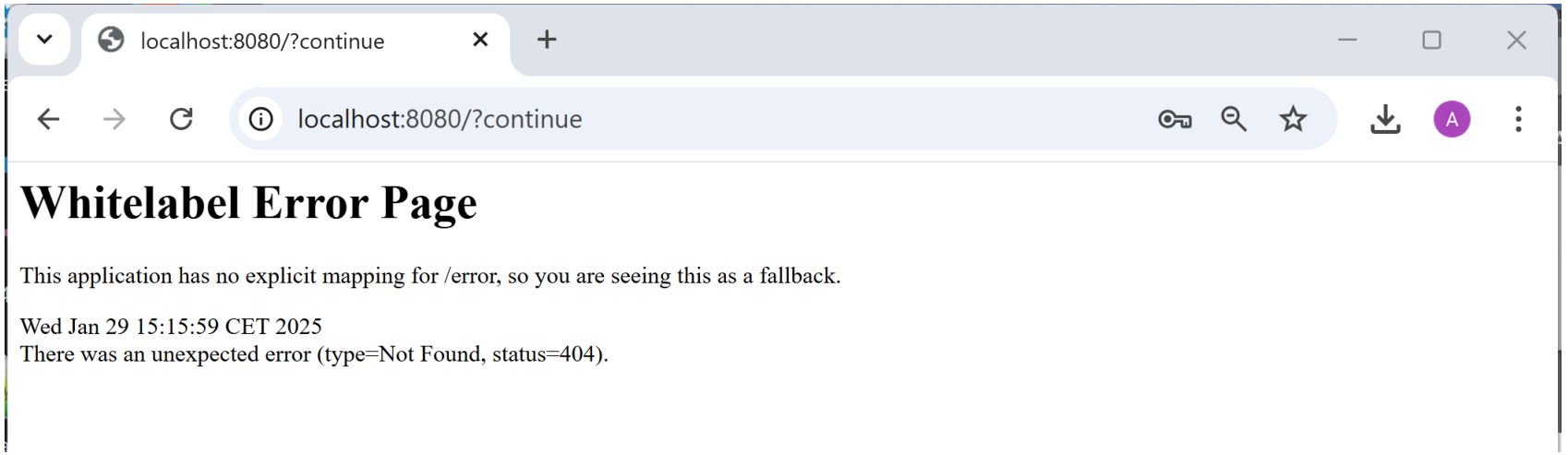
See "user" / generated password in console (none configured yet by default)



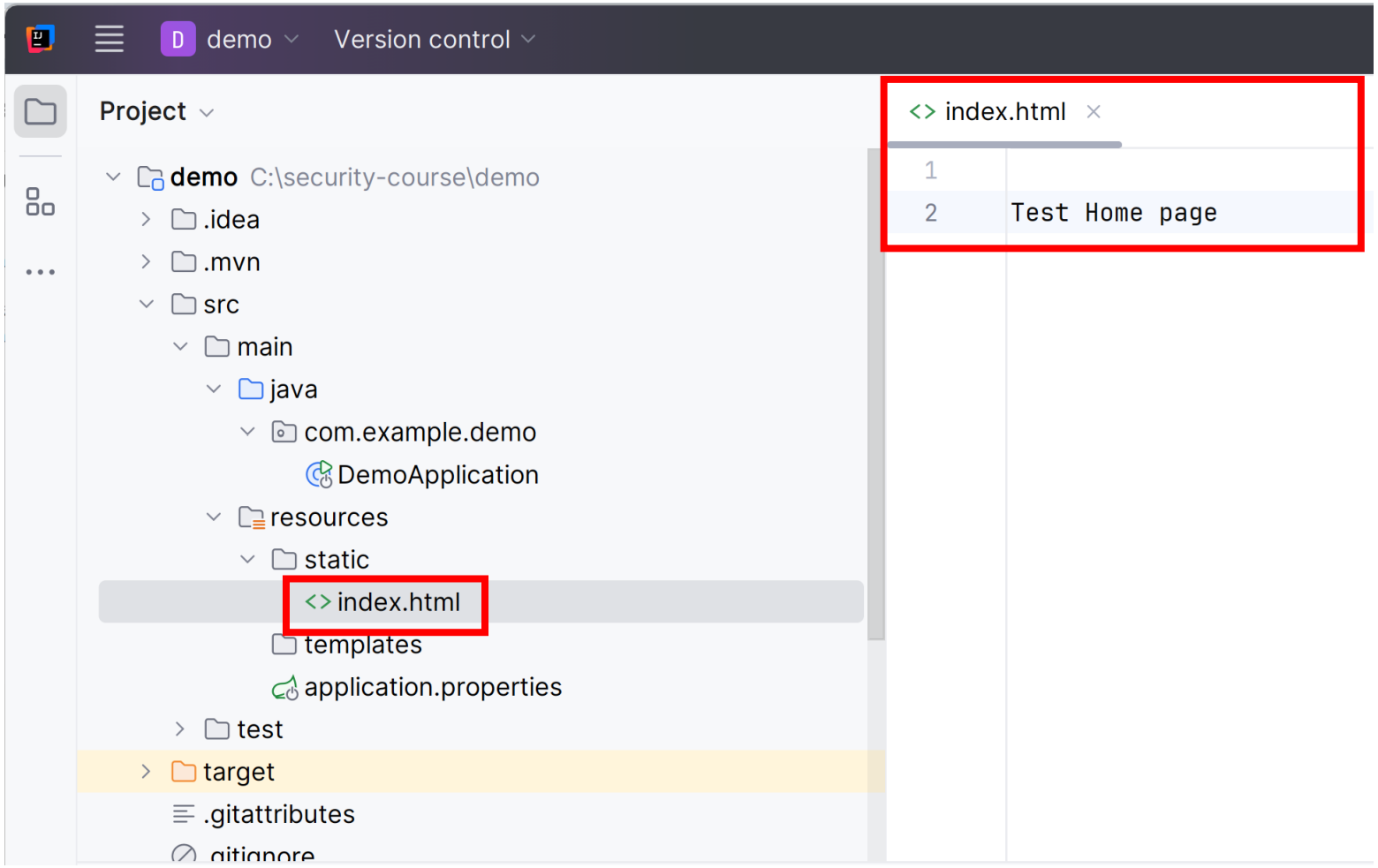
Testing user / generated password



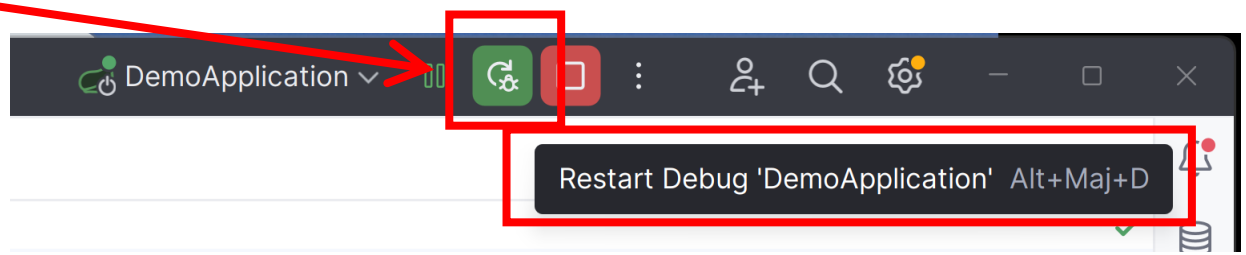
no default page exists yet... = > 404
This is OK (see next slide for "index.html")

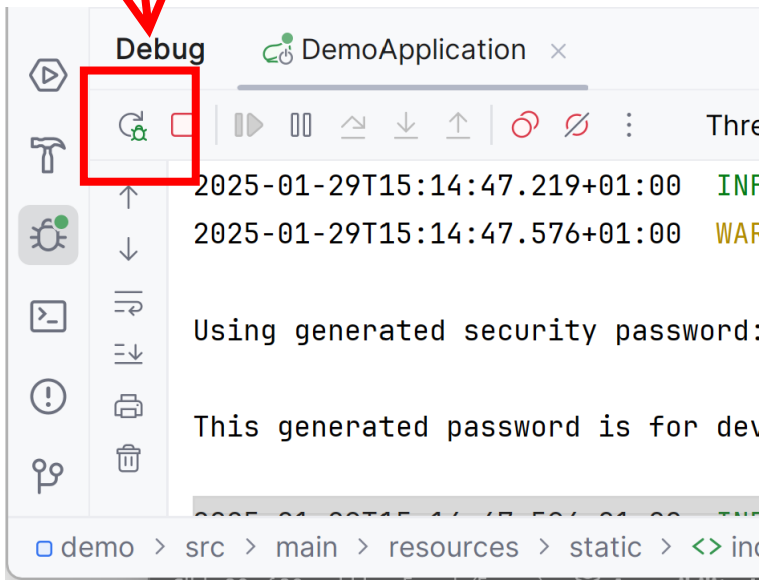


Adding new file "src/main/resource/static/index.html"



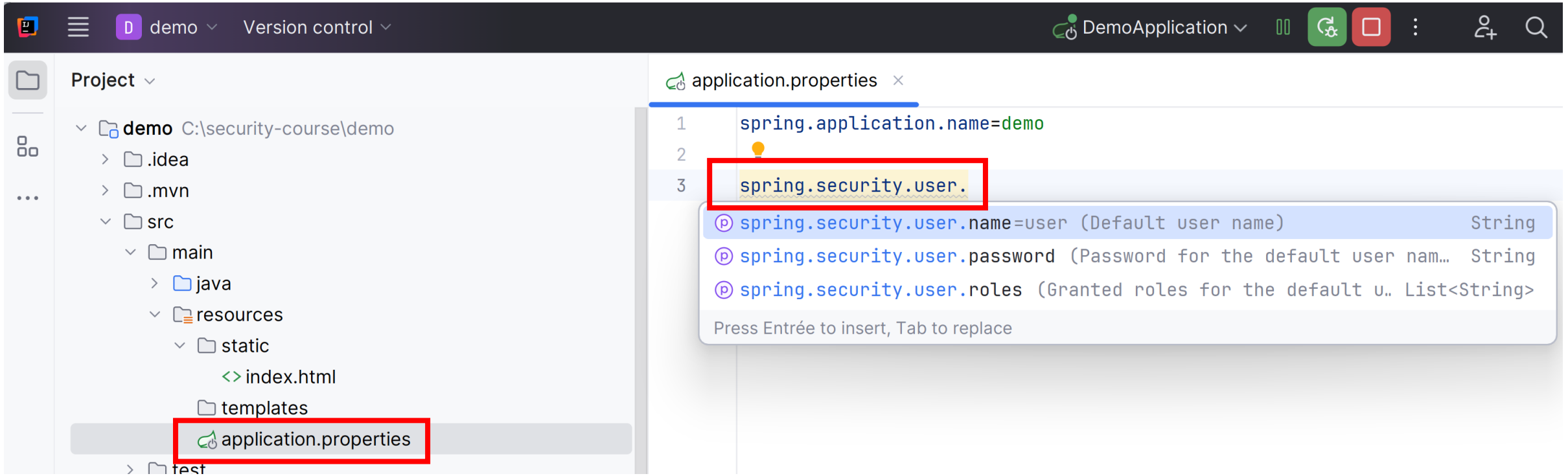
Relaunch


DemoApplication
Restart Debug 'DemoApplication' Alt+Maj+D


Debug
DemoApplication
2025-01-29T15:14:47.219+01:00
2025-01-29T15:14:47.576+01:00
Using generated security password:
This generated password is for dev
demo
src
main
resources
static

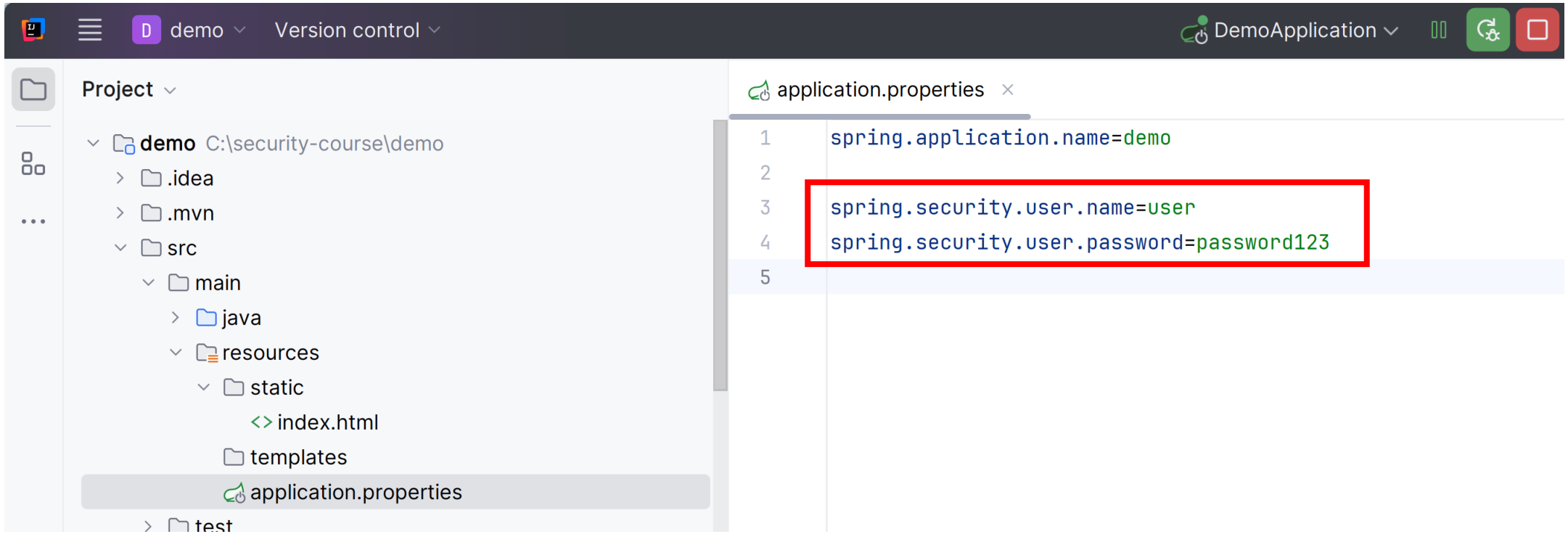
Retry ...
Resfresh (F5) Web page
login user / (new) generated password

Edit file src/main/resources/application.properties to force user - password

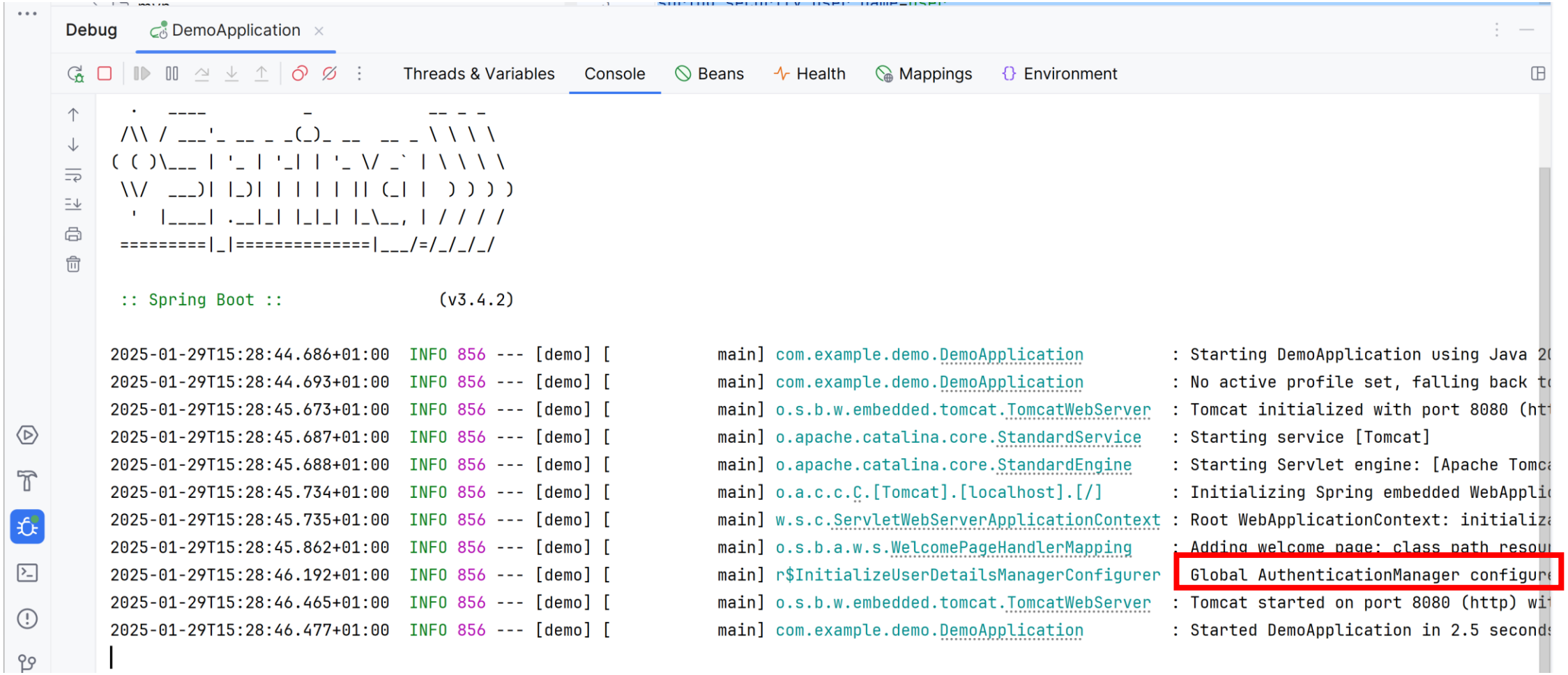


Edit file, example

```
spring.security.user.name=user
spring.security.user.password=password123    # not for production...
```



Relaunch + Refresh web page
Notice … no more generated password



TD Outline

TD Goals = debug yourself the basic Http authentication scheme =

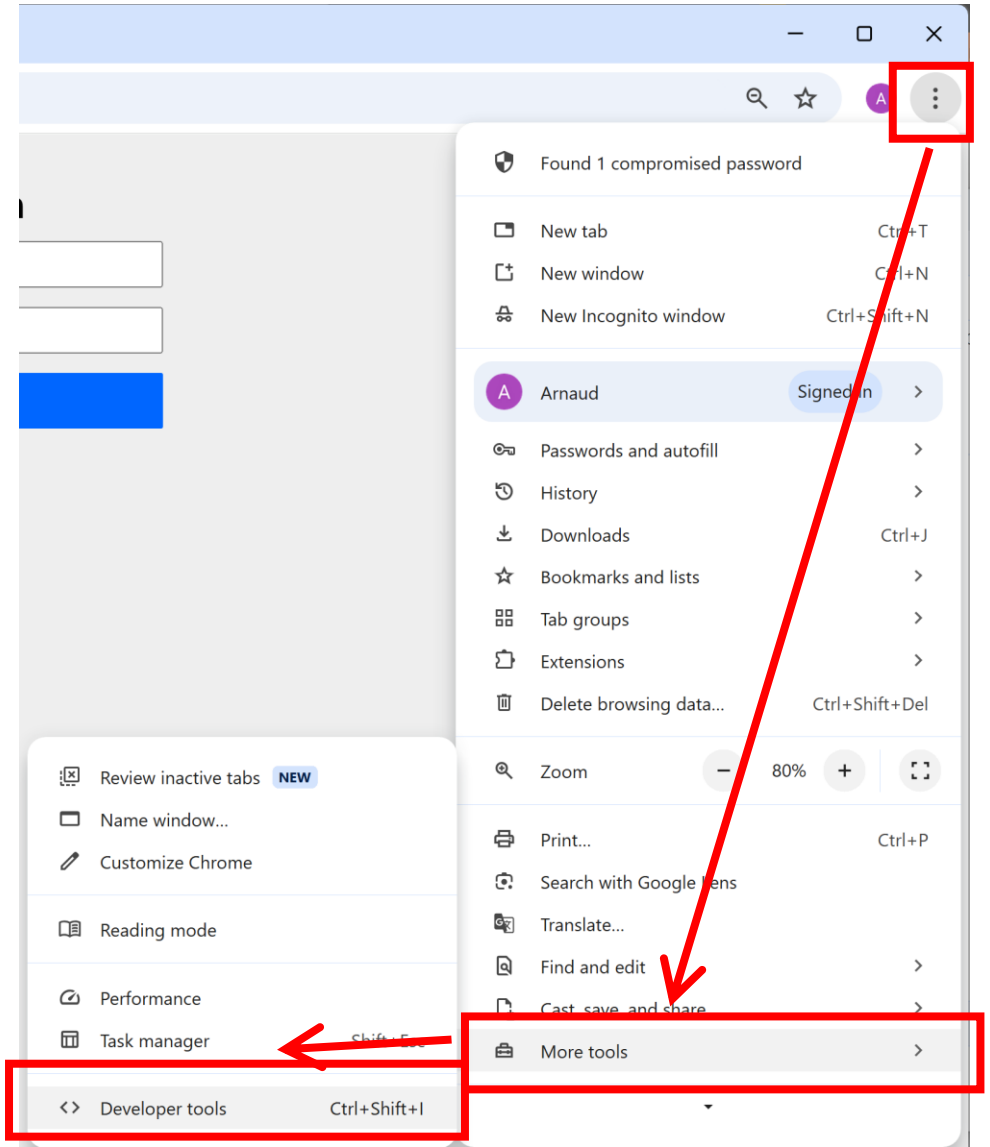
1/ install "Java JDK"
2/ install IDE : "IntelliJ IDEA"
3/ setup your app, using spring "initializr" click+download
4/ run your app
5/ open http://localhost:8080
  check app is a secured http webapp, with login/password dialog
6/ inspect network requests
7/ test Http requests

Open Chrome "DevTools"

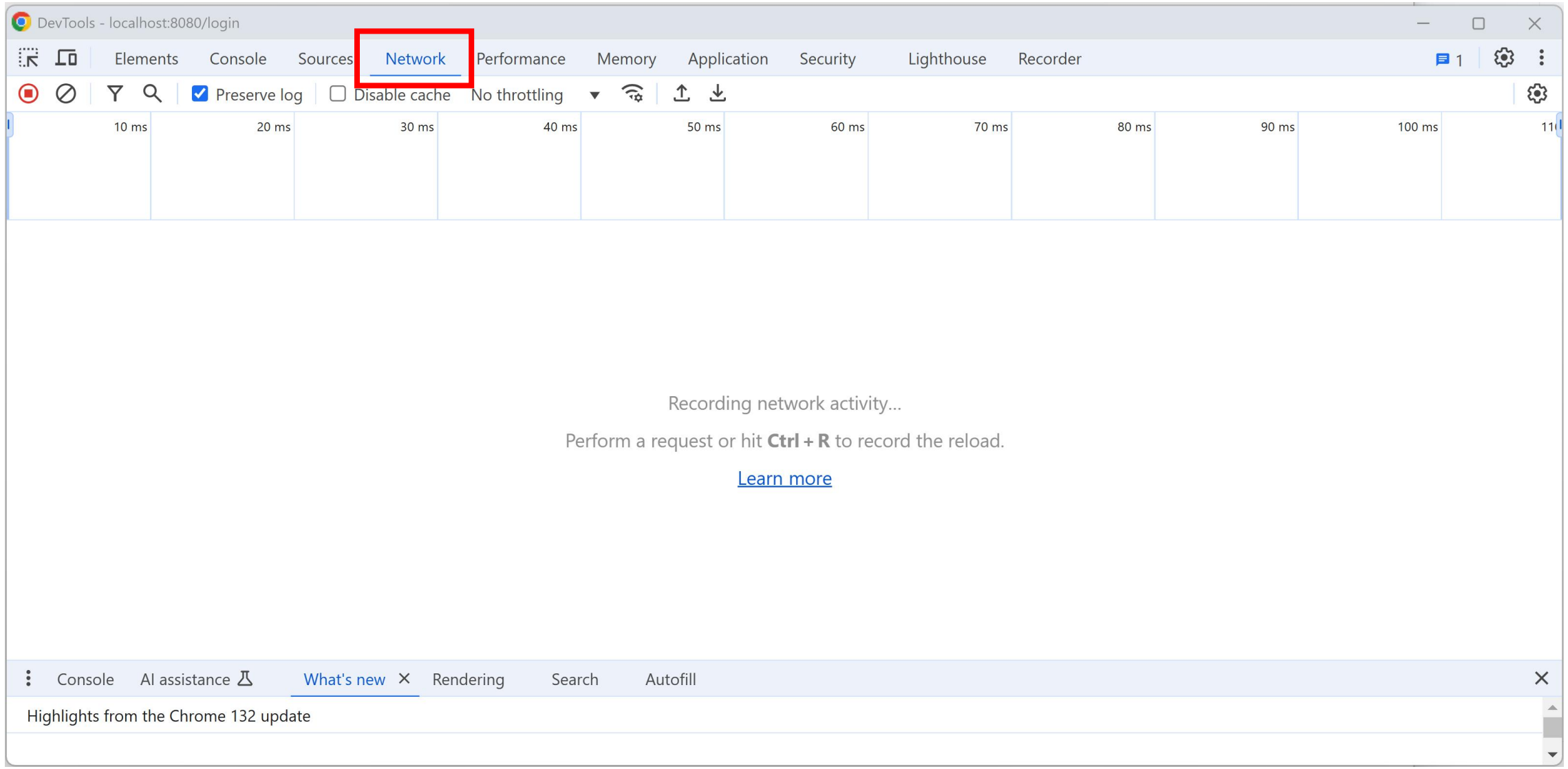
Menu > More tools > Developer tools

or F12

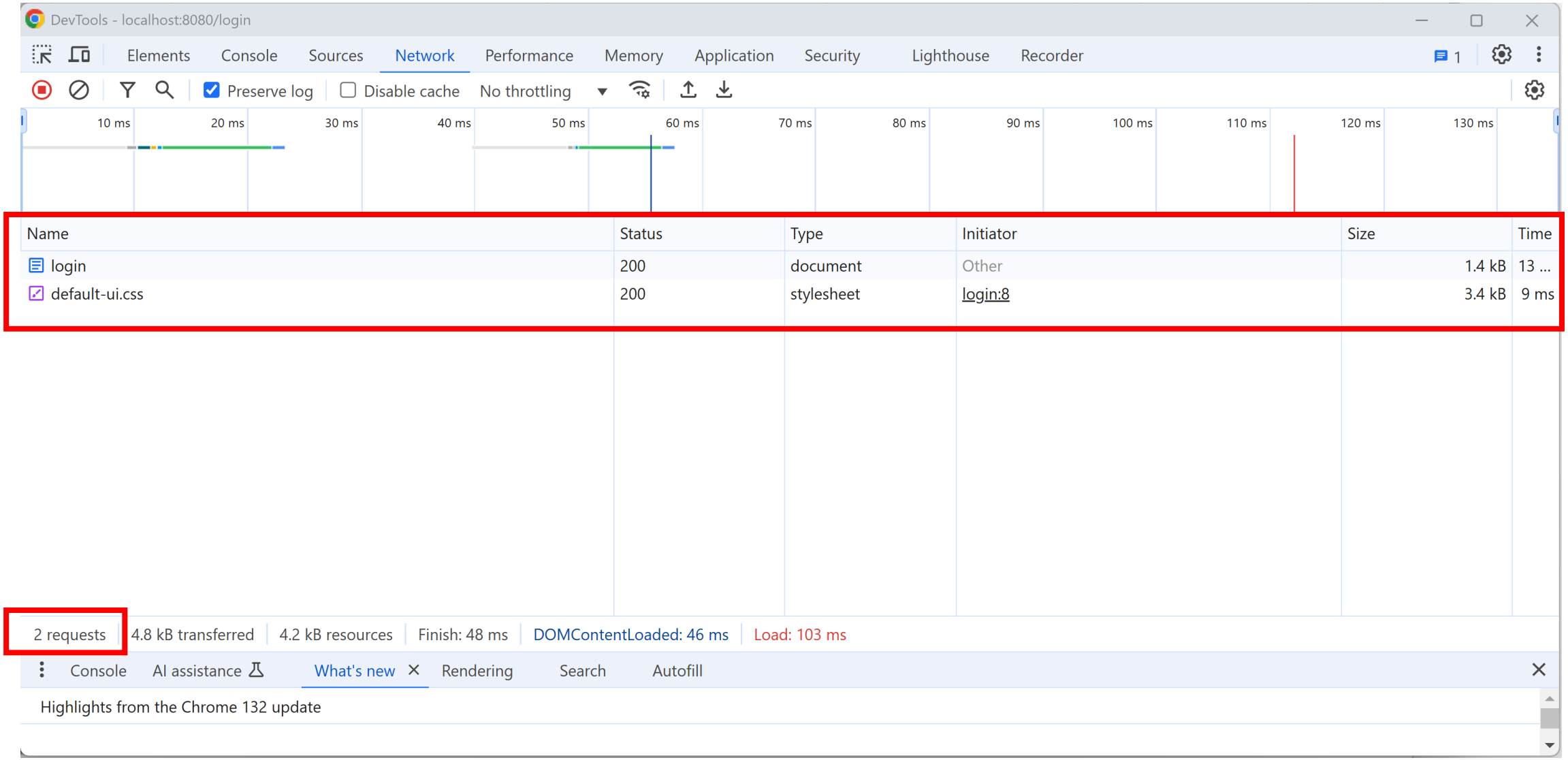
or Ctrl+Shift+I



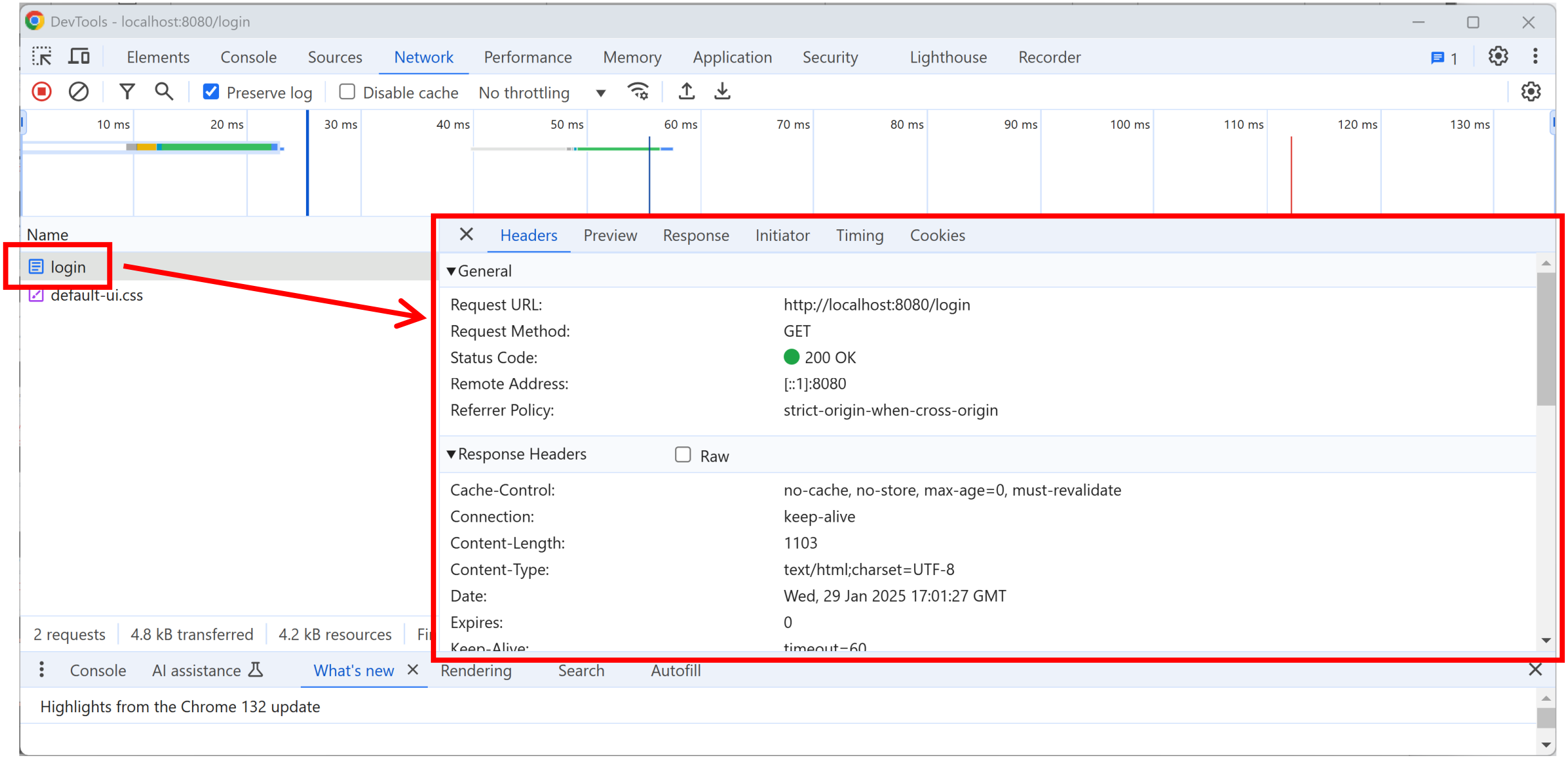
DevTool > tab "Network"



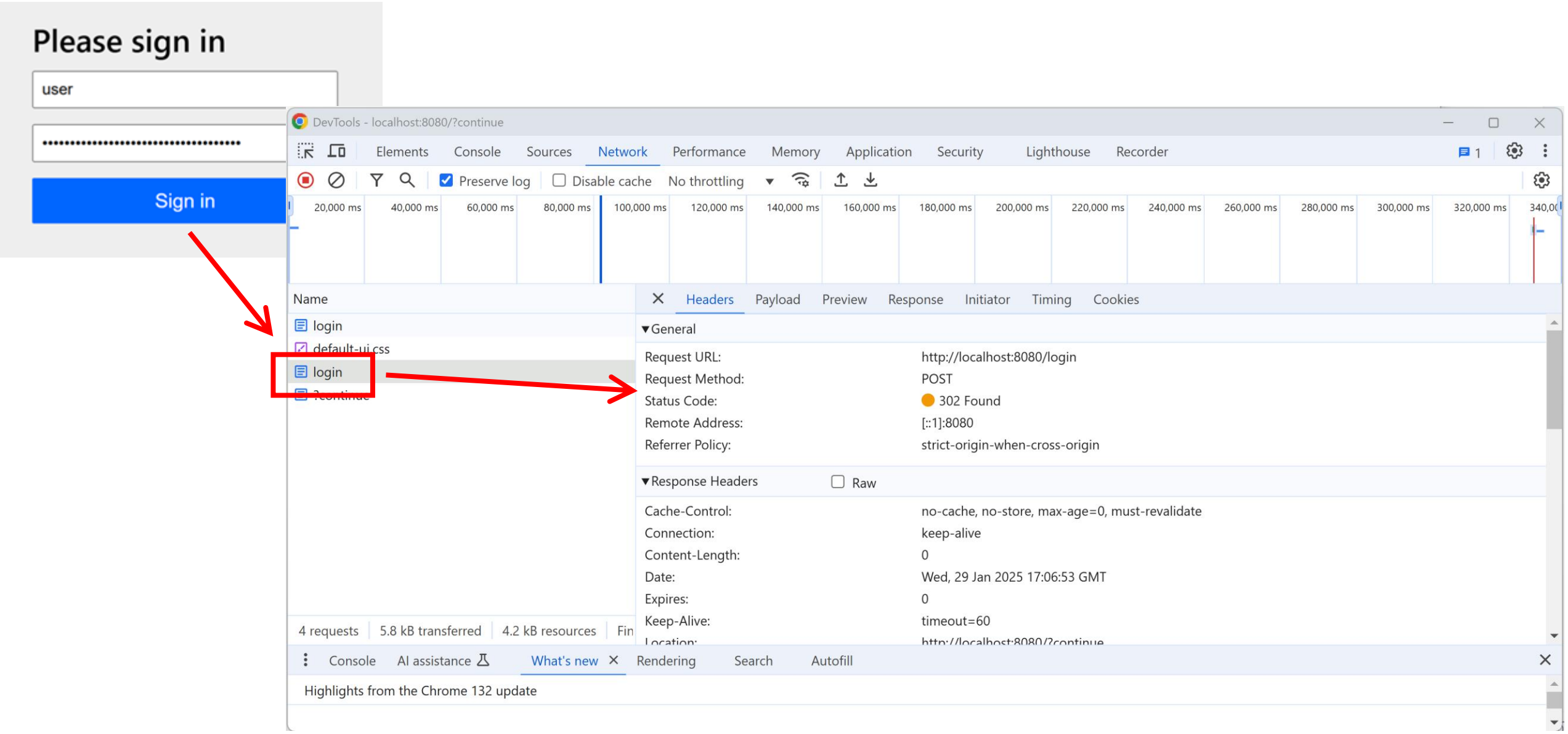
Reload Html page (F5 or Ctrl+R)
see 2 requests



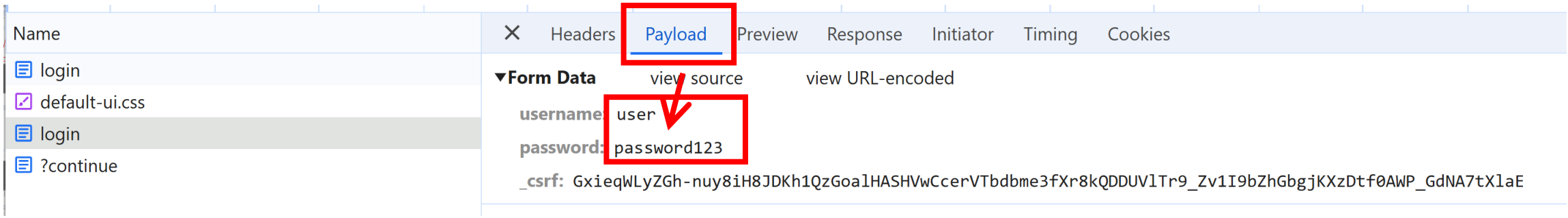
Detail Request 1 "login"



login user / password => see Http request 3 "login"



see the Payload user / password in clear text passed to Http POST

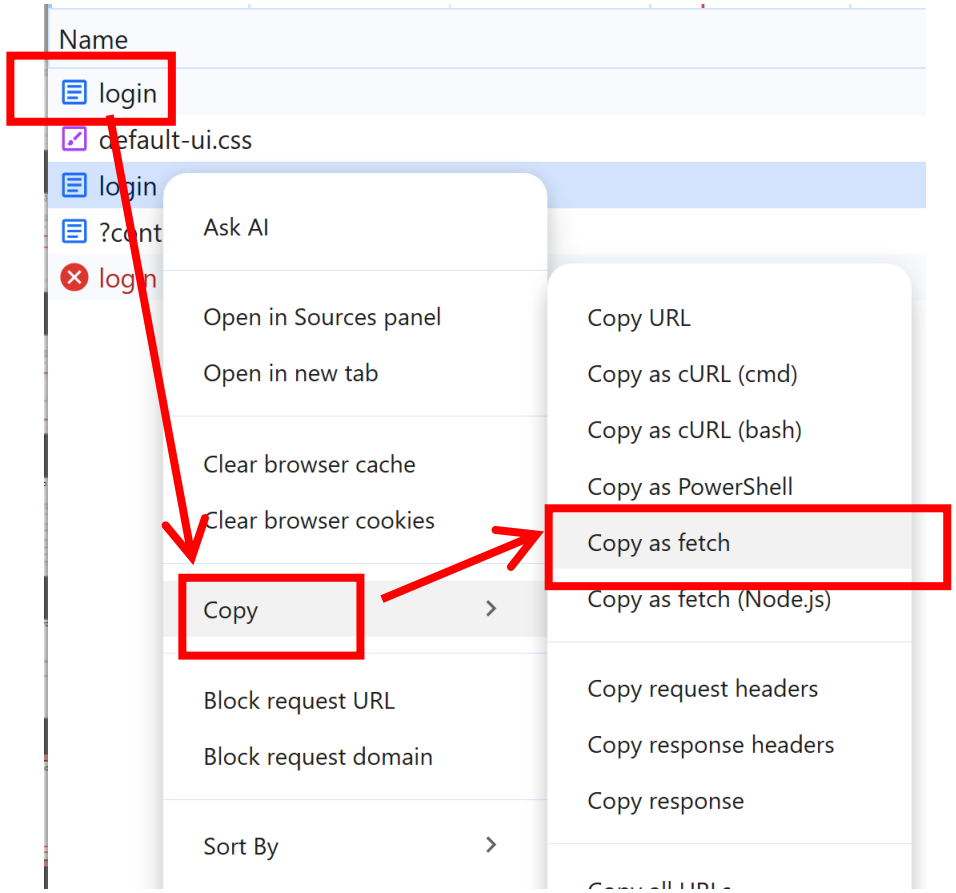


TD Outline

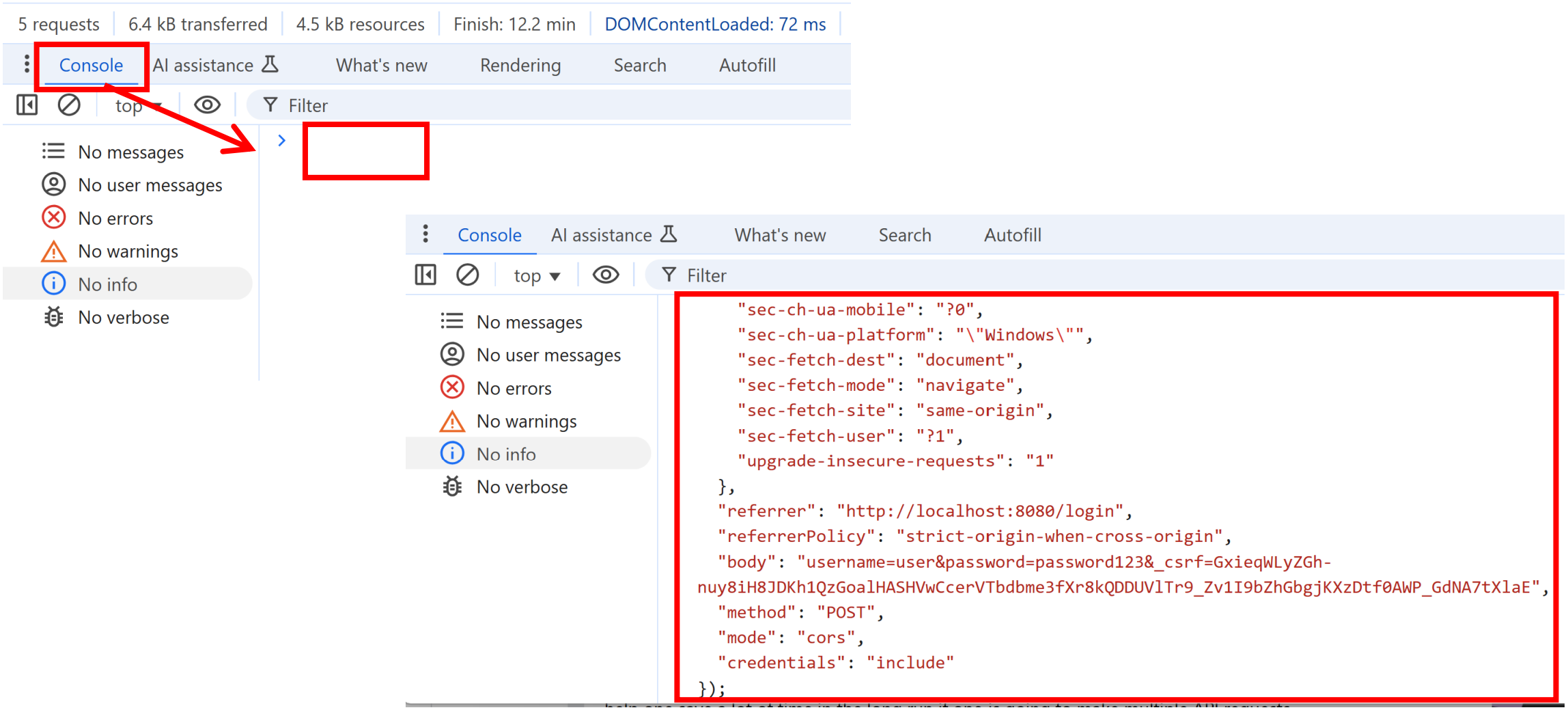
TD Goals = debug yourself the basic Http authentication scheme =

1/ install "Java JDK"
2/ install IDE : "IntelliJ IDEA"
3/ setup your app, using spring "initializr" click+download
4/ run your app
5/ open http://localhost:8080
  check app is a secured http webapp, with login/password dialog
6/ inspect network requests
7/ test Http requests

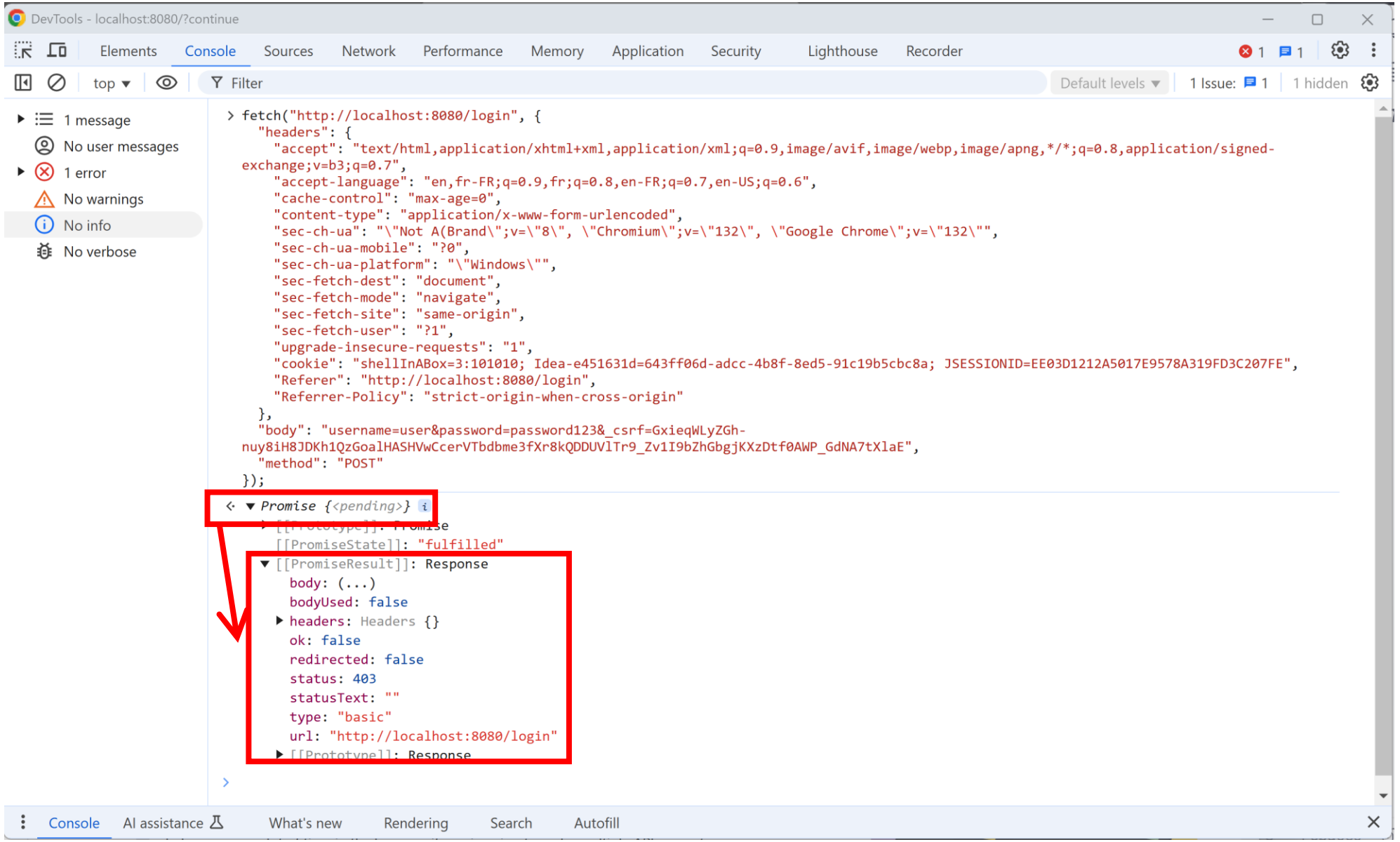
Copy Http Request from DevTools



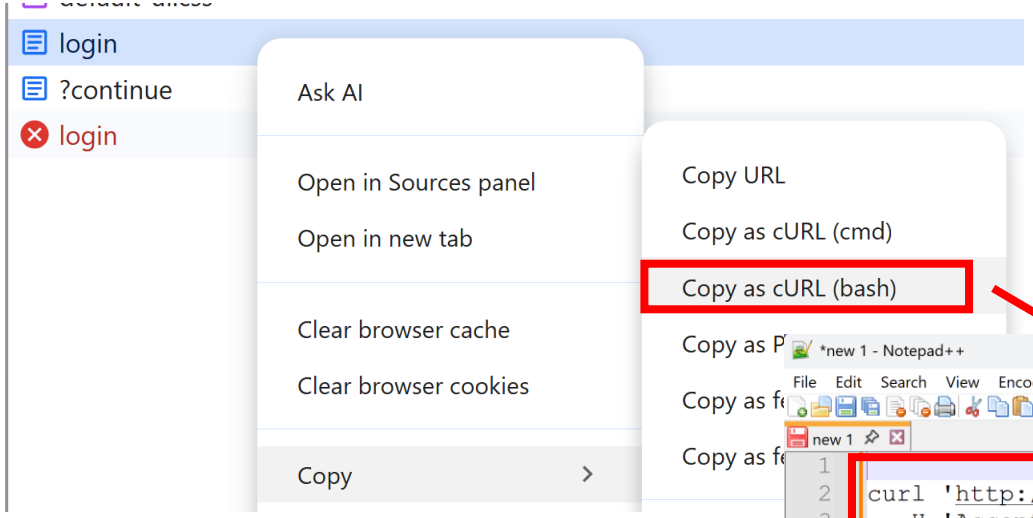
in Console, Paste > "fetch(...)"
then <enter> to Execute javascript code

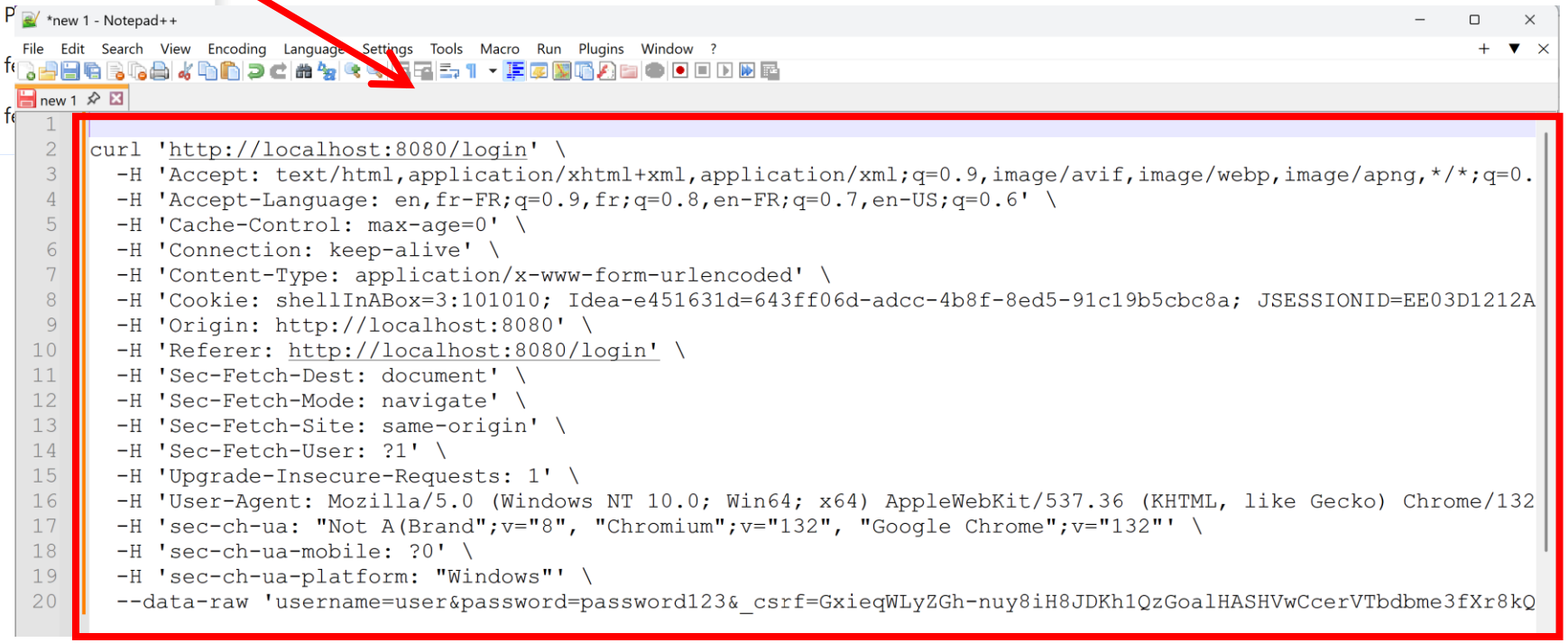


execute -> Promise -> Response -> status,body..



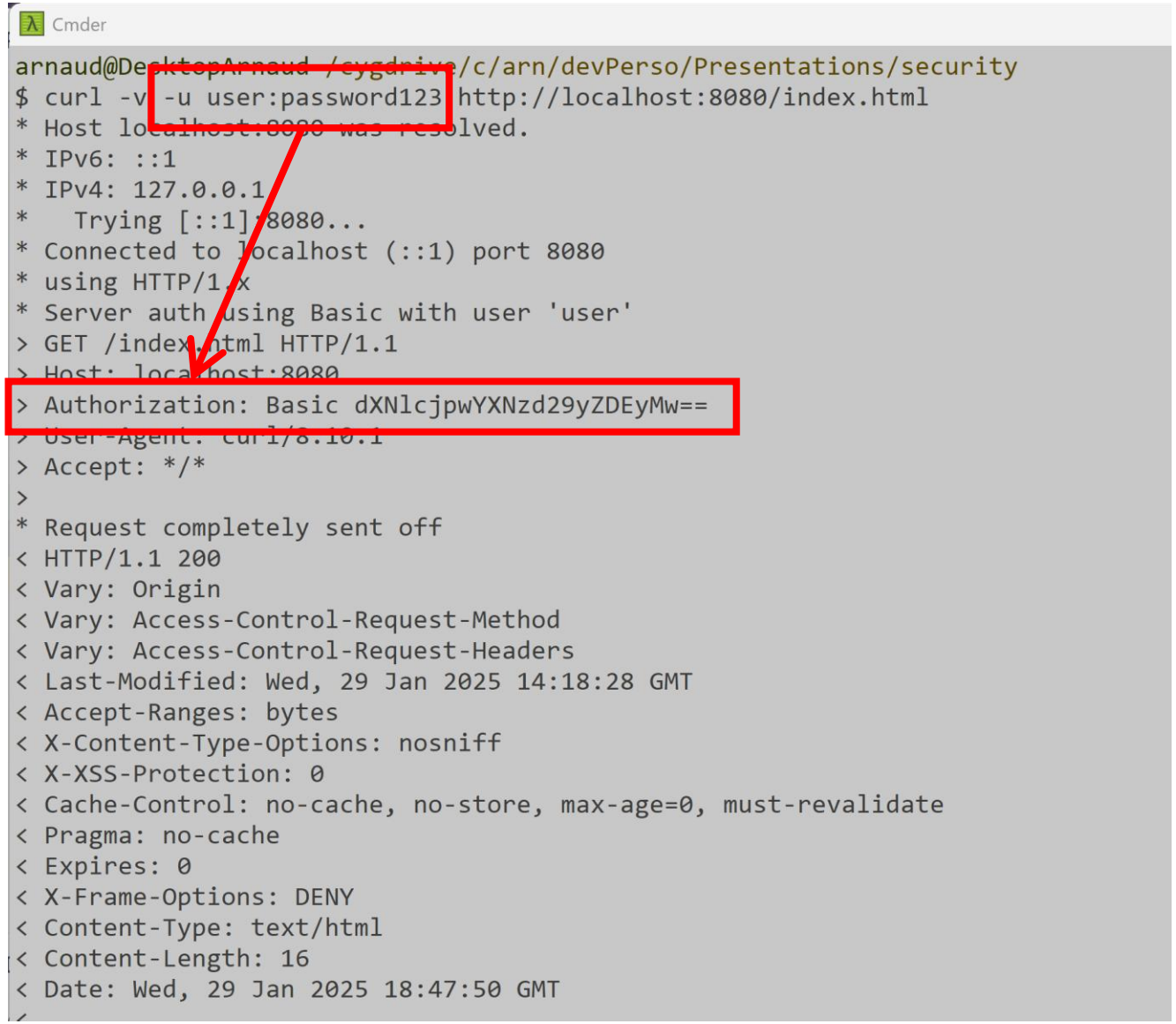
Other method to replay Http request ...
Export to "Curl"
(standard unix command line tool)



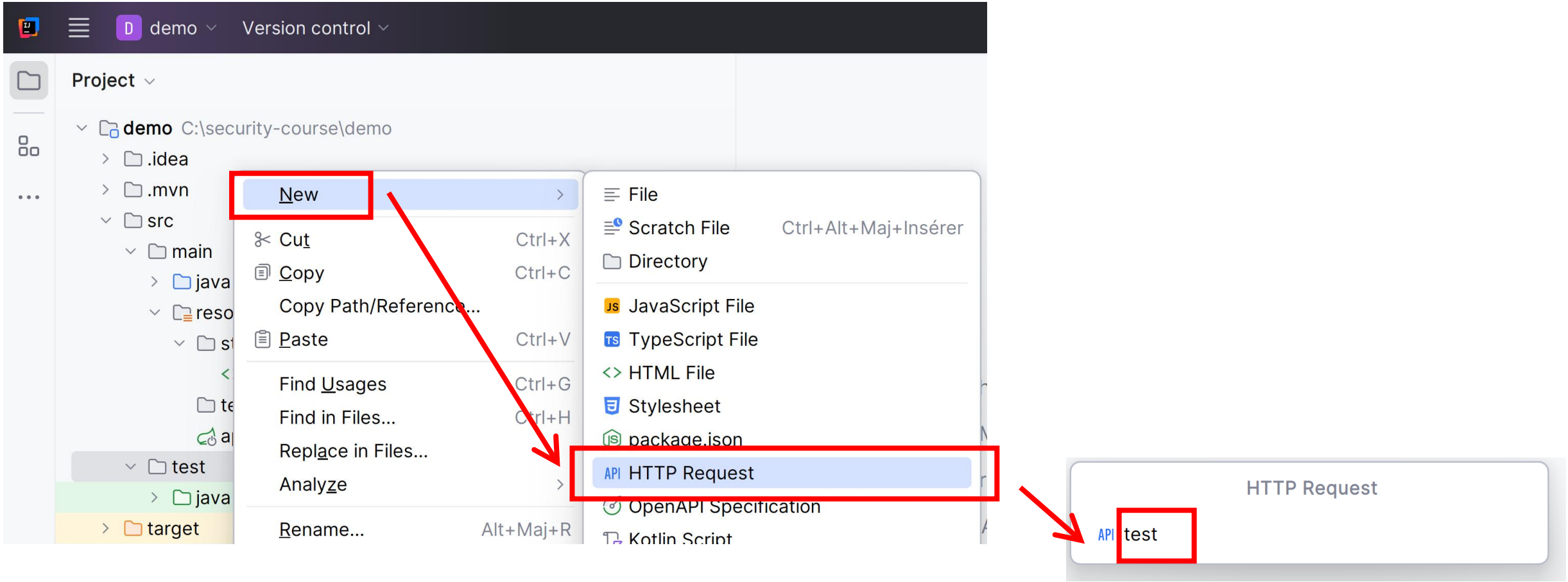


Execute curl command (bash)
simplify, fix newlines "\" + add "-v" for verbose

curl -u user:password
(encoded in base64, in header Authorization)



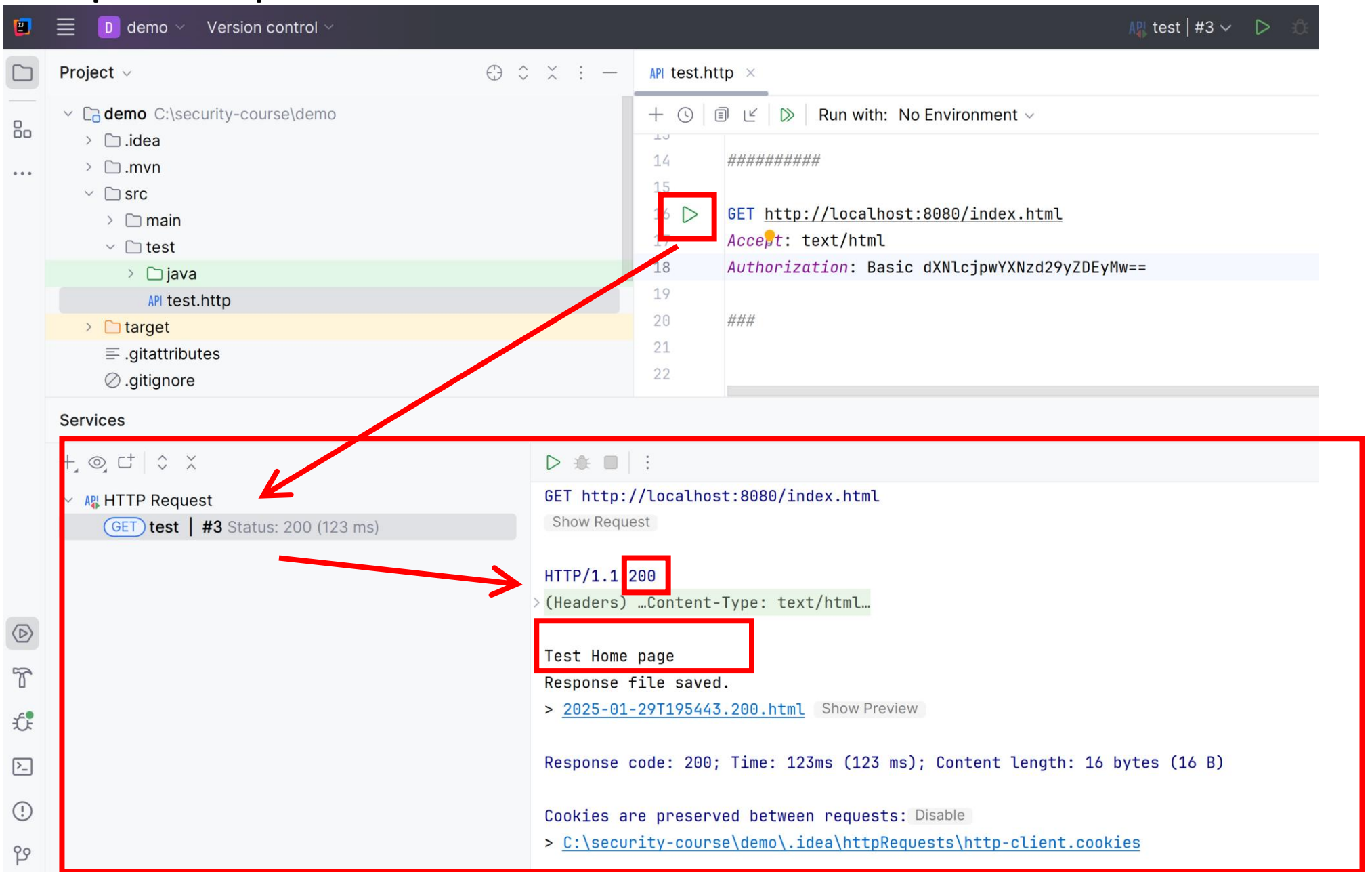
Other method to execute Http requests From IntelliJ, create "test.http" file



Edit Http Requests file


```
####
# simple non-interactive request, with base64 user:password

GET http://localhost:8080/index.html
Accept: text/html
Authorization: Basic dXNlcjpwYXNzd29yZDEyMw==

####
```

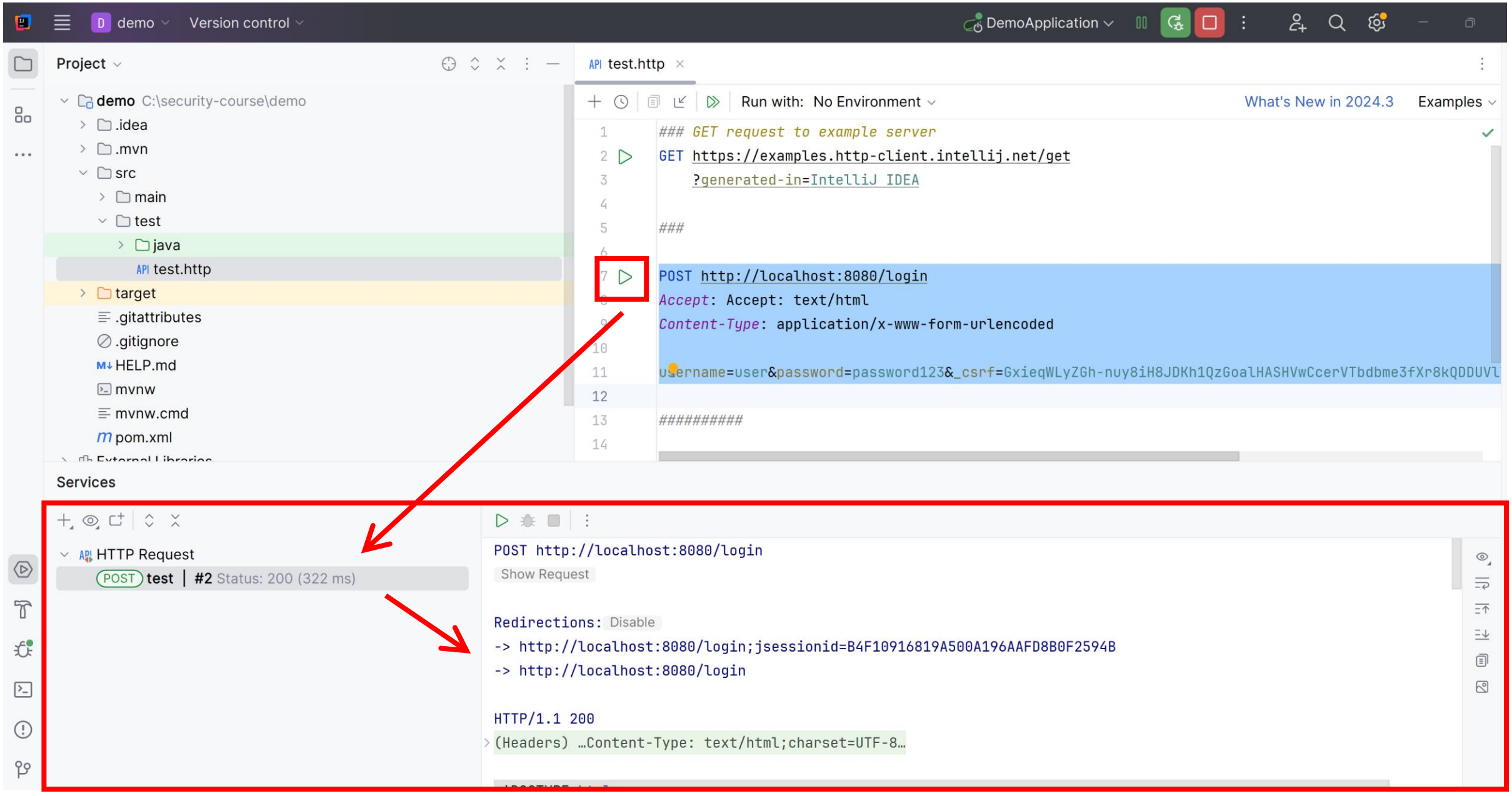
Run Http Request



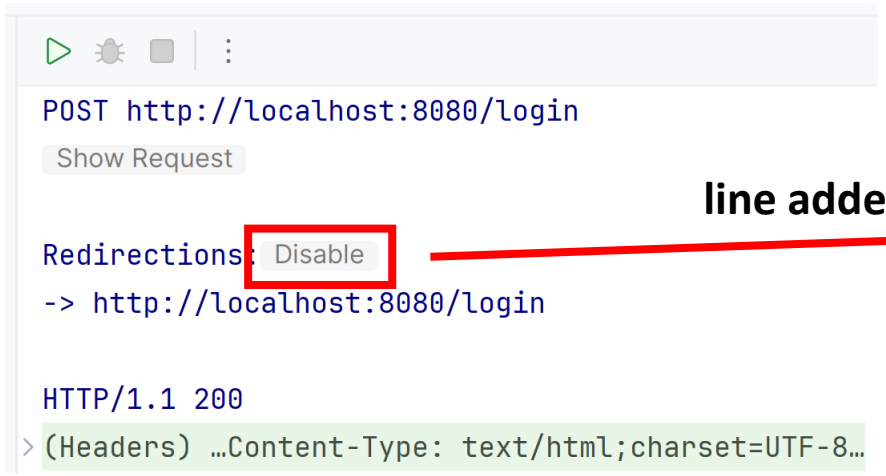
Edit More complex request interactive => redirect to login.html page..


```
############

POST http://localhost:8080/login
Accept: Accept: text/html
Content-Type: application/x-www-form-urlencoded

username=user&password=password123&_csrf=... (cf your inspected csrf value) ....
```

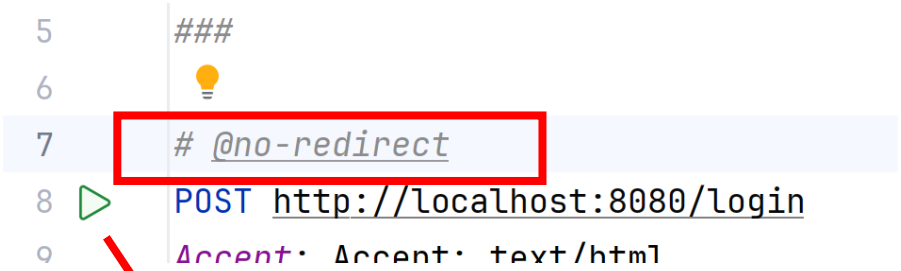
Run Http Request

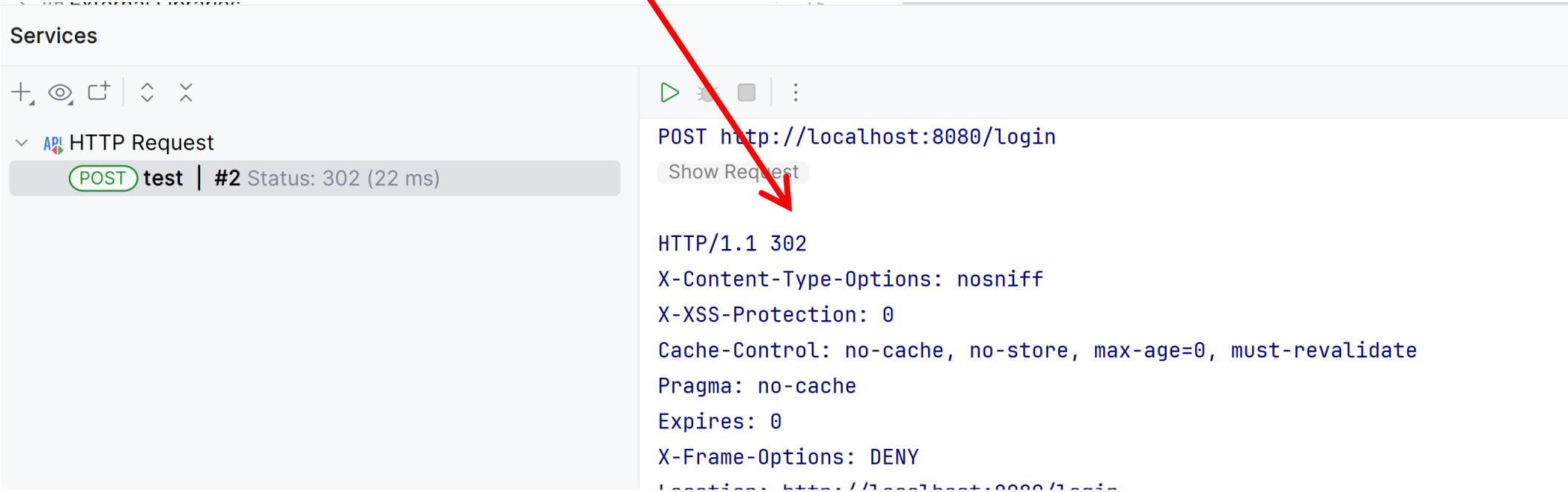


Run with "Disable redirection"



line added





Questions ?