

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)

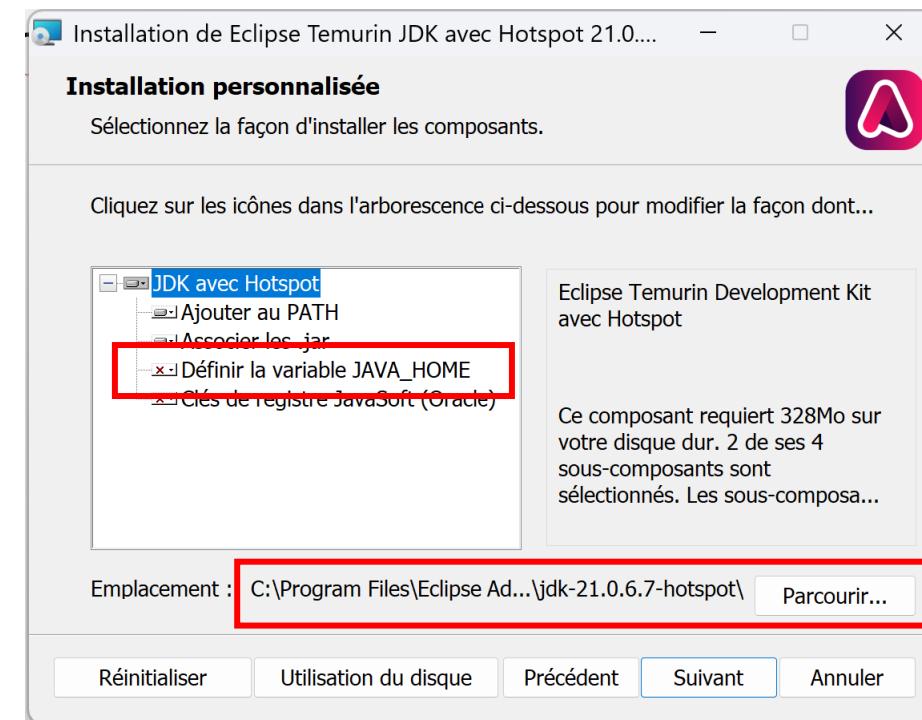
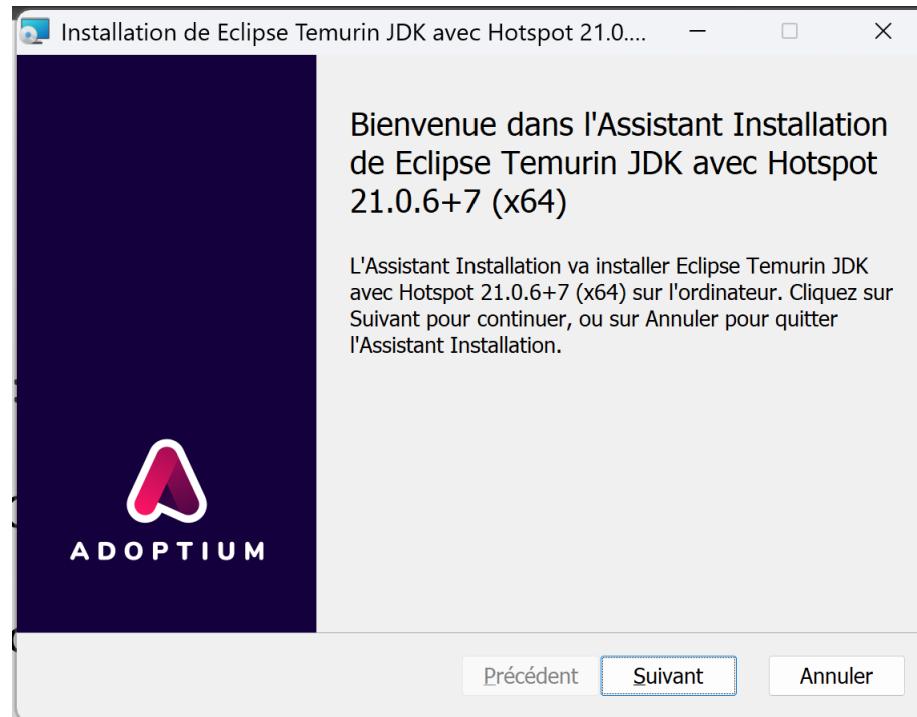
JDK is Open-Source, but pre-built binaries may be licenced if downloading from oracle.com

Download from « adoptium » (previously « adoptOpenJdk »)

<https://adoptium.net/>

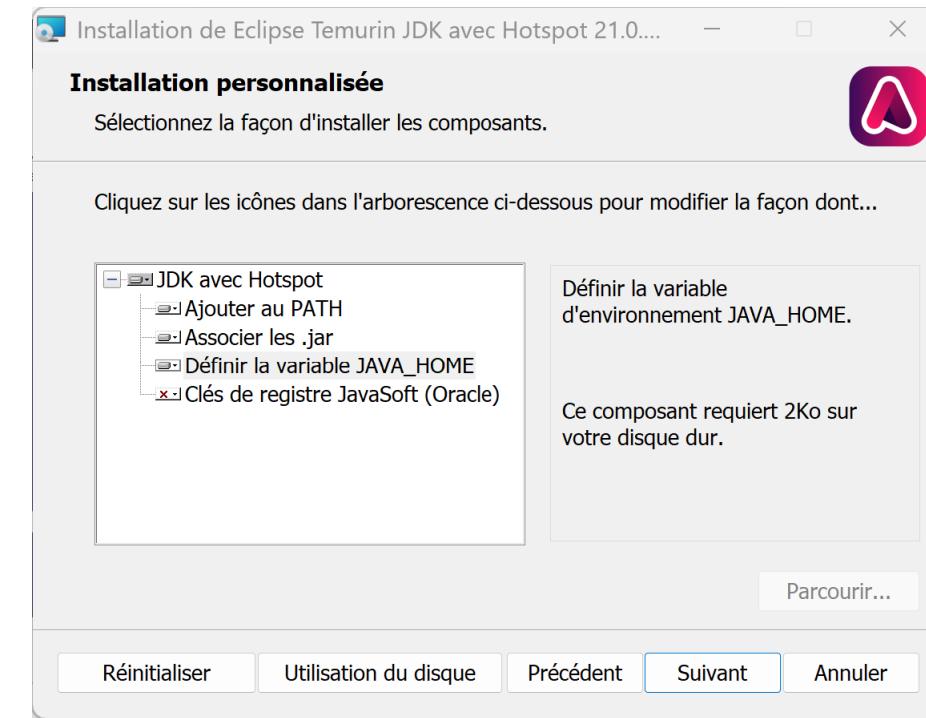
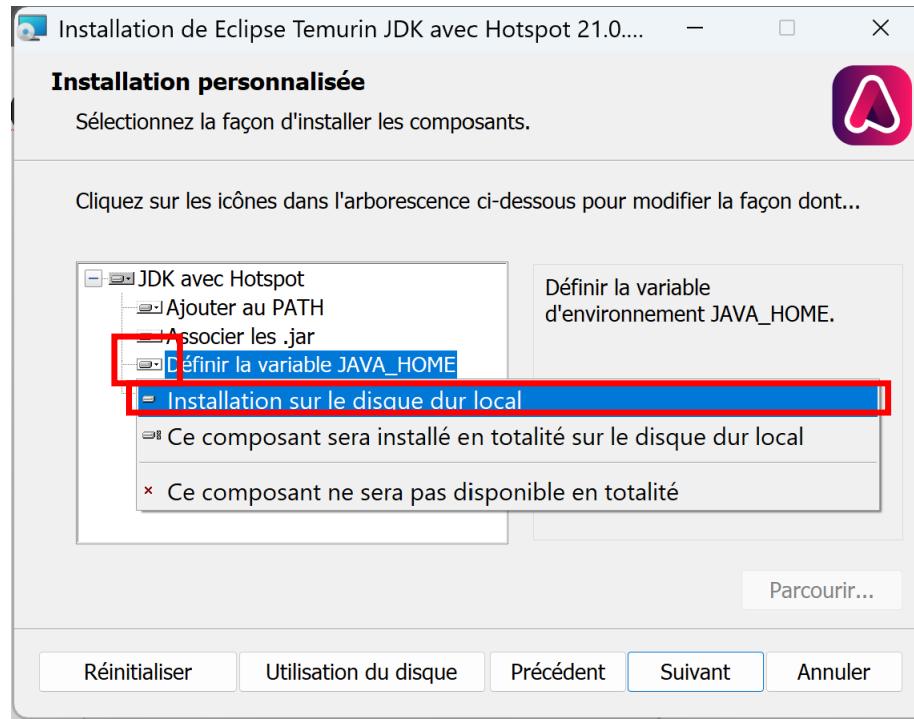
The screenshot shows the Adoptium website's main landing page. At the top, there's a dark header with the Adoptium logo and navigation links for Home, Marketplace, Documentation, FAQ, Projects, and Further Information. A red banner at the top of the main content area displays a message about the Eclipse Temurin 19 GA release. Below the banner, the text "Prebuilt OpenJDK Binaries for Free!" is prominently displayed. To the right of this text is a large, stylized 3D illustration of computer hardware, including a monitor, keyboard, and server racks. On the left, there's a detailed description of Java and the Adoptium Working Group, followed by a "Download Temurin™ for Windows x64" button. A "Latest LTS Release" button is also visible. At the bottom of the page, there's a link for "Other platforms and versions". The footer includes social media icons and a "Change Language" dropdown.

Installing ...



You may change defaults ... see next

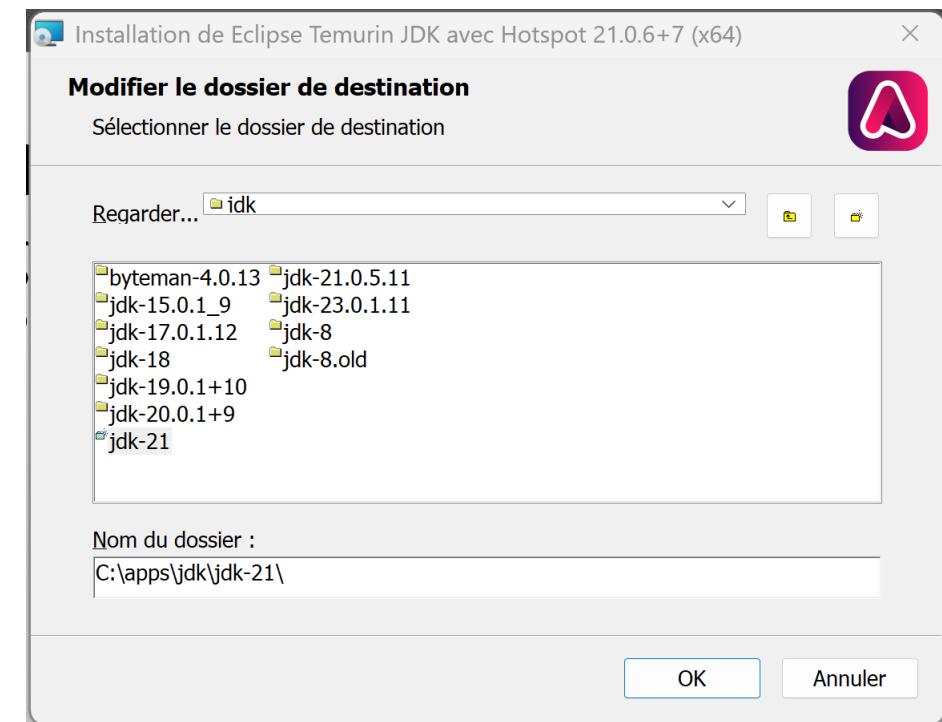
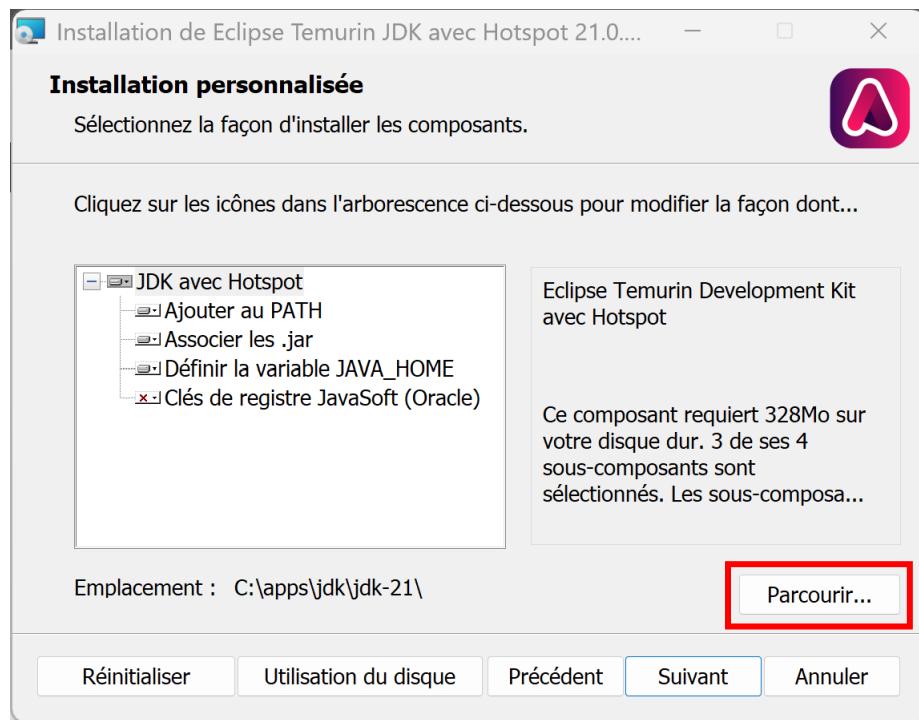
Optional : defining env variable JAVA_HOME



Optional

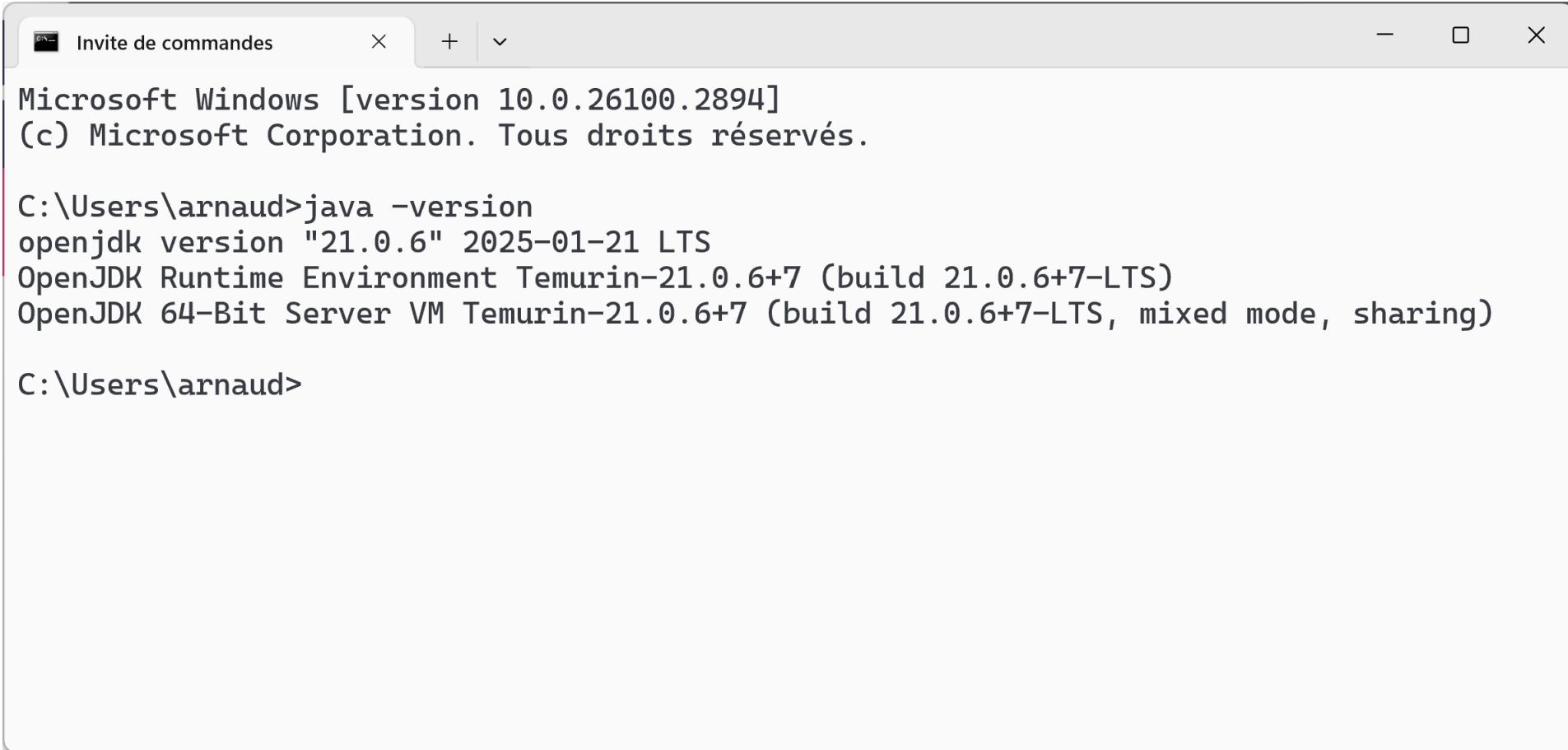
Changing directory "c:\apps\jdk"

(avoid long names, and space in dir name
so avoid "Program & Files\...")



Check Java installation OK

C:> java -version



The screenshot shows a Windows Command Prompt window titled "Invite de commandes". The window contains the following text:

```
Microsoft Windows [version 10.0.26100.2894]
(c) Microsoft Corporation. Tous droits réservés.

C:\Users\arnaud>java -version
openjdk version "21.0.6" 2025-01-21 LTS
OpenJDK Runtime Environment Temurin-21.0.6+7 (build 21.0.6+7-LTS)
OpenJDK 64-Bit Server VM Temurin-21.0.6+7 (build 21.0.6+7-LTS, mixed mode, sharing)

C:\Users\arnaud>
```

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>

The screenshot shows the official IntelliJ IDEA download page on the JetBrains website. At the top, there's a navigation bar with links for Developer Tools, Team Tools, Education, Solutions, Support, Store, and a search icon. Below the navigation is a main menu with IntelliJ IDEA selected, followed by JetBrains IDEs, Coming in 2024.3, What's New, Features, Resources, Pricing, and a prominent blue Download button.

Below the main menu, there are three tabs for different operating systems: Windows (selected), macOS, and Linux. The Windows section features a large "IntelliJ IDEA Ultimate" heading with a sub-headline "The Leading Java and Kotlin IDE". It includes a "Download" button and a ".exe (Windows)" dropdown menu, with the ".exe (Windows)" option highlighted and surrounded by a red box. A "Free 30-day trial" link is also present. To the right of this section is a screenshot of the IntelliJ IDEA interface showing a Java project structure and code editor tabs for VisitController.java and OwnerController.java.

At the bottom of the page, there's footer information including the version (2024.2.1), build number (Build: 242.21829.142), and date (29 August 2024). There are also links for System requirements, Other versions, Installation instructions, and Third-party software.

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

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

version "Ultimate" is much more complete

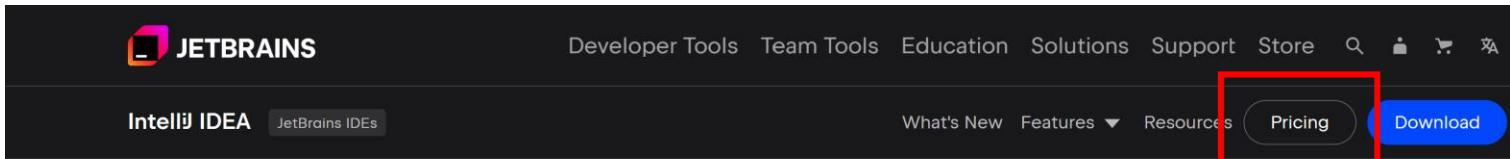
... and is FREE for students

... is paid by most companies for all developpers who ask

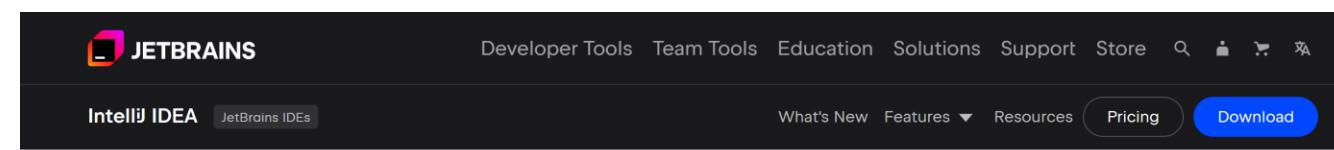
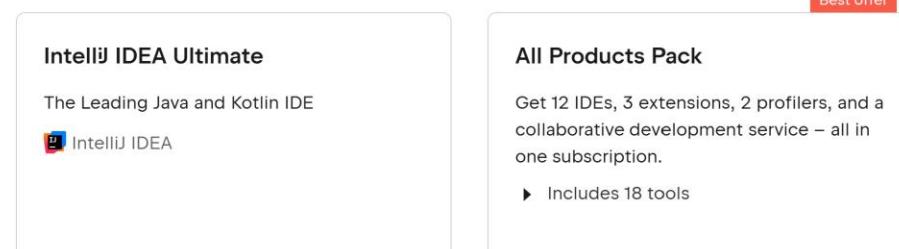
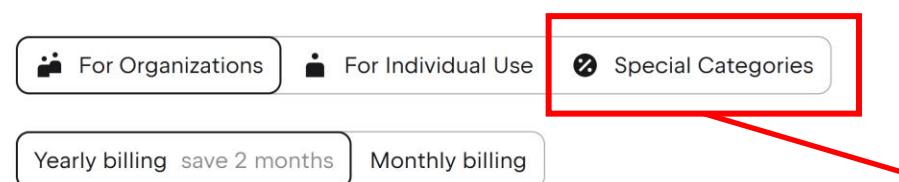
For doing today TD, you can just use "30-day trial"

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



Subscription Options and Pricing



Subscription Options and Pricing



For students and teachers FREE

Students and academic staff members are eligible to use all JetBrains tools free, upon verification of their university/college domain email or ISIC card.

[Learn more](#)

For classroom assistance FREE

Universities, colleges, schools, and non-commercial educational organizations are eligible for free licensing to install all JetBrains tools in classrooms and computer labs for educational purposes.

[Learn more](#)

For Open Source projects FREE

Core maintainers of open-source projects are welcome to explore collaboration opportunities and receive free access to JetBrains tools.

[Learn more](#)

For universities and educational organizations 50% off

For startups 50% off

Private software development companies which

For training courses, coding schools, and bootcamps FREE

Steps ...

The diagram illustrates the process of applying for a JetBrains educational license. It consists of two screenshots:

- Screenshot 1: Free educational licenses**
 - Shows four rules:
 - 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.
 - A large button at the bottom says "Get free access to all developer tools from JetBrains!" with a red box around the "Apply now" button.

A red arrow points from the "Apply now" button in Screenshot 1 to the "University email address" field in Screenshot 2.

Screenshot 2: JetBrains Products for Learning
 - Header: "JetBrains Products for Learning"
 - Text: "Before you apply, please read the [Educational Subscription Terms and FAQ](#).
 - Form fields:
 - "Apply with": "University email address" (highlighted with a red box)
 - "Status": Radio buttons for "I'm a student" (selected) and "I'm a teacher".
 - "Country / region": "France"
 - "Level of study": "Undergraduate"
 - "Is Computer Science or Engineering your major field of study?": Radio buttons for "Yes" (selected) and "No".
 - "Email address": "University email address @esilv" (highlighted with a red box)

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>

springboot starter

All Images Videos News Web Books Finance Tools

These are results for **spring boot** starter
Search instead for [springboot starter](#)

 Spring Initializr
<https://start.spring.io> ::

[Spring Initializr](#)
Initializr generates **spring boot** project with just what you need to start quickly!

Open Springboot Initializr

<https://start.spring.io>

The screenshot shows the Spring Initializr web application at <https://start.spring.io>. The interface is a form-based configuration tool for generating Spring Boot projects.

Project Configuration:

- Project:** Maven (selected)
- Language:** Java (selected)
- Spring Boot:** 3.4.2 (selected)
- Spring Boot Versions:** 3.5.0 (SNAPSHOT), 3.5.0 (M1), 3.4.3 (SNAPSHOT), 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 (selected)

Dependencies: No dependency selected. A button labeled "ADD DEPENDENCIES... CTRL + B" is available.

Build Tools: Java 22, Java 21, Java 17 (selected)

Buttons: GENERATE CTRL + ⌘, EXPLORE CTRL + SPACE, ...

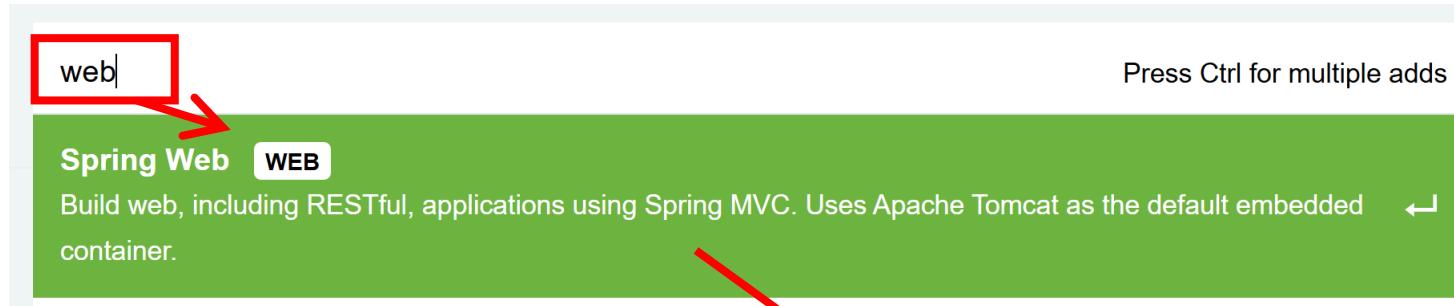
Click "ADD DEPENDENCIES..."

The screenshot shows the Spring Initializer web interface at start.spring.io. The interface is divided into several sections:

- Project Selection:** Shows options for Project (Gradle - Groovy, Gradle - Kotlin, Maven), Spring Boot (3.5.0 (SNAPSHOT), 3.5.0 (M1), 3.3.9 (SNAPSHOT), 3.3.8), and Project Metadata (Group: com.example, Artifact: demo, Name: demo, Description: Demo project for Spring Boot, Package name: com.example.demo, Packaging: Jar).
- Developer Tools:** A green section listing tools like GraalVM Native Support, GraphQL DGS Code Generation, Spring Boot DevTools, Lombok, Spring Configuration Processor, Docker Compose Support, and Spring Modular.
- Dependencies:** A section where users can add dependencies. It includes a search bar with placeholder text "Web, Security, JPA, Actuator, Devtools...", a button to "ADD DEPENDENCIES... CTRL + B", and another button to "ADD DEPENDENCIES... CTRL + B".

Two red arrows point from the text "Click \"ADD DEPENDENCIES...\" in the video" to the "ADD DEPENDENCIES..." buttons in the Dependencies section and the developer tools section.

Add "web"



A screenshot of the Spring Initializr configuration interface. On the left, there are sections for "Project" (Gradle - Groovy, Gradle - Kotlin, Maven, Maven is selected), "Language" (Java, Kotlin, Groovy, Java is selected), and "Spring Boot" (3.5.0 (SNAPSHOT), 3.5.0 (M1), 3.4.3 (SNAPSHOT), 3.4.2, 3.3.9 (SNAPSHOT), 3.3.8, 3.4.2 is selected). On the right, there is a "Dependencies" section with a card for "Spring Web WEB" which has the same description as the one in the search interface. A red box highlights this card. Below the dependencies, there is a "Project Metadata" section with fields: Group (com.example), Artifact (demo), Name (demo), Description (Demo project for Spring Boot), Package name (com.example.demo), and Packaging (Jar, Jar is selected). At the bottom, there are buttons for "GENERATE CTRL + ⌘" and "EXPLORE CTRL + SPACE".

Then add "security"

The screenshot shows a user interface for managing project dependencies. At the top left, there is a search bar containing the text "security" with a red arrow pointing to it. To the right of the search bar is a button labeled "Press Ctrl for multiple adds". Below the search bar, a green card displays the "Spring Security" dependency, which is categorized under "SECURITY". The card includes a brief description: "Highly customizable authentication and access-control framework for Spring applications." On the right side of the interface, there is a vertical scroll bar. In the center, there is a section titled "Dependencies" with a button labeled "ADD DEPENDENCIES... CTRL + B". Below this, two dependency cards are listed: "Spring Web" (WEB) and "Spring Security" (SECURITY). Both cards have their descriptions highlighted with a red rectangle. A large red arrow points from the search result "security" down towards the "Spring Security" dependency card.

Press Ctrl for multiple adds

security

Spring Security SECURITY

Highly customizable authentication and access-control framework for Spring applications.

Dependencies

ADD DEPENDENCIES... CTRL + B

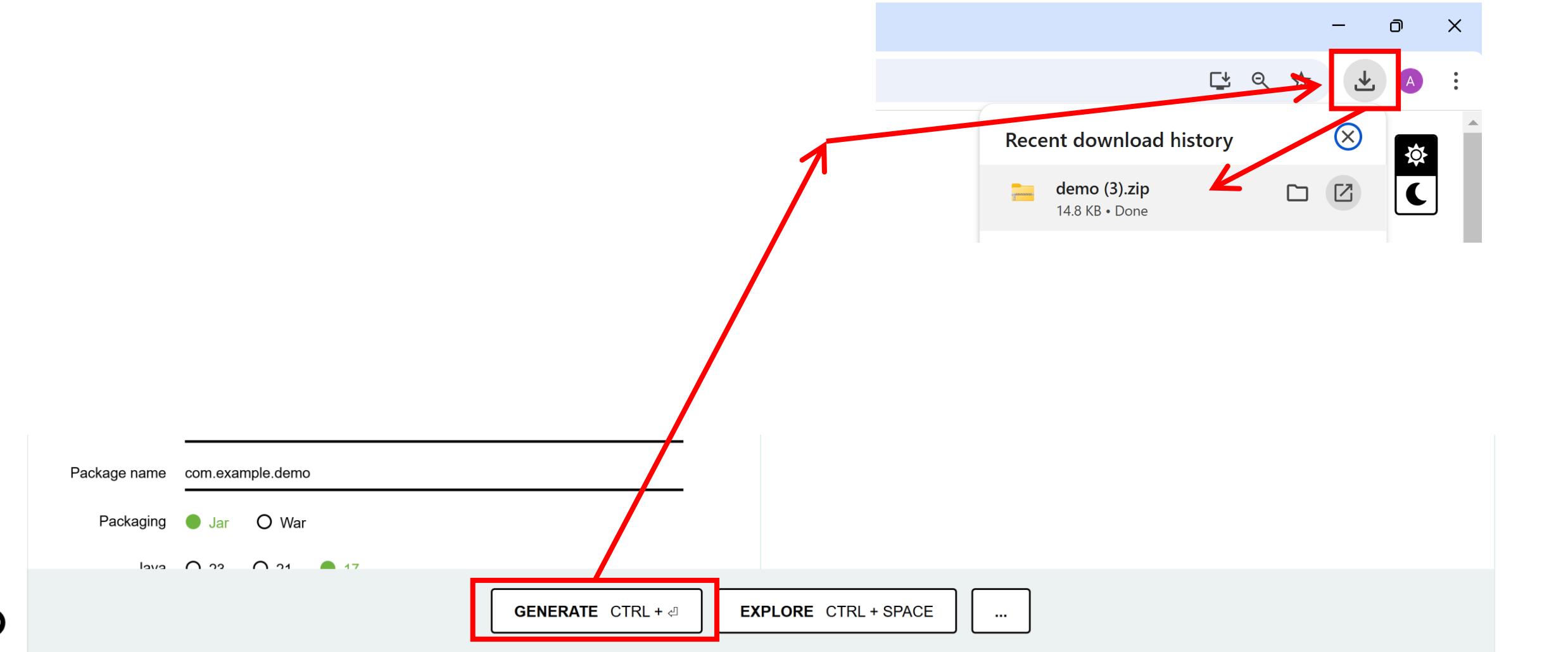
Spring Web WEB

Build web, including RESTful, applications using Spring MVC. Uses Apache Tomcat as the default embedded container.

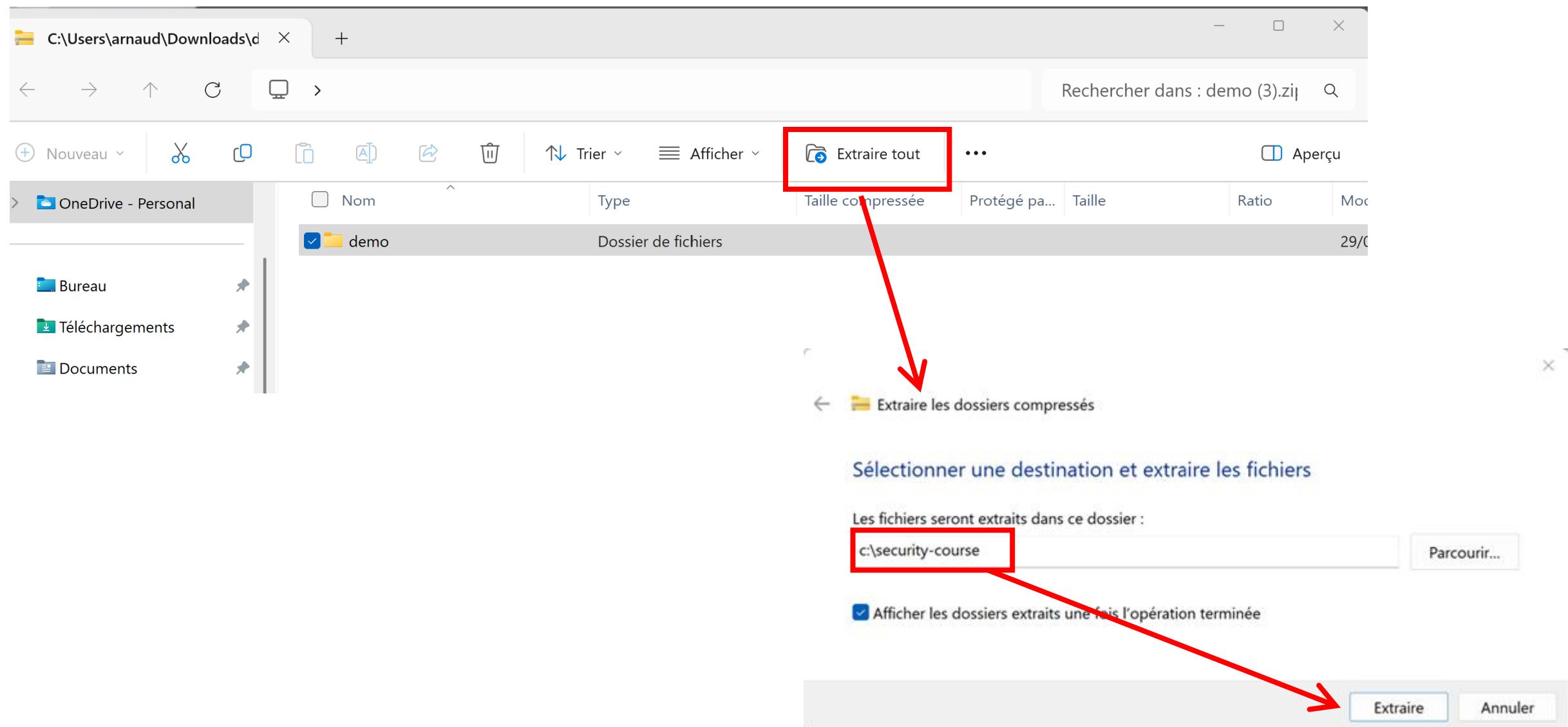
Spring Security SECURITY

Highly customizable authentication and access-control framework for Spring applications.

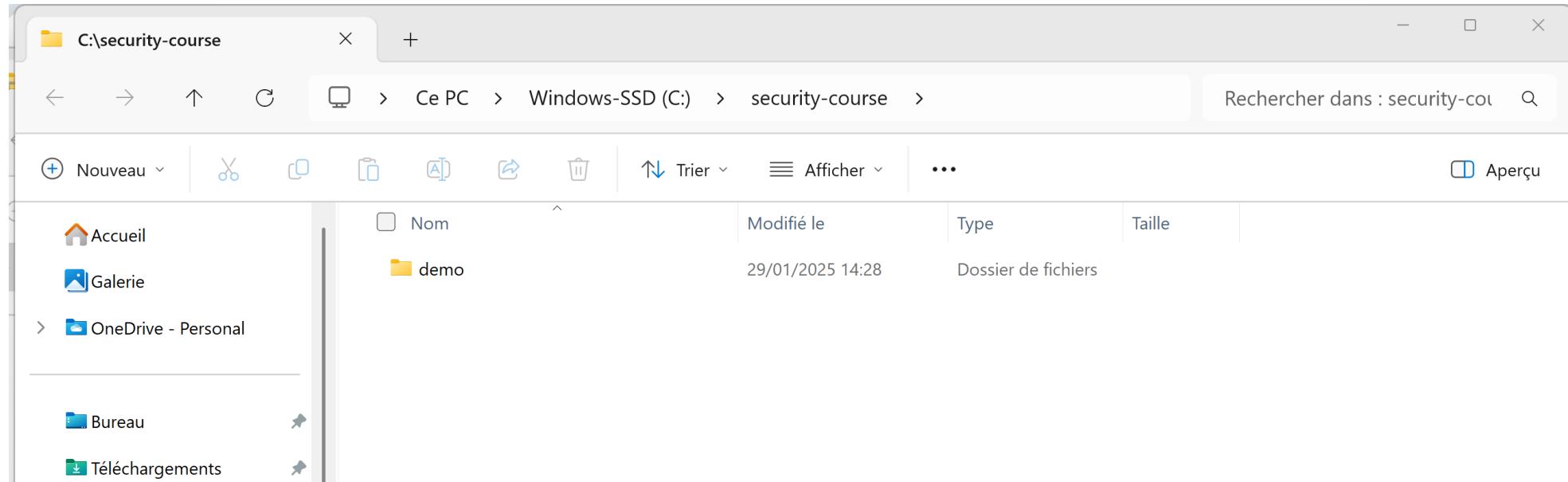
Then Click "GENERATE", open downloaded zip



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



Explore Dir & Files



Basically 2 interesting 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"

The screenshot shows a web interface for exploring a Spring Boot project. On the left, a sidebar displays the contents of a zip archive named "demo.zip". The contents include:

- .gitattributes
- .gitignore
- .mvn
- HELP.md
- mvnw
- mvnw.cmd
- pom.xml
- src
 - main
 - java
 - com
 - example
 - demo
 - resources
 - application.properties
 - static

On the right, the "pom.xml" file is displayed as an XML document. The XML code is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>3.4.2</version>
    <relativePath/> <!-- Lookup parent from repository -->
  </parent>
  <groupId>com.example</groupId>
  <artifactId>demo</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>demo</name>
  <description>Demo project for Spring Boot</description>
  <url/>
  <licenses>
    <license/>
  </licenses>
  <developers>
    <developer/>
  </developers>
  <scm>
    <connection/>
  </scm>
</project>
```

Two buttons are visible at the top right: "DOWNLOAD" and "COPY".

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



```
m pom.xml (demo) ×

2 <project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-in
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

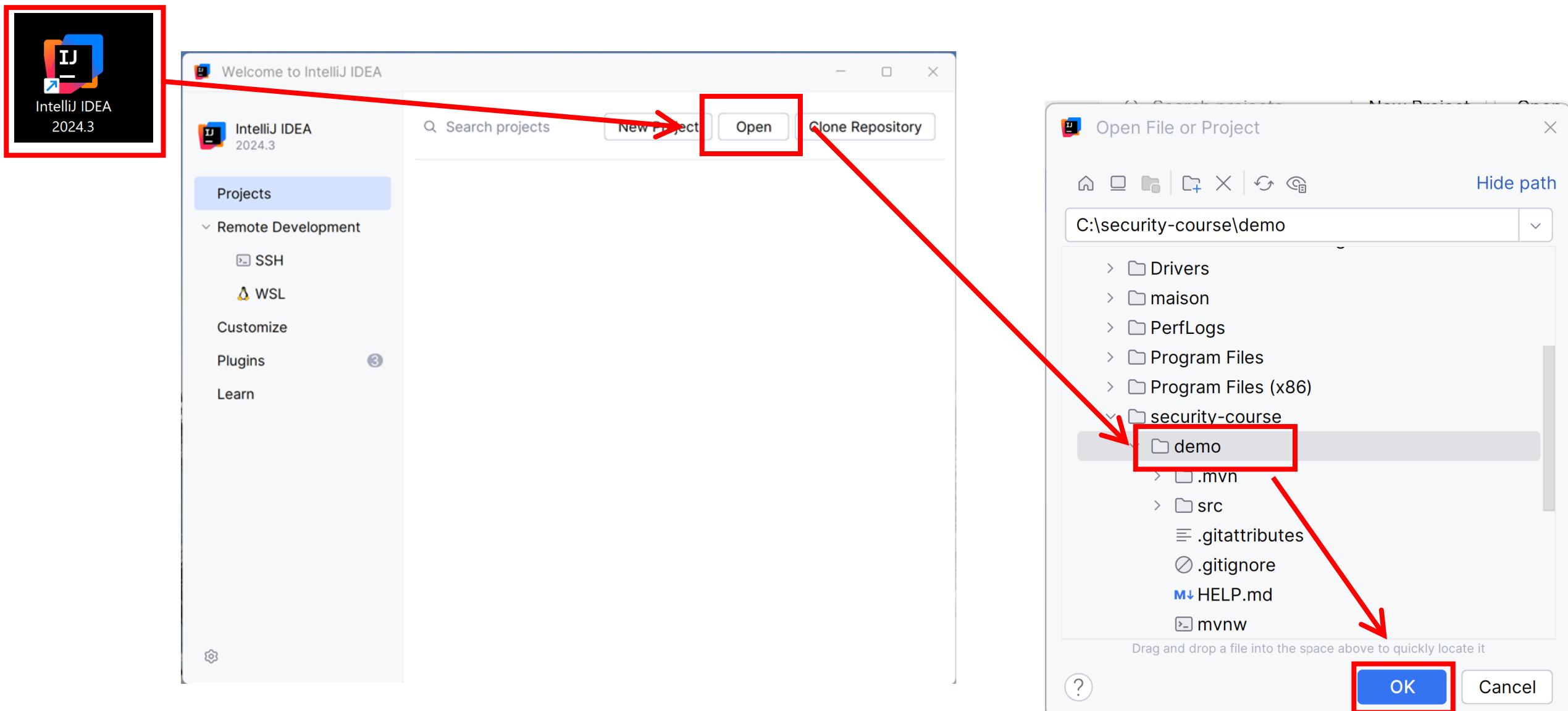
5/ open http://localhost:8080

check app is a secured http webapp, with login/password dialog

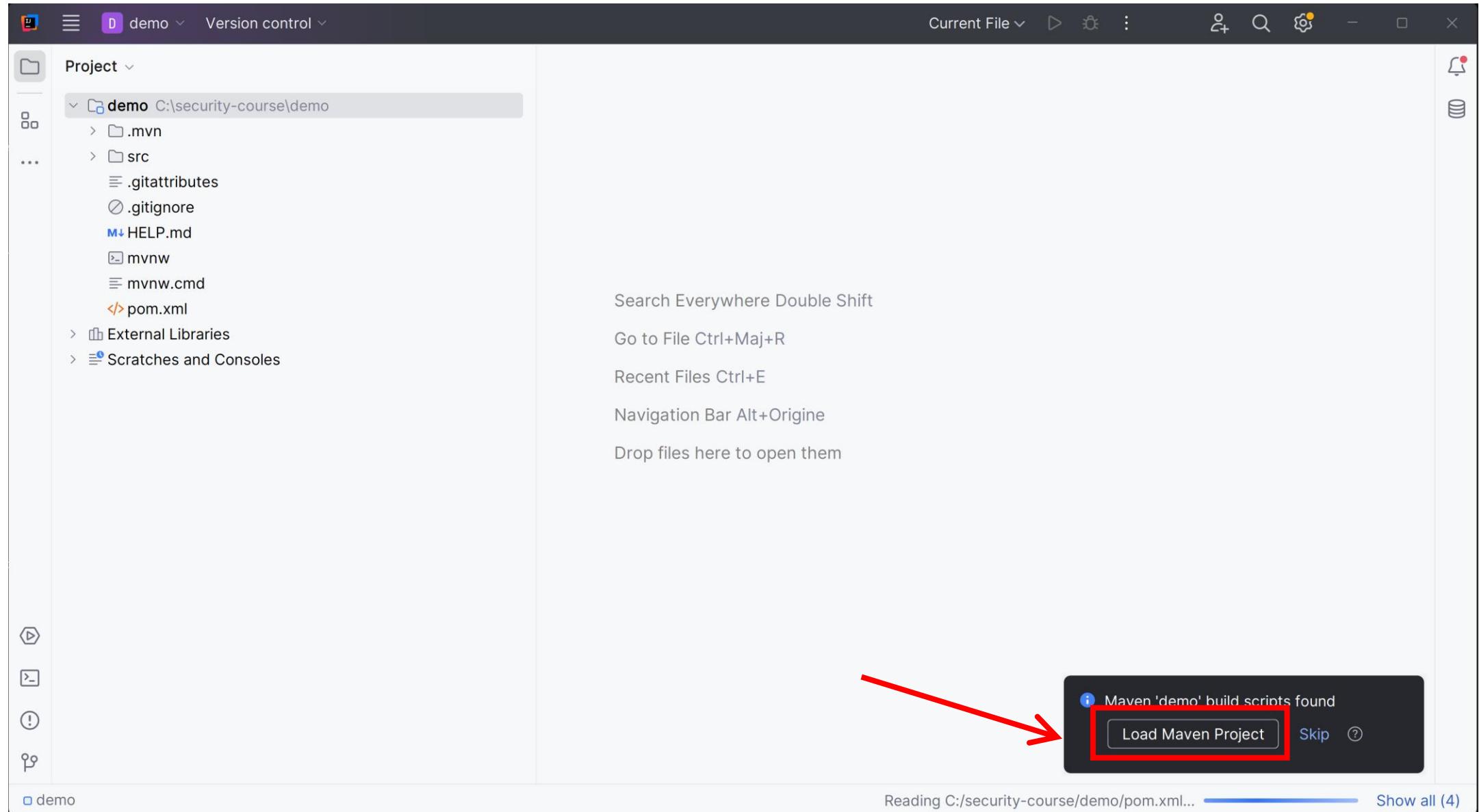
6/ inspect network requests

7/ test Http requests

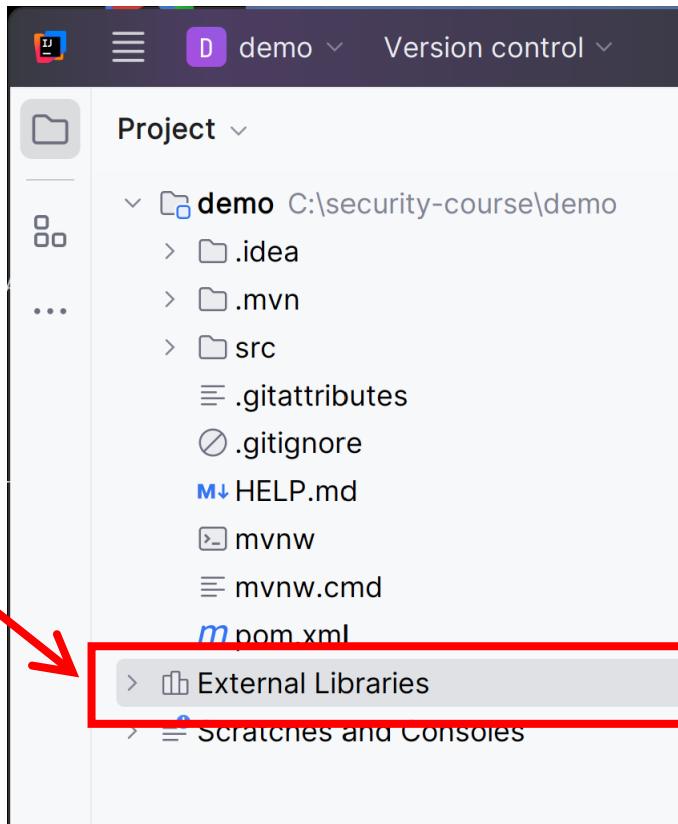
Launch your IDEA IntelliJ Open Project



Click default "Load Maven Project"



Wait few minutes (downloading dependencies ...)



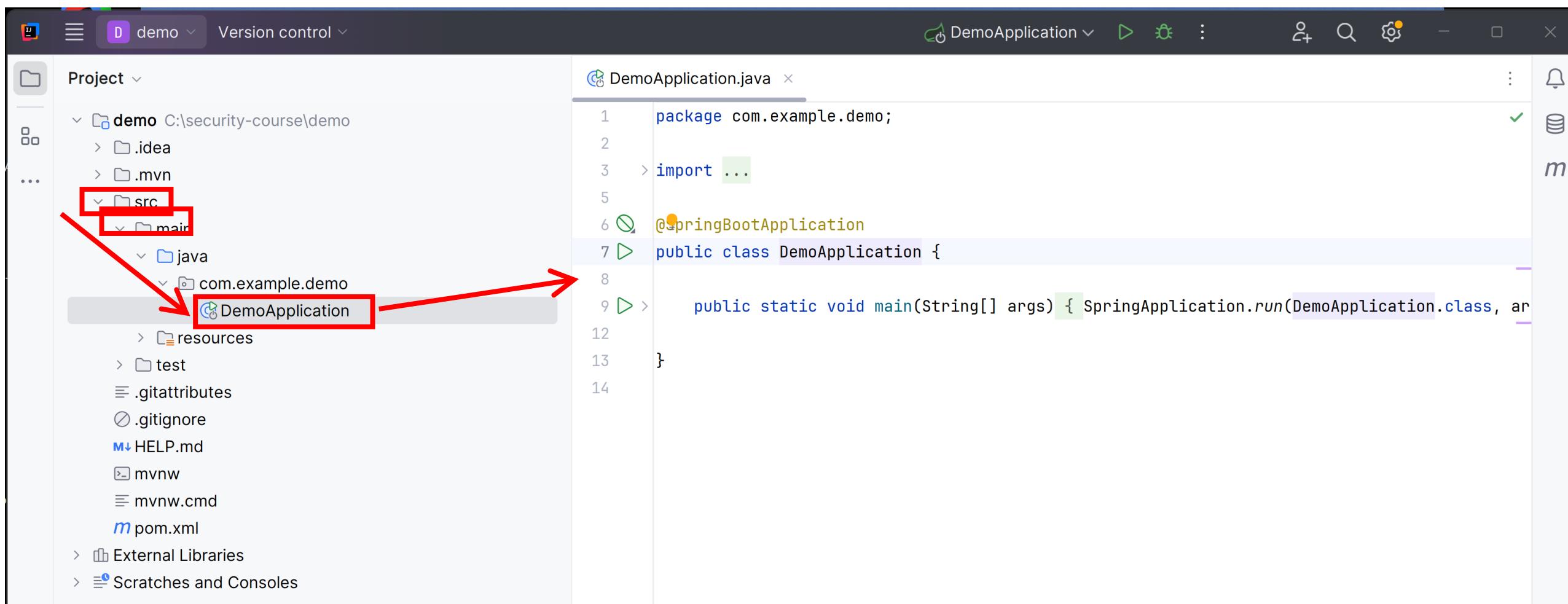
when finished
All dependencies
are automatically
downloaded

spring -> tomcat -> ...

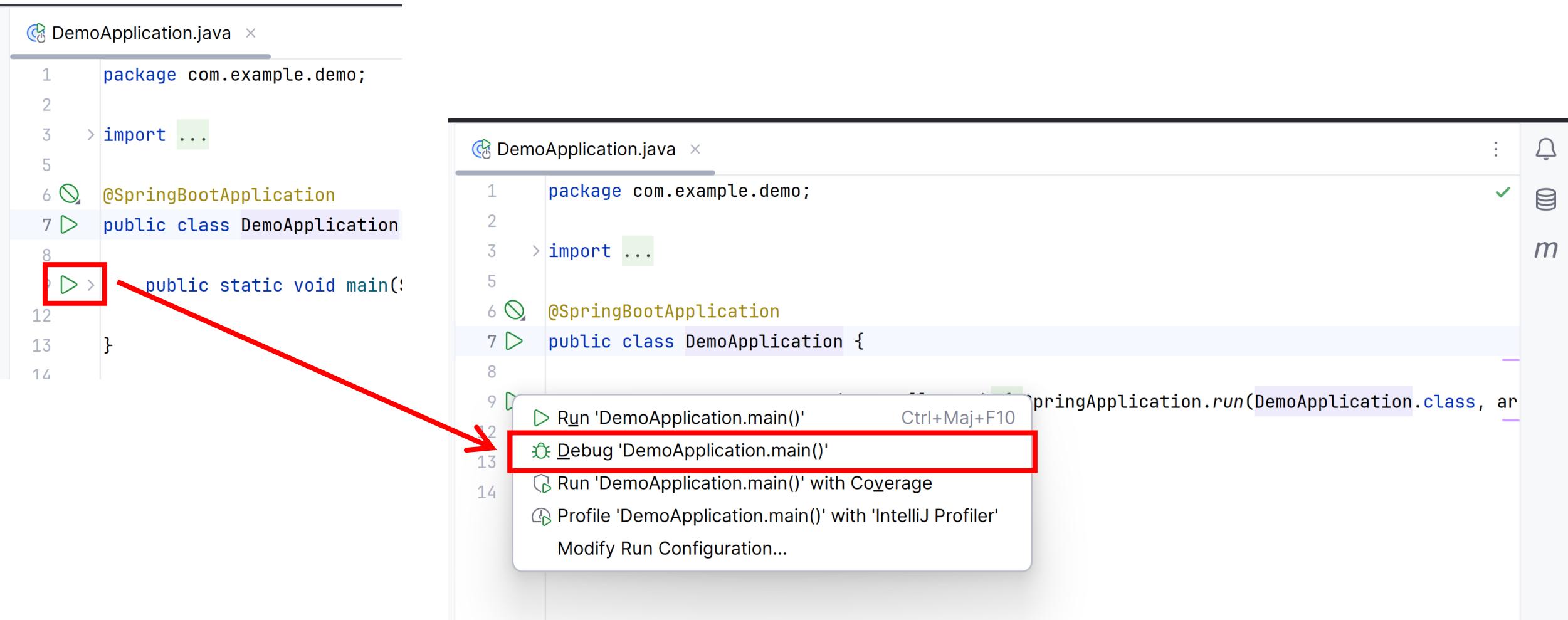
The screenshot shows the 'External Libraries' section of the 'pom.xml' editor. A red box highlights the entire list of dependencies. The list includes:

- < 17 > C:\apps\jdk\jdk-20.0.1+9
- Maven: ch.qos.logback:logback-classic:1.5.16
- Maven: ch.qos.logback:logback-core:1.5.16
- Maven: com.fasterxml.jackson.core:jackson-annotations:2.18.2
- Maven: com.fasterxml.jackson.core:jackson-core:2.18.2
- Maven: com.fasterxml.jackson.core:jackson-databind:2.18.2
- Maven: com.fasterxml.jackson.datatype:jackson-datatype-pe
- Maven: com.fasterxml.jackson.datatype:jackson-datatype-pe
- Maven: com.fasterxml.jackson.module:jackson-module-pe
- Maven: com.jayway.jsonpath:json-path:2.9.0
- Maven: com.vaadin.external.google:android-json:0.0.2013
- Maven: io.micrometer:micrometer-commons:1.14.3
- Maven: io.micrometer:micrometer-observation:1.14.3
- Maven: jakarta.activation:jakarta.activation-api:2.1.3
- Maven: jakarta.annotation:jakarta.annotation-api:2.1.1
- Maven: jakarta.xml.bind:jakarta.xml.bind-api:4.0.2
- Maven: net.bytebuddy:byte-buddy:1.15.11
- Maven: net.bytebuddy:byte-buddy-agent:1.15.11
- Maven: net.minidev:accessors-smart:2.5.1
- Maven: net.minidev:json-smart:2.5.1
- Maven: org.apache.logging.log4j:log4j-api:2.24.3
- Maven: org.apache.logging.log4j:log4j-to-slf4j:2.24.3
- Maven: org.apache.tomcat.embed:tomcat-embed-core:10.0.17

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

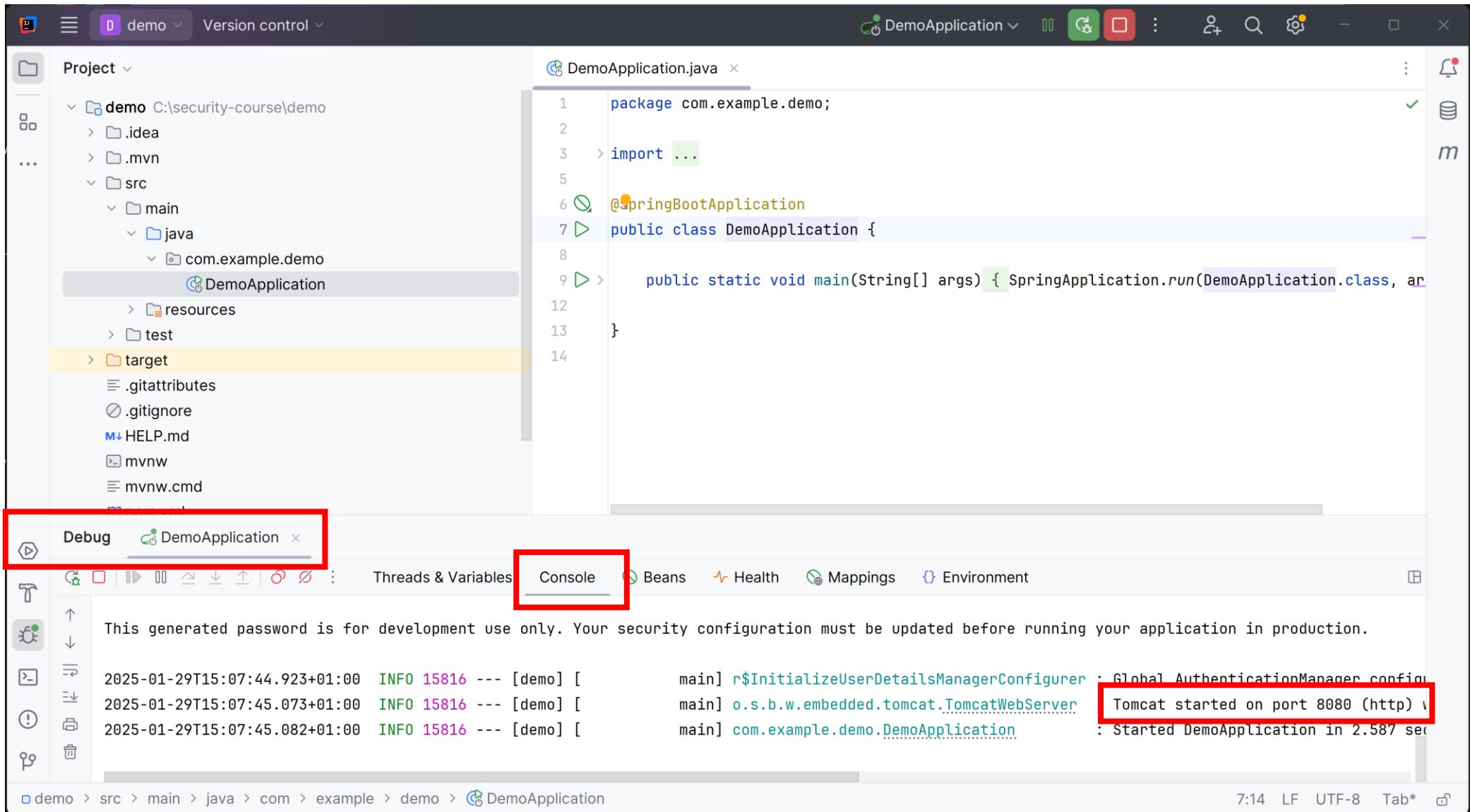


Click on "Run" Debug button

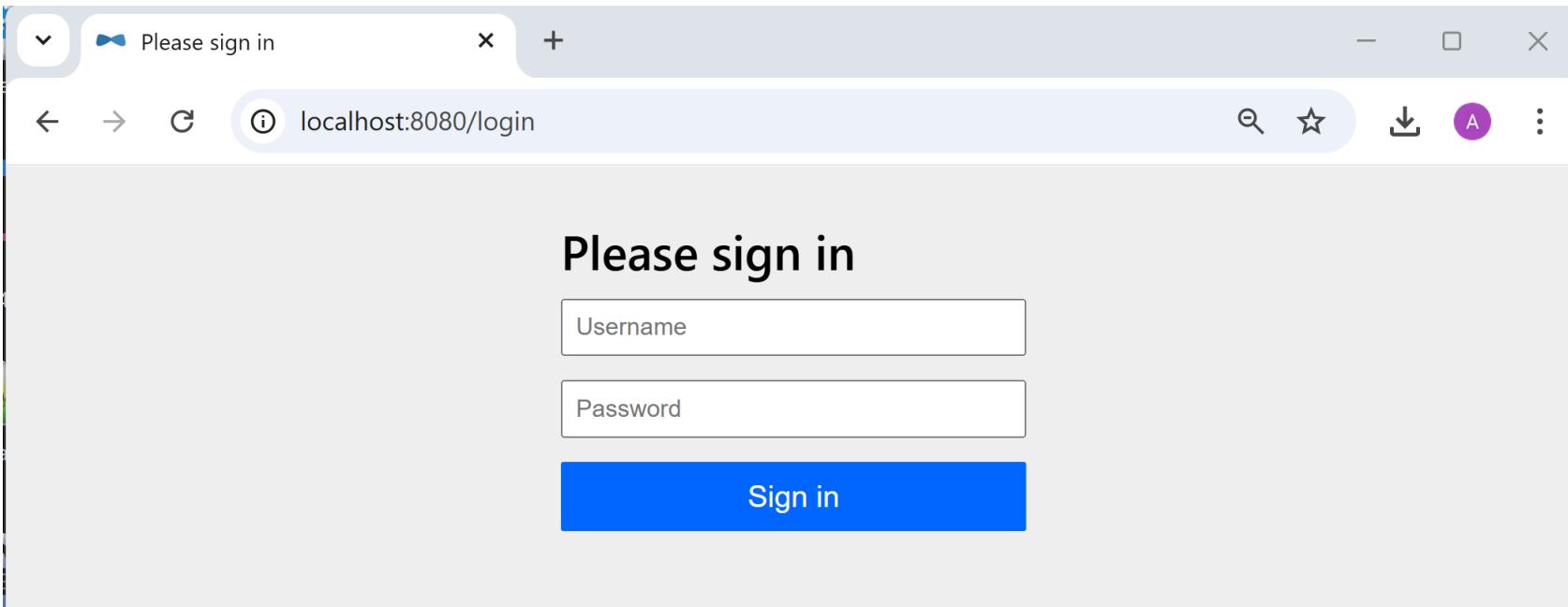


See Debug "Console" logs...

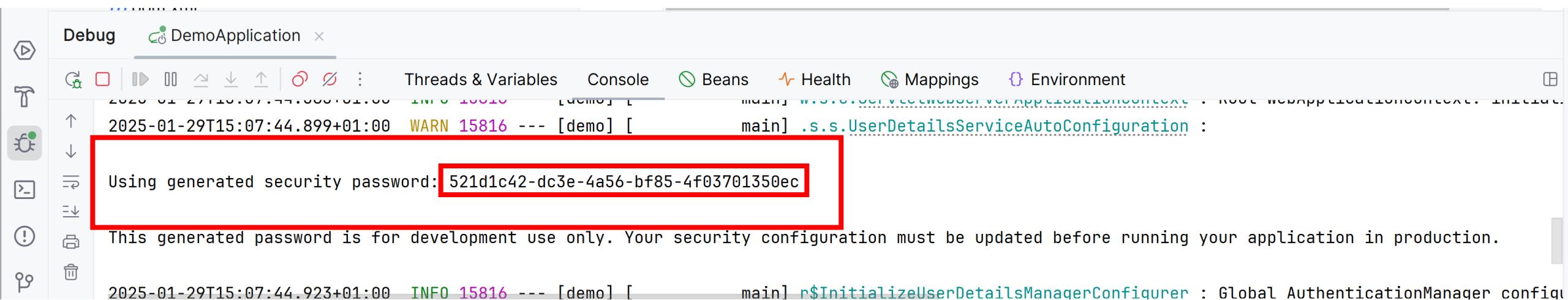
Tomcat started on port 8080 (http)



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



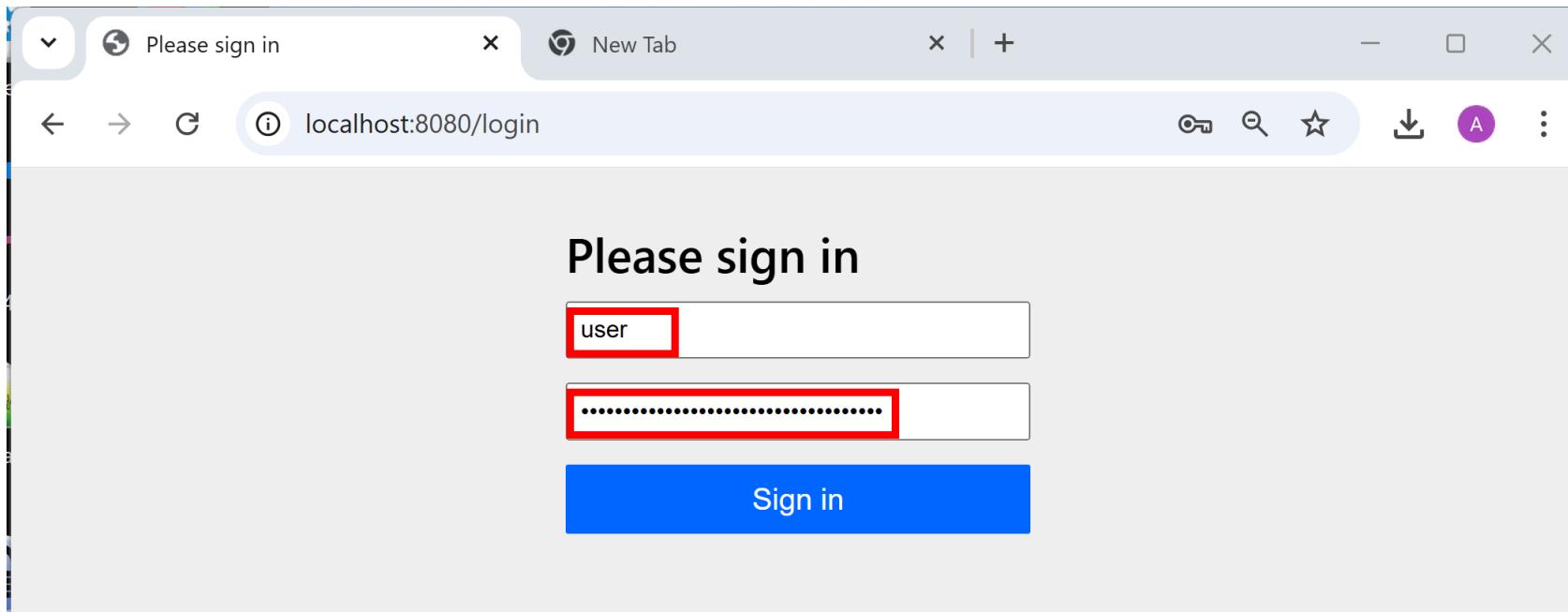
See "user" / generated password in console
(none configured yet by default)



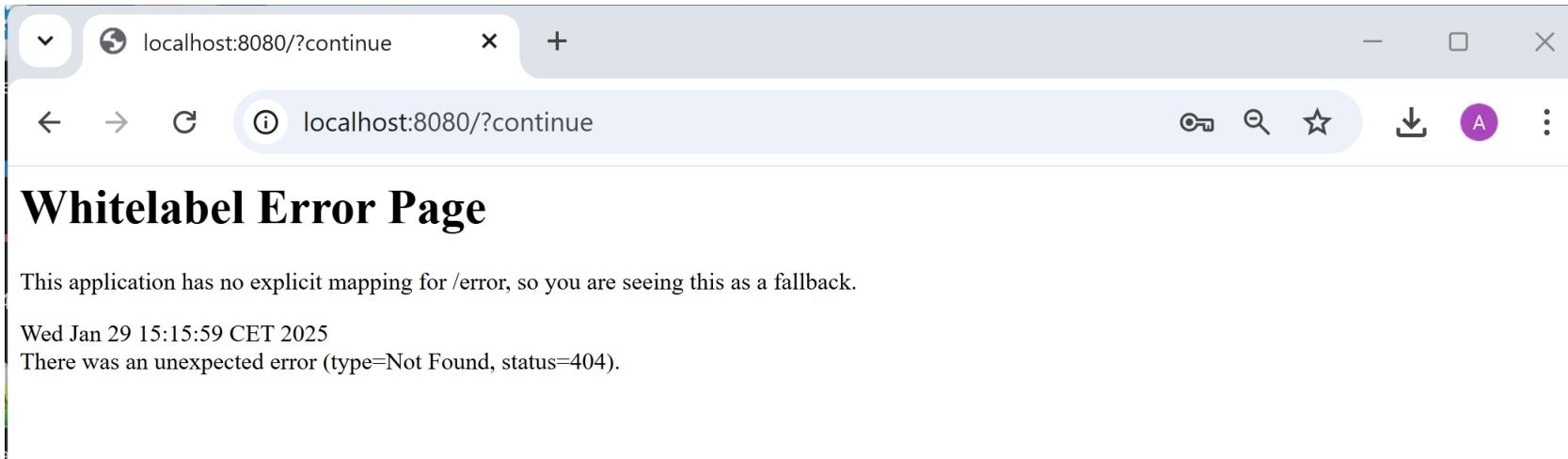
The screenshot shows the IntelliJ IDEA interface in the 'Debug' perspective, with the project 'DemoApplication' selected. The central area displays the application's log output. A red box highlights a specific log entry:

```
2025-01-29T15:07:44.899+01:00  WARN 15816 --- [demo] [main] .s.s.UserDetailsServiceAutoConfiguration :  
Using generated security password: 521d1c42-dc3e-4a56-bf85-4f03701350ec  
This generated password is for development use only. Your security configuration must be updated before running your application in production.  
2025-01-29T15:07:44.923+01:00  INFO 15816 --- [demo] [main] r$InitializeUserDetailsManagerConfigurer : Global AuthenticationManager configu
```

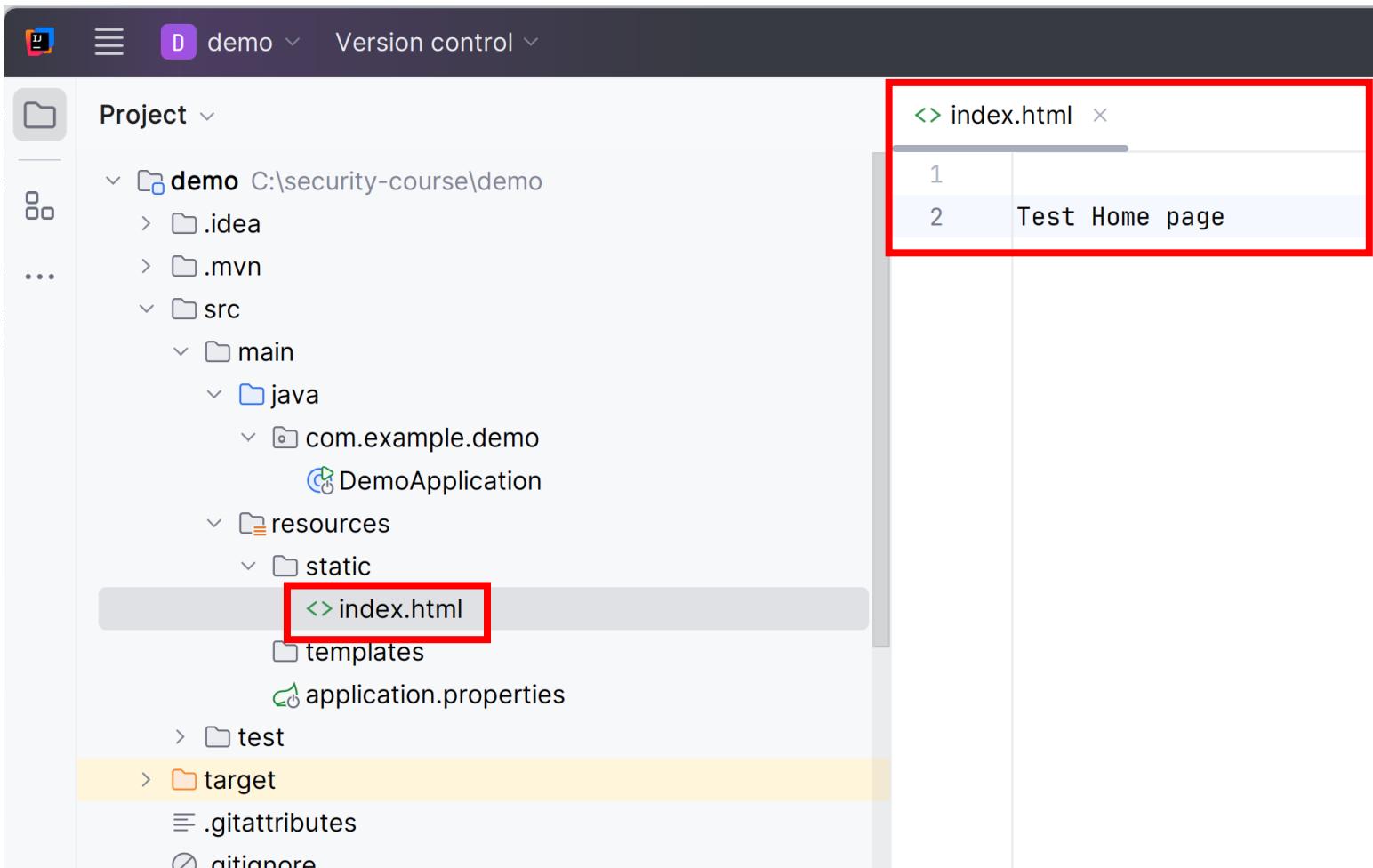
Testing user / generated password



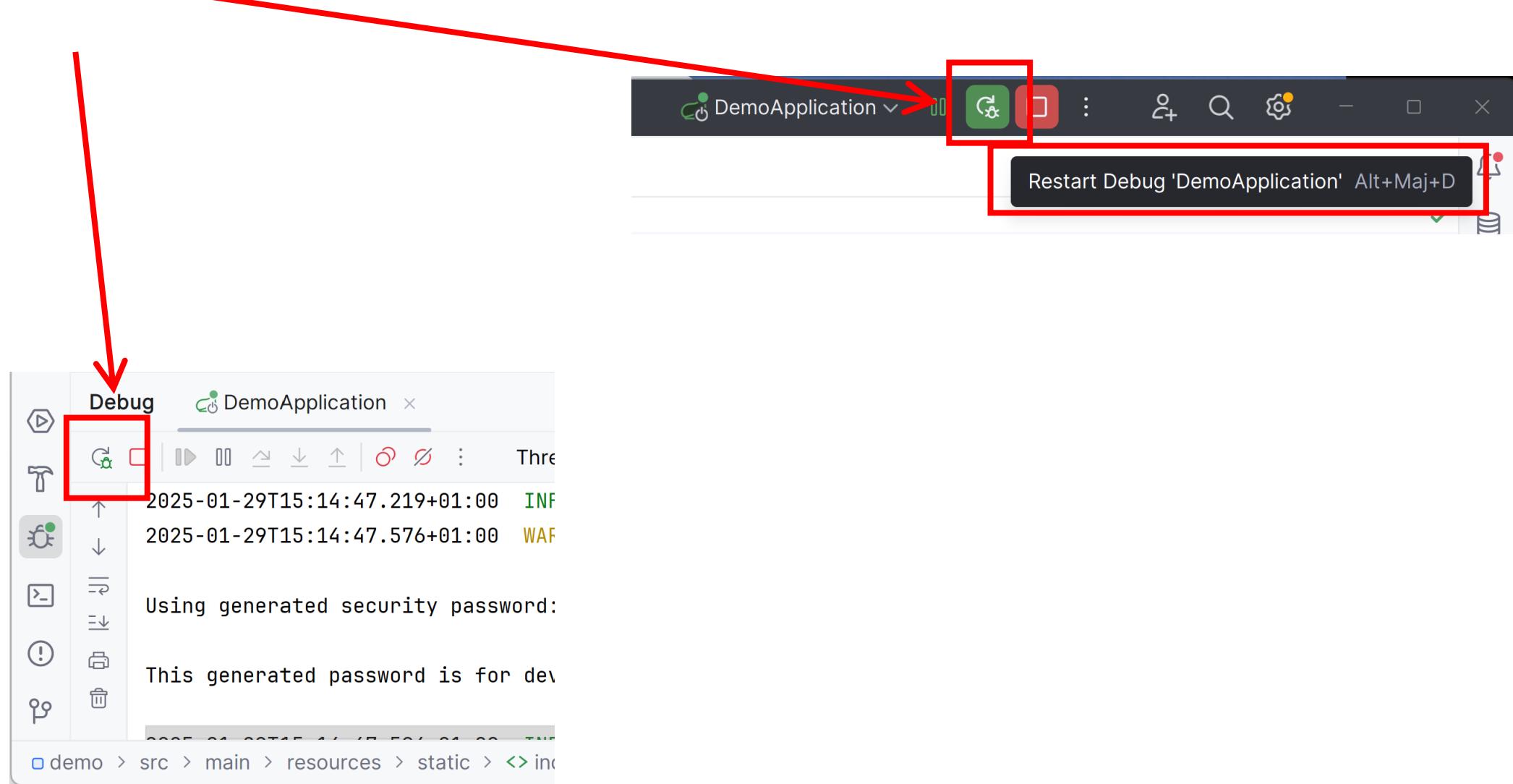
no default page exists yet... => 404
This is OK (see next slide for "index.html")



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



Relaunch

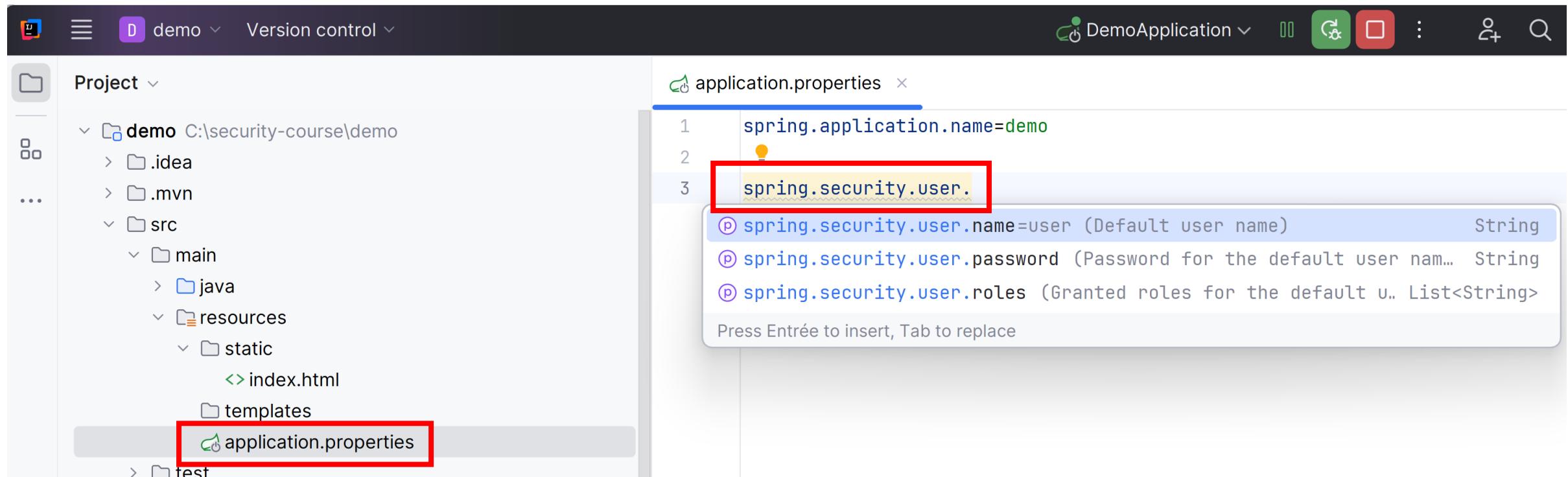


Retry ...
Resfresh (F5) Web page
login user / (new) generated password



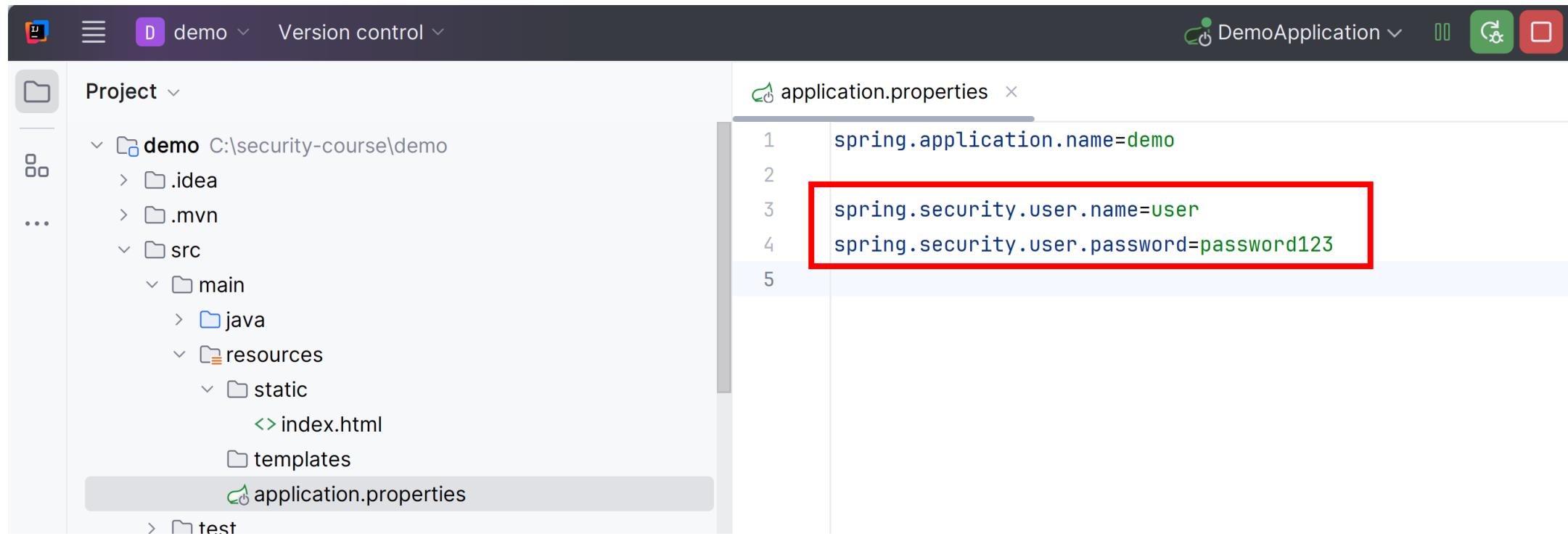
Edit file

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...
```



The screenshot shows the IntelliJ IDEA interface with a project named "demo" open. The left sidebar displays the project structure, including a .idea folder, .mvn folder, and a src directory containing main, java, resources, static, templates, and application.properties. The application.properties file is currently selected and shown in the main editor area. The file contains the following content:

```
spring.application.name=demo  
spring.security.user.name=user  
spring.security.user.password=password123
```

The lines "spring.security.user.name=user" and "spring.security.user.password=password123" are highlighted with a red rectangular box.

Relaunch + Refresh web page
Notice ... no more generated password

The screenshot shows the IntelliJ IDEA Debug tool window for a project named "DemoApplication". The "Console" tab is selected, displaying the application's startup logs. The logs indicate the application is using Spring Boot v3.4.2 and is starting up using Java 20. It shows the configuration of a Tomcat web server and the initialization of a Spring embedded WebApplicationContext. A specific log entry highlights the configuration of a Global AuthenticationManager.

```
2025-01-29T15:28:44.686+01:00 INFO 856 --- [demo] [main] com.example.demo.DemoApplication : Starting DemoApplication using Java 20
2025-01-29T15:28:44.693+01:00 INFO 856 --- [demo] [main] com.example.demo.DemoApplication : No active profile set, falling back to default profiles
2025-01-29T15:28:45.673+01:00 INFO 856 --- [demo] [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port 8080 (http)
2025-01-29T15:28:45.687+01:00 INFO 856 --- [demo] [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2025-01-29T15:28:45.688+01:00 INFO 856 --- [demo] [main] o.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/10.1.10]
2025-01-29T15:28:45.734+01:00 INFO 856 --- [demo] [main] o.a.c.c.C.[Tomcat].[localhost].[] : Initializing Spring embedded WebApplication
2025-01-29T15:28:45.735+01:00 INFO 856 --- [demo] [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialized
2025-01-29T15:28:45.862+01:00 INFO 856 --- [demo] [main] o.s.b.a.w.s.WelcomePageHandlerMapping : Adding welcome page: class path resource
2025-01-29T15:28:46.192+01:00 INFO 856 --- [demo] [main] r$InitializeUserDetailsManagerConfigurer : Global AuthenticationManager configured
2025-01-29T15:28:46.465+01:00 INFO 856 --- [demo] [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port 8080 (http) with
2025-01-29T15:28:46.477+01:00 INFO 856 --- [demo] [main] com.example.demo.DemoApplication : Started DemoApplication in 2.5 seconds
```

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

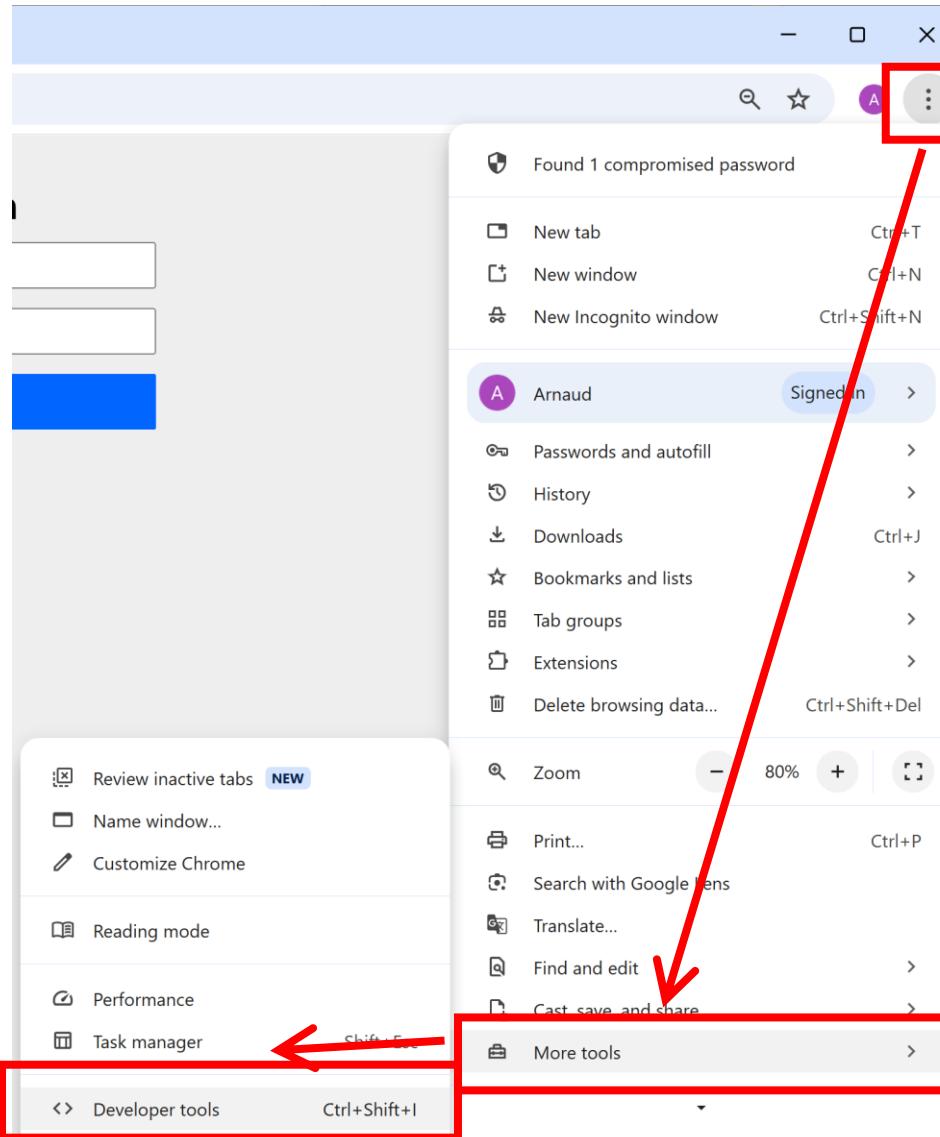
7/ test Http requests

Open Chrome "DevTools"

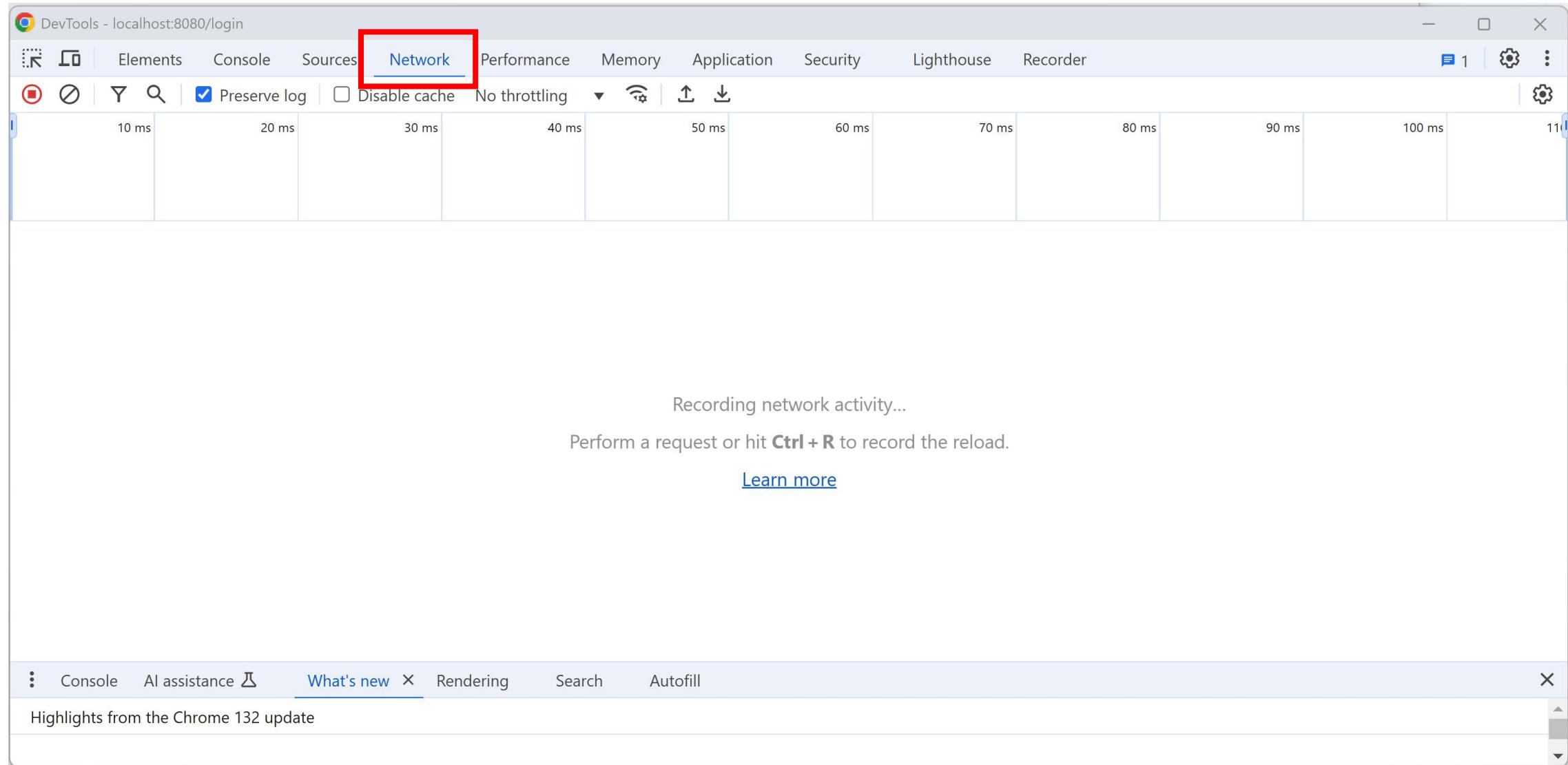
Menu > More tools > Developer tools

or F12

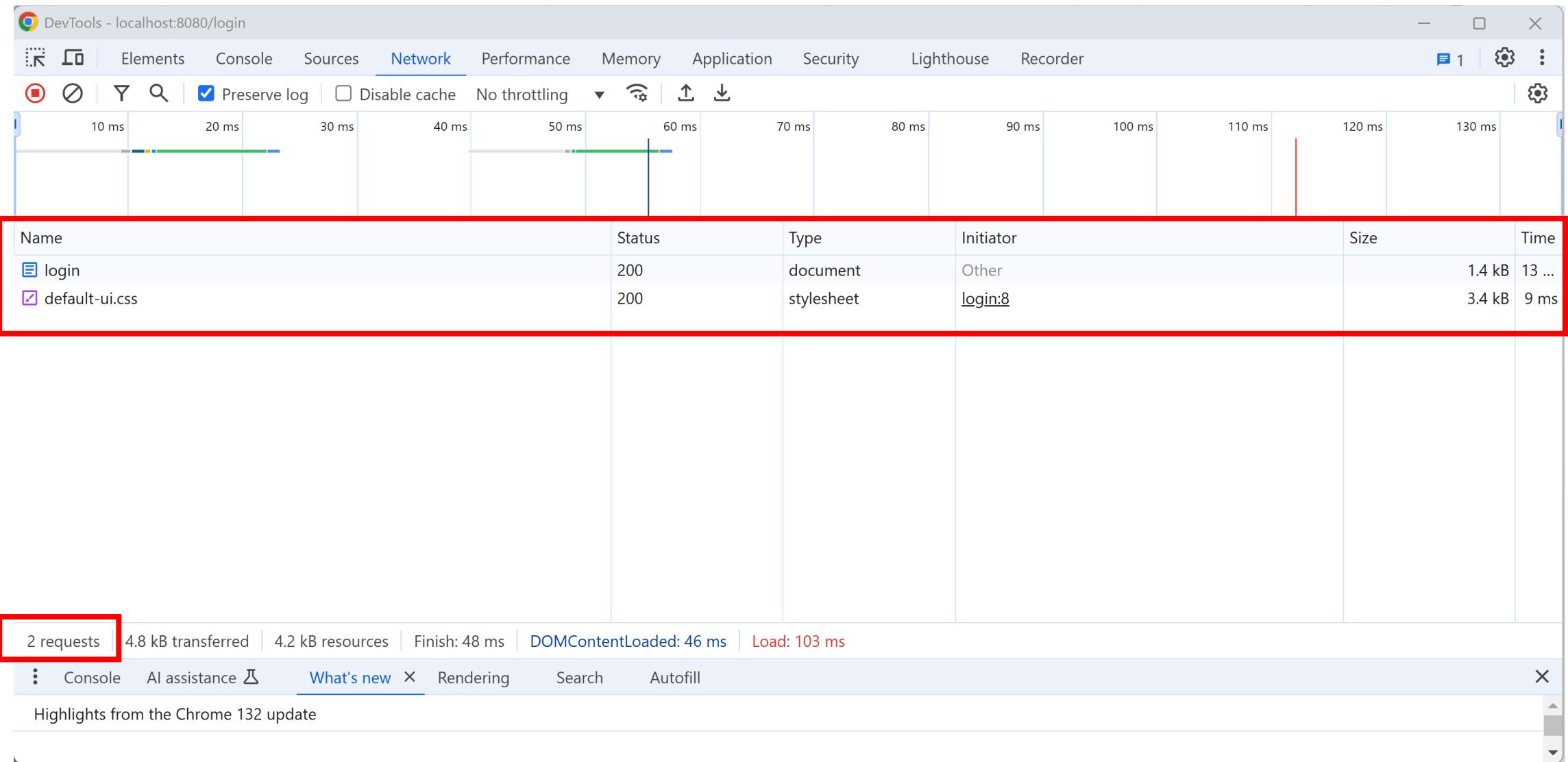
or Ctrl+Shift+I



DevTool > tab "Network"



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



The screenshot shows the Network tab in Google Chrome DevTools. At the top, there are several tabs: Elements, Console, Sources, Network (which is selected), Performance, Memory, Application, Security, Lighthouse, and Recorder. Below the tabs, there are filter options: a red square icon, a circular icon with a slash, a magnifying glass icon, a checkmark icon labeled 'Preserve log' (which is checked), and an unchecked 'Disable cache' checkbox. There's also a 'No throttling' dropdown set to 'No throttling'. The main area displays a timeline from 10 ms to 130 ms with two horizontal bars: a green bar for the 'login' request and a blue bar for the 'default-ui.css' request. A red box highlights the table below, which lists the requests with columns for Name, Status, Type, Initiator, Size, and Time.

Name	Status	Type	Initiator	Size	Time
login	200	document	Other	1.4 kB	13 ...
default-ui.css	200	stylesheet	login:8	3.4 kB	9 ms

At the bottom of the DevTools interface, there are several status indicators: '2 requests' (highlighted with a red box), '4.8 kB transferred', '4.2 kB resources', 'Finish: 48 ms', 'DOMContentLoaded: 46 ms', and 'Load: 103 ms'. Below these are buttons for 'Console', 'AI assistance', 'What's new' (which is highlighted with a red box), 'Rendering', 'Search', and 'Autofill'. A message 'Highlights from the Chrome 132 update' is also visible.

Detail Request 1 "login"

The screenshot shows the Chrome DevTools Network tab with a single request listed: "login". A red box highlights the "login" entry in the list, and a red arrow points from it to the expanded details panel on the right.

Name

- login
- default-ui.css

Headers

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

Response Headers

Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Connection: keep-alive
Content-Length: 1103
Content-Type: text/html; charset=UTF-8
Date: Wed, 29 Jan 2025 17:01:27 GMT
Expires: 0
Keep-Alive: timeout=60

2 requests | 4.8 kB transferred | 4.2 kB resources | Fin

Console AI assistance What's new Rendering Search Autofill

Highlights from the Chrome 132 update

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

Please sign in

Sign in

The screenshot shows the Chrome DevTools Network tab with three requests listed:

- Request 1: Name: login, Request URL: http://localhost:8080/login, Method: POST, Status Code: 302 Found, Response URL: http://localhost:8080/?continue.
- Request 2: Name: default-ui.css, Request URL: http://localhost:8080/default-ui.css, Method: GET, Status Code: 200 OK, Response URL: http://localhost:8080/default-ui.css.
- Request 3: Name: login, Request URL: http://localhost:8080/login, Method: POST, Status Code: 302 Found, Response URL: http://localhost:8080/?continue.

A red arrow points from the 'Sign in' button on the left to the third request in the Network tab. Another red box highlights the 'login' entry in the list, and a red arrow points from it to the detailed view on the right.

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

The screenshot shows the Network tab of a browser developer tools interface. The 'Payload' tab is selected, highlighted with a red box. A red arrow points from the 'Payload' tab to the 'Form Data' section. The 'Form Data' section contains the following entries:

- username: user
- password: password123
- _csrf: GxieqWLyzGh-nuy8iH8JDKh1QzGoalHASHVwCcerVTbdbme3fXr8kQDDUVlTr9_Zv1I9bZhGbgjKXzDtf0AWP_GdNA7tXlaE

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

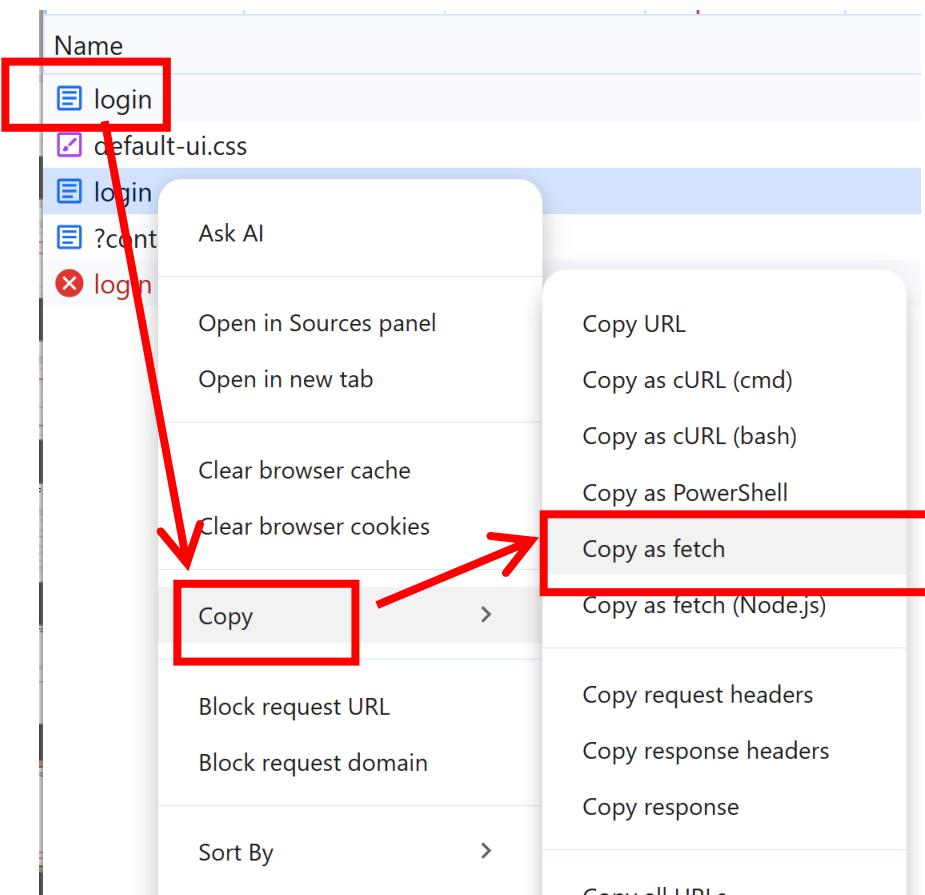
5/ open http://localhost:8080

check app is a secured http webapp, with login/password dialog

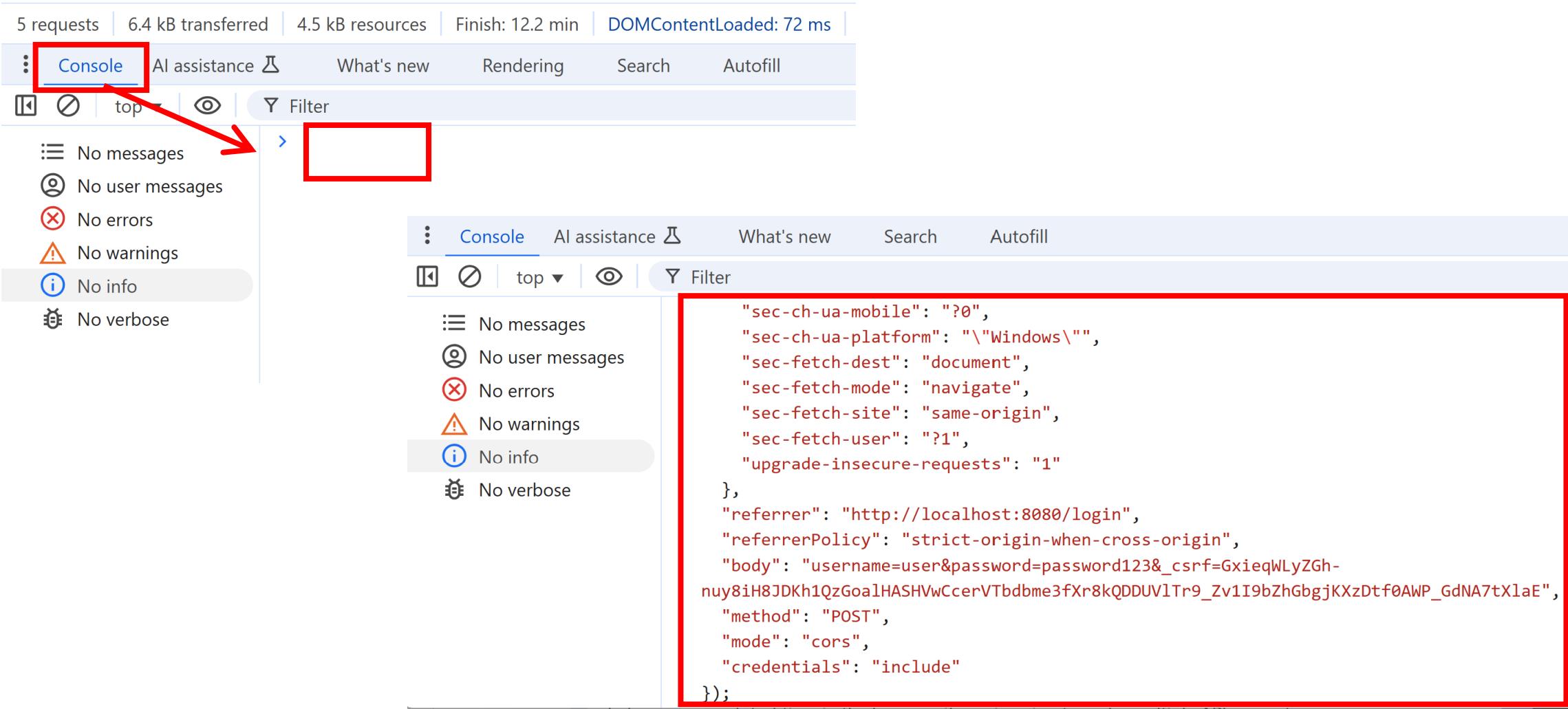
6/ inspect network requests

7/ test Http requests

Copy Http Request from DevTools



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



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

The screenshot shows the Google DevTools Console interface for a browser window titled "DevTools - localhost:8080/?continue". The "Console" tab is selected. On the left, a sidebar displays message logs: 1 message (No user messages), 1 error (No warnings), 0 info, and 0 verbose. The main area shows a stack trace of a promise chain:

```
> fetch("http://localhost:8080/login", {
  "headers": {
    "accept": "text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.7",
    "accept-language": "en,fr-FR;q=0.9,fr;q=0.8,en-FR;q=0.7,en-US;q=0.6",
    "cache-control": "max-age=0",
    "content-type": "application/x-www-form-urlencoded",
    "sec-ch-ua": "\"Not A(Brand\";v=\"8\", \"Chromium\";v=\"132\", \"Google Chrome\";v=\"132\")",
    "sec-ch-ua-mobile": "?0",
    "sec-ch-ua-platform": "\"Windows\"",
    "sec-fetch-dest": "document",
    "sec-fetch-mode": "navigate",
    "sec-fetch-site": "same-origin",
    "sec-fetch-user": "?1",
    "upgrade-insecure-requests": "1",
    "cookie": "shellInABox=3:101010; Idea-e451631d=643ff06d-adcc-4b8f-8ed5-91c19b5cbc8a; JSESSIONID=EE03D1212A5017E9578A319FD3C207FE",
    "Referer": "http://localhost:8080/login",
    "Referrer-Policy": "strict-origin-when-cross-origin"
  },
  "body": "username=user&password=password123_&csrf=GxieqWLyzGh-nuy8iH8JDKh1QzGoalHASHVwCcerVTbdbme3fxr8kQDDUVlTr9_Zv1I9bzHGb gjKXzDt f0AWP_GdNA7tXlaE",
  "method": "POST"
});
```

A red box highlights the promise object returned by the first `fetch` call, which is then resolved (fulfilled) into a `Response` object. Another red box highlights the `Response` object itself, showing its properties: body, bodyUsed, headers, ok, redirected, status, statusText, type, and url.

```
< - Promise {<pending>} i
  > [[Prototype]].Promise
  [[PromiseState]]: "fulfilled"
  [[PromiseResult]]: Response
    body: (...)

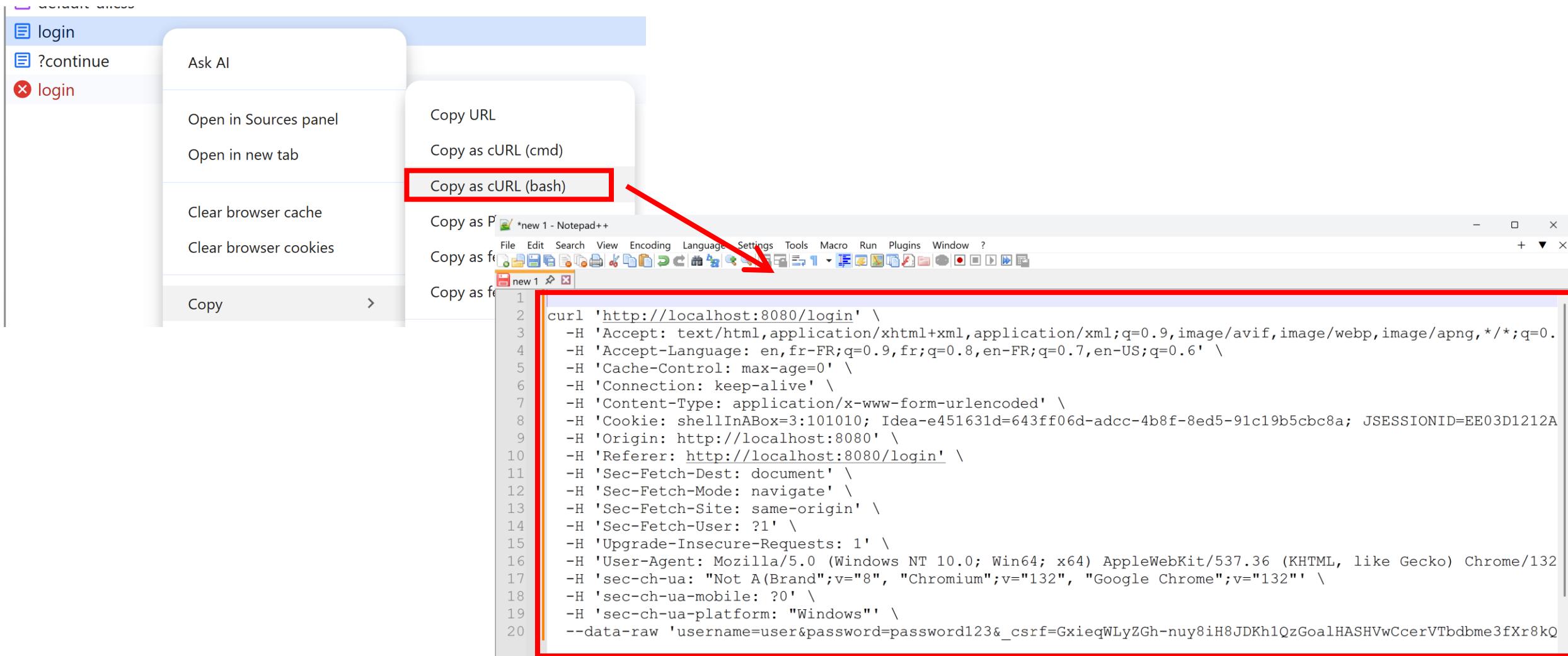
    bodyUsed: false
    > headers: Headers {}
      ok: false
      redirected: false
      status: 403
      statusText: ""
      type: "basic"
      url: "http://localhost:8080/login"
    [[Prototype]]: Response
```

At the bottom of the DevTools interface, there are tabs for "Console", "AI assistance", "What's new", "Rendering", "Search", and "Autofill".

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



The screenshot shows a Cmder terminal window with the title bar "Cmder". The command entered is:

```
arnaud@DesktopArnaud /cygdrive/c/security-course
$ curl -v 'http://localhost:8080/login' -H 'Accept: text/html' -H 'Content-Type: application/x-www-form-urlencoded' --data-raw 'username=user&password=password123&_csrf=GxieqWLyzGh-nuy8iH8JDKh1QzGoalHASHVwCcerVTbdbme3fXr8kQDDUVlTr9_Zv1I9bZhGbgjKXzDtf0AWP_GdNA7tXlaE'
```

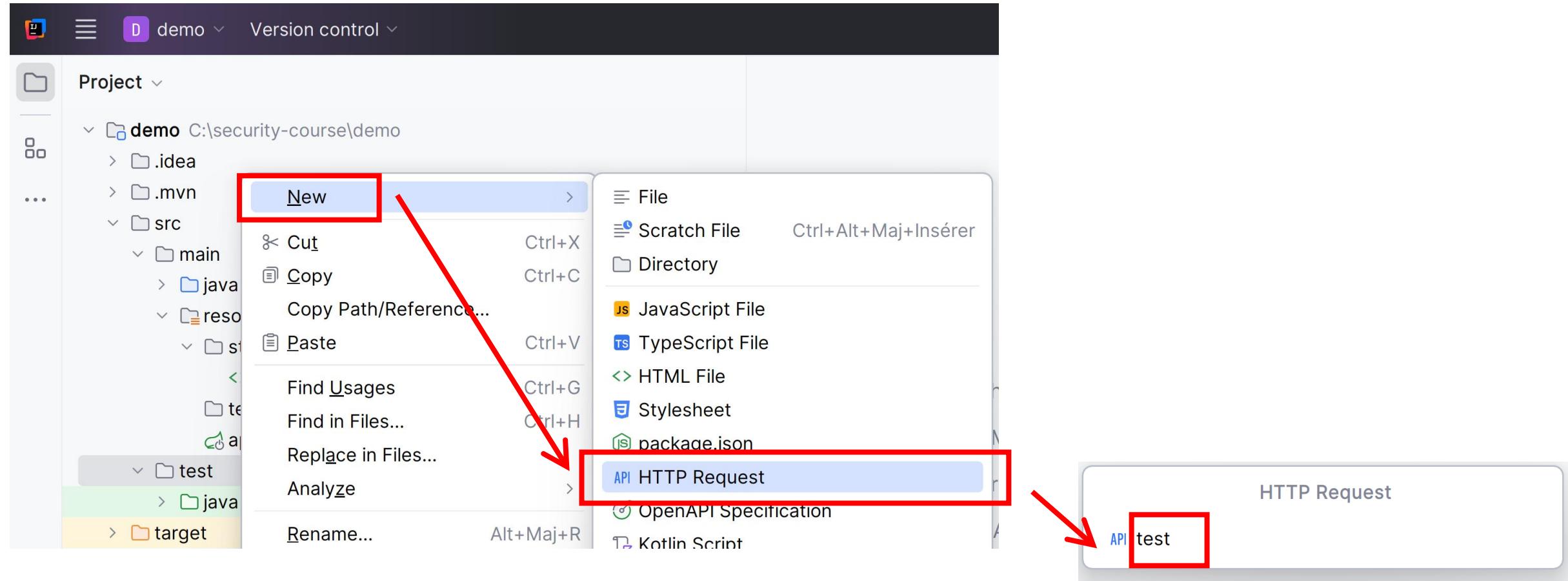
The output of the curl command is displayed below the command line, showing the verbose log of the request and response. The response includes a Set-Cookie header for JSESSIONID and various HTTP headers like X-Content-Type-Options, X-XSS-Protection, Cache-Control, Pragma, Expires, X-Frame-Options, and Location.

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

```
  Cmder
arnaud@DesktopArnaud /cygdrive/c/arn/devPerso/Presentations/security
$ curl -v -u user:password123 http://localhost:8080/index.html
* Host localhost:8080 was resolved.
* IPv6: ::1
* IPv4: 127.0.0.1
* Trying [::1]:8080...
* Connected to localhost (::1) port 8080
* using HTTP/1.1x
* Server auth using Basic with user 'user'
> GET /index.html HTTP/1.1
> Host: localhost:8080
> Authorization: Basic dXNlcjpwYXNzd29yZDEyMw==
> User-Agent: curl/8.16.1
> Accept: */*
>
* Request completely sent off
< HTTP/1.1 200
< Vary: Origin
< Vary: Access-Control-Request-Method
< Vary: Access-Control-Request-Headers
< Last-Modified: Wed, 29 Jan 2025 14:18:28 GMT
< Accept-Ranges: bytes
< X-Content-Type-Options: nosniff
< X-XSS-Protection: 0
< Cache-Control: no-cache, no-store, max-age=0, must-revalidate
< Pragma: no-cache
< Expires: 0
< X-Frame-Options: DENY
< Content-Type: text/html
< Content-Length: 16
< Date: Wed, 29 Jan 2025 18:47:50 GMT
'
```

Other method to execute Http requests

From IntelliJ, create "test.http" file



Edit Http Requests file

```
API test.http x
+ ⏲ | ⏴ | ⏵ | ⏶ | Run with: No Environment ▾
1   ### GET request to example server
2 ▷ GET https://examples.http-client.intellij.net/get
3   ?generated-in=IntelliJ IDEA
4   ⚡
5   ###
```

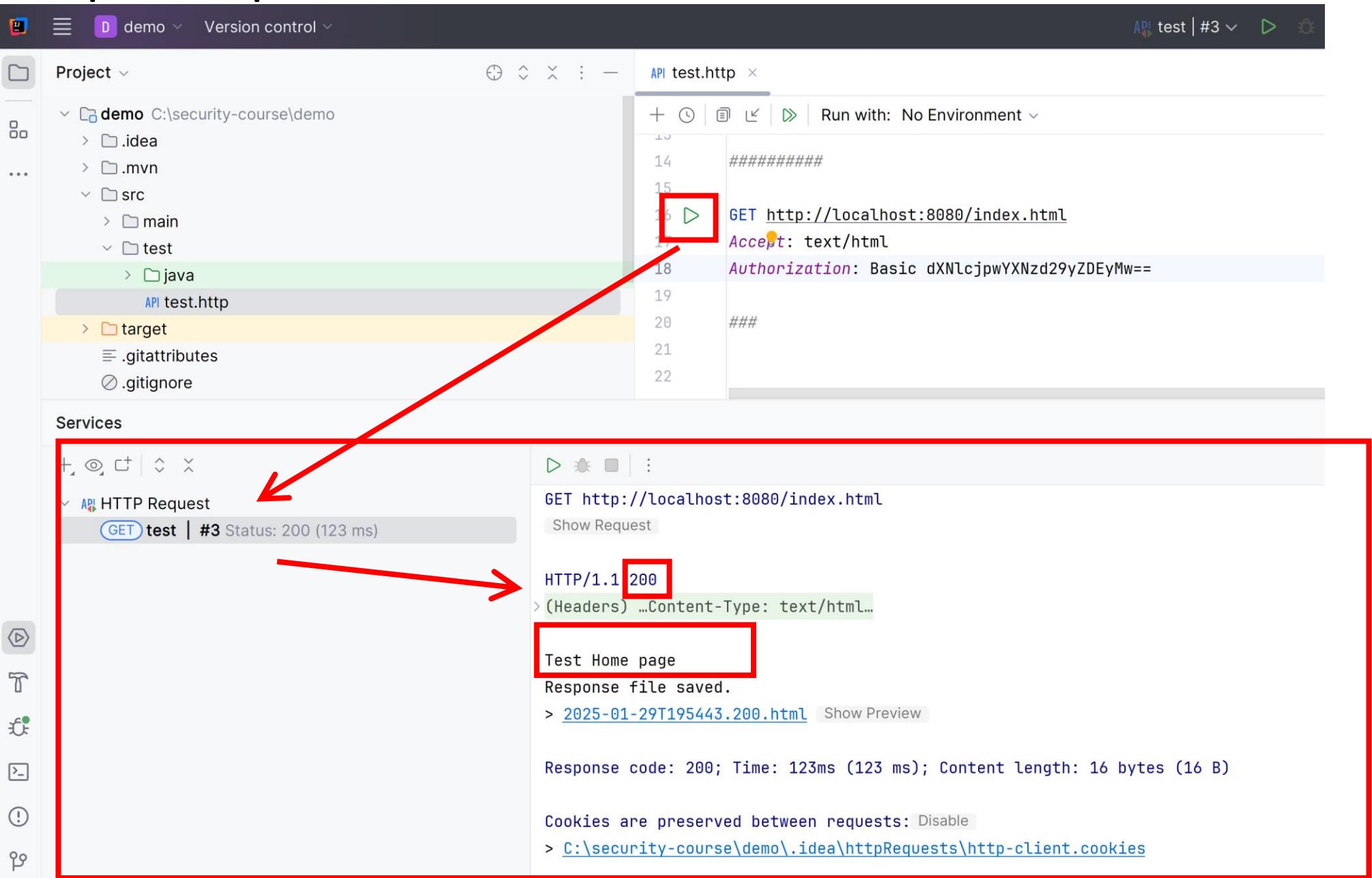


```
#####
# simple non-interactive request, with base64 user:password

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..



```
API test.http ×  
+ ⏱ ⏴ ⏵ ⏶ Run with: No Environment ▾  
1   ### GET request to example server  
2 > GET https://examples.http-client.intellij.net/get  
3   ?generated-in=IntelliJ IDEA  
4   💡  
5   ###
```

#####

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

The screenshot shows the IntelliJ IDEA interface with the following components visible:

- Project View:** On the left, it shows a project structure for "demo" located at "C:\security-course\demo". The "src" and "test" directories are expanded, with "test" currently selected.
- Code Editor:** The main editor window displays an API test script named "test.http". It contains two requests:
 - Line 2: GET https://examples.http-client.intellij.net/get ?generated-in=IntelliJ IDEA
 - Line 7: POST http://localhost:8080/login
- Services Panel:** At the bottom, there is a "Services" panel with a red border around its title bar and content area. It shows a history of API requests. One entry is highlighted: "POST test | #2 Status: 200 (322 ms)".
- Request Details:** To the right of the Services panel, a detailed view of the most recent POST request is shown:
 - Request:** POST http://localhost:8080/login
 - Headers:** Accept: Accept: text/html, Content-Type: application/x-www-form-urlencoded
 - Body:** username=user&password=password123&_csrf=GxieqWLyzGh-nuy8ih8JDKh1QzGoalHASHVwCcerVTbdbme3fXr8kQDDUVL
 - Redirections:** Redirections are disabled, showing two entries: "http://localhost:8080/login;jsessionid=B4F10916819A500A196AAFD8B0F2594B" and "http://localhost:8080/login".
 - Response:** HTTP/1.1 200, Headers: Content-Type: text/html; charset=UTF-8...

Two red arrows point from the "Services" panel to the "Redirections" section of the request details, highlighting the connection between the recorded request and the configuration of redirections.

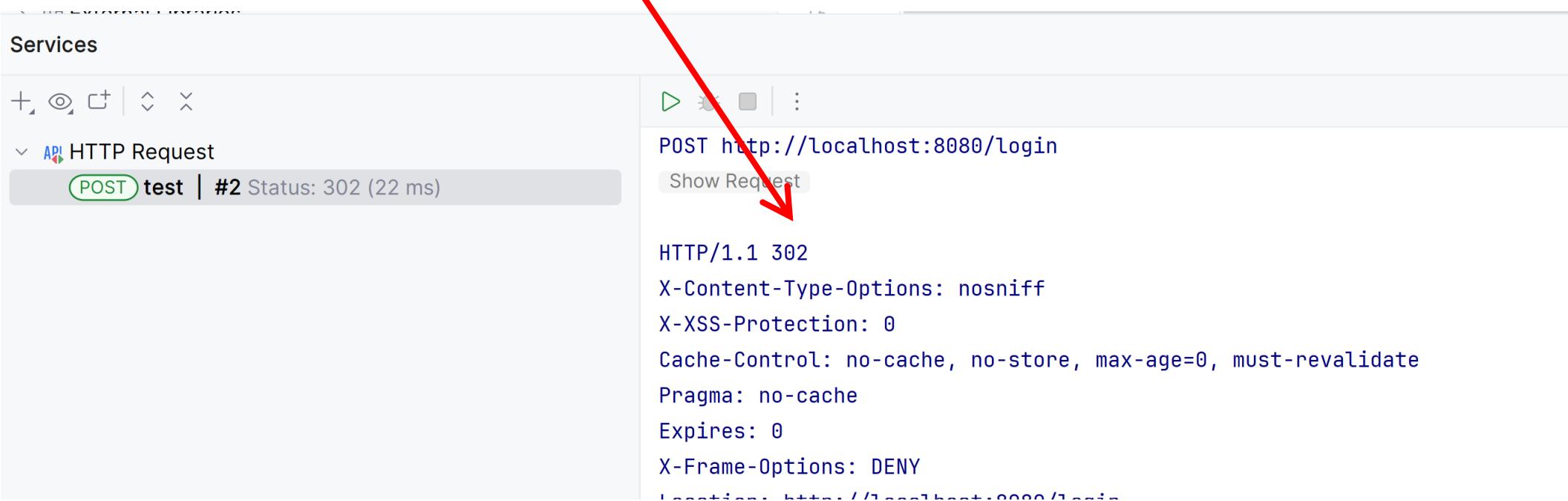
Run with "Disable redirection"

```
POST http://localhost:8080/login
Show Request

Redirections: Disable → http://localhost:8080/login

HTTP/1.1 200
> (Headers) ...Content-Type: text/html; charset=UTF-8...
```

```
5     ###
6     ⚡
7 # @no-redirect
8 > POST http://localhost:8080/login
9   Accent · Accent · text/html
```



Questions ?