

Security & Privacy

TD2 (Password Management)

Esilv 2025

arnaud.nauwynck@gmail.com

Outline (= Same as Course)

Authentication

Authorization - Permission

Confidentiality/Encryption

Backend-end: storing password in Database

Cryptographic "Hash" function, Salt

Who is "John The Ripper" ?

Summary

Authentication	 Transmit Password to Http
Authorization - Permission	password=>authentication=>role
Confidentiality/Encryption	password not in clear over internet
Backend-end: storing password in Database	storing password ...
Cryptographic "Hash" function, Salt	... storing hashed+salted password
Who is "John The Ripper" ?	finding password from hashed

Outline



Authentication

Authorization - Permission

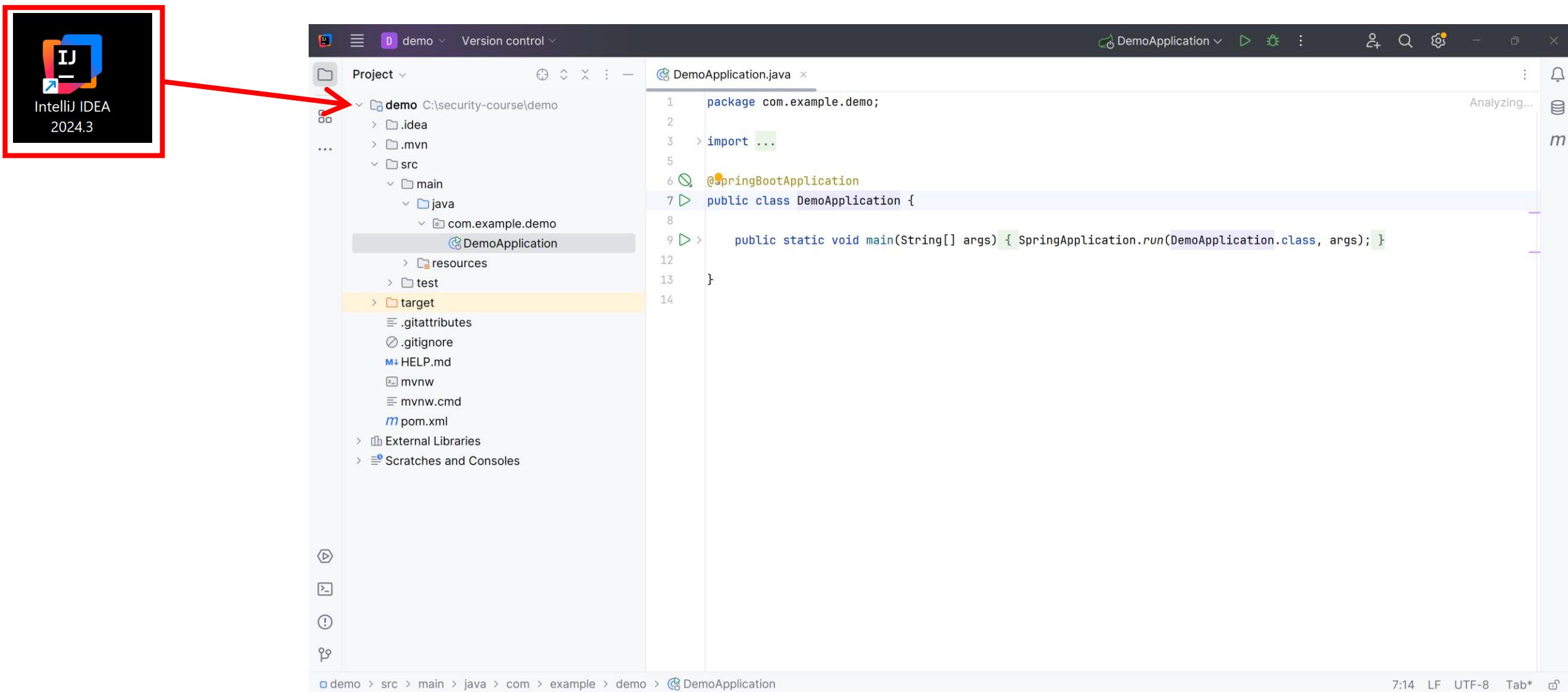
Confidentiality/Encryption

Backend-end: storing password in Database

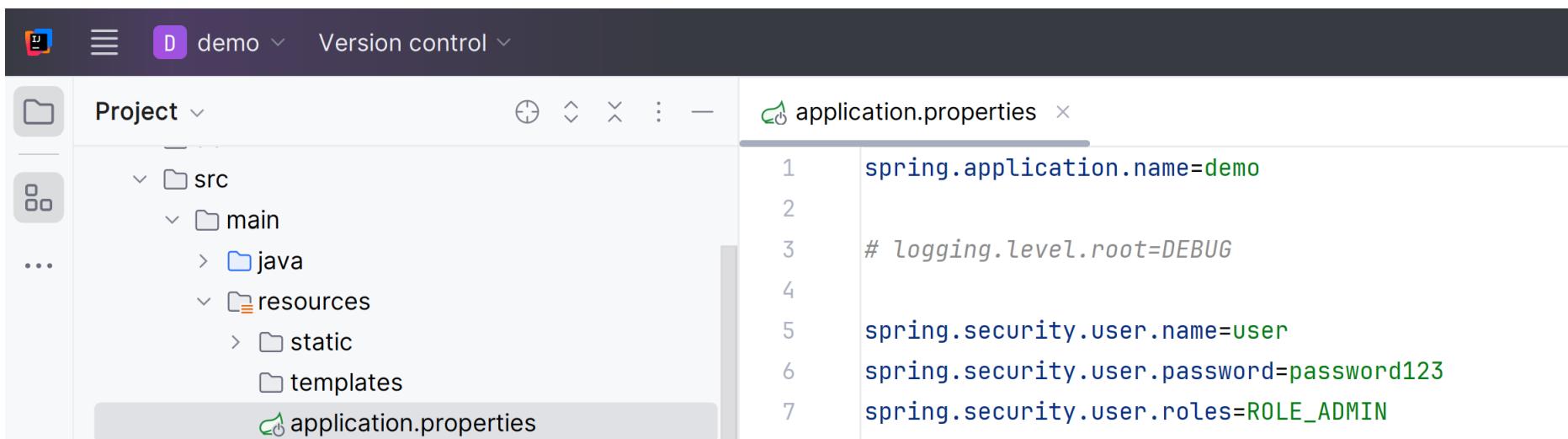
Cryptographic "Hash" function, Salt

Who is "John The Ripper" ?

Pre-Requisite Step1: relaunch your IntelliJ IDE reopen project c:\security-course\demo



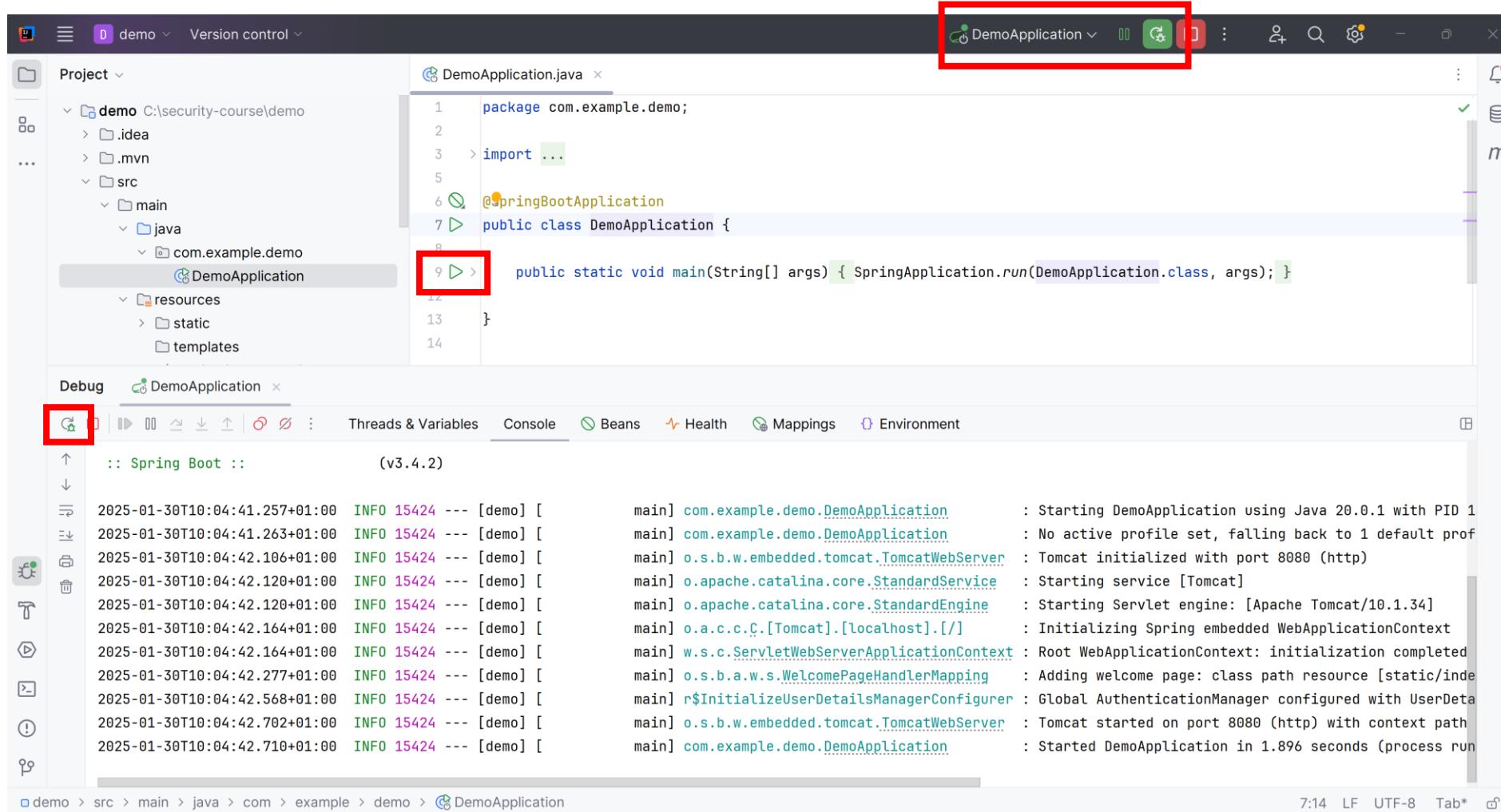
Pre-Requisite Step 2: check file
src/main/resources/application.properties
for "user" / "password123"
roles: ROLE_ADMIN



The screenshot shows a code editor interface with a dark theme. The top bar includes icons for file, edit, and version control, with the project name 'demo' selected. The left sidebar displays the project structure under 'Project': 'src' is expanded, showing 'main' (with 'java' and 'resources' subfolders), 'static', 'templates', and 'application.properties'. The 'resources' folder is also expanded. The right pane shows the contents of 'application.properties'. The code is as follows:

```
spring.application.name=demo
# logging.level.root=DEBUG
spring.security.user.name=user
spring.security.user.password=password123
spring.security.user.roles=ROLE_ADMIN
```

Pre-Requisite Step 3: relaunch webapp server



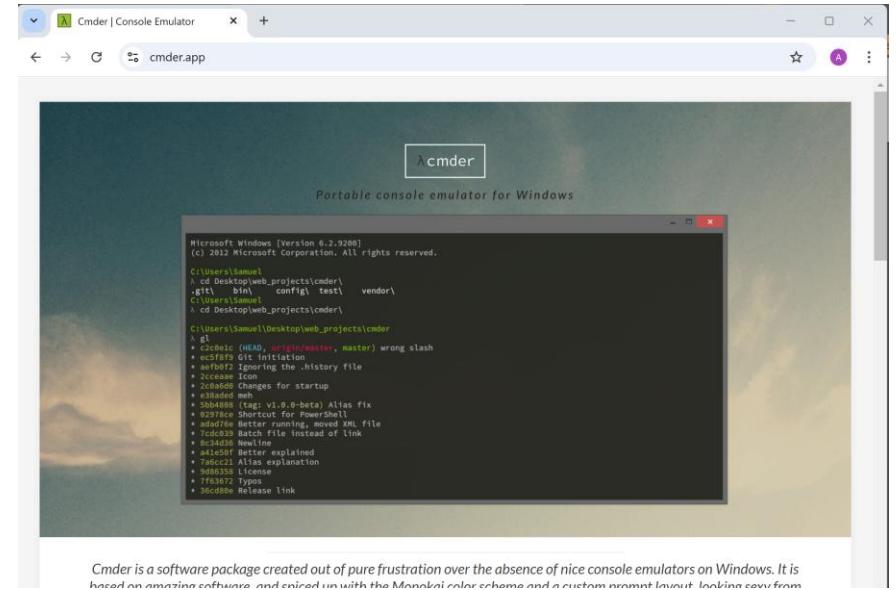
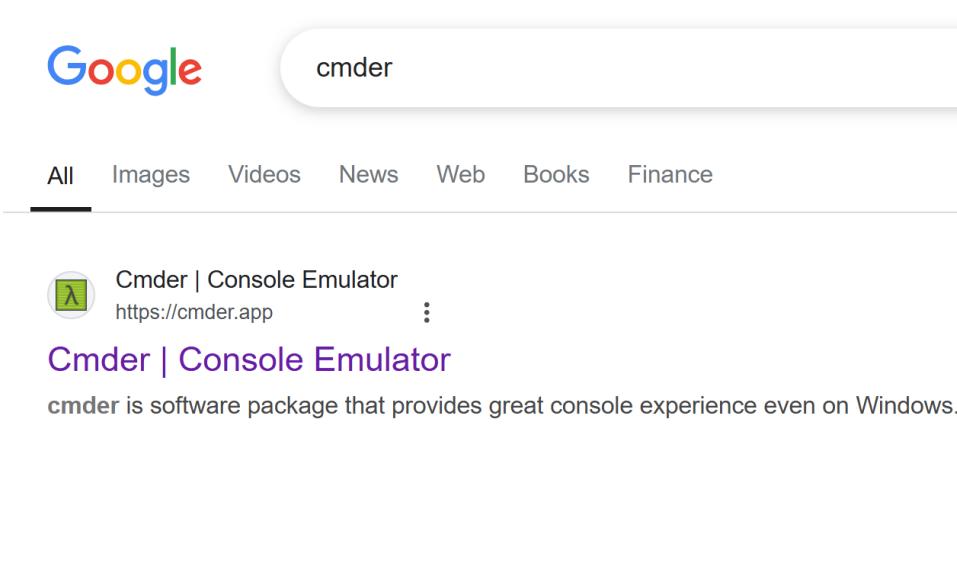
Pre-Requisite : Install (unix) shell tools
"curl" and "base64"

on linux => built-in tools on system

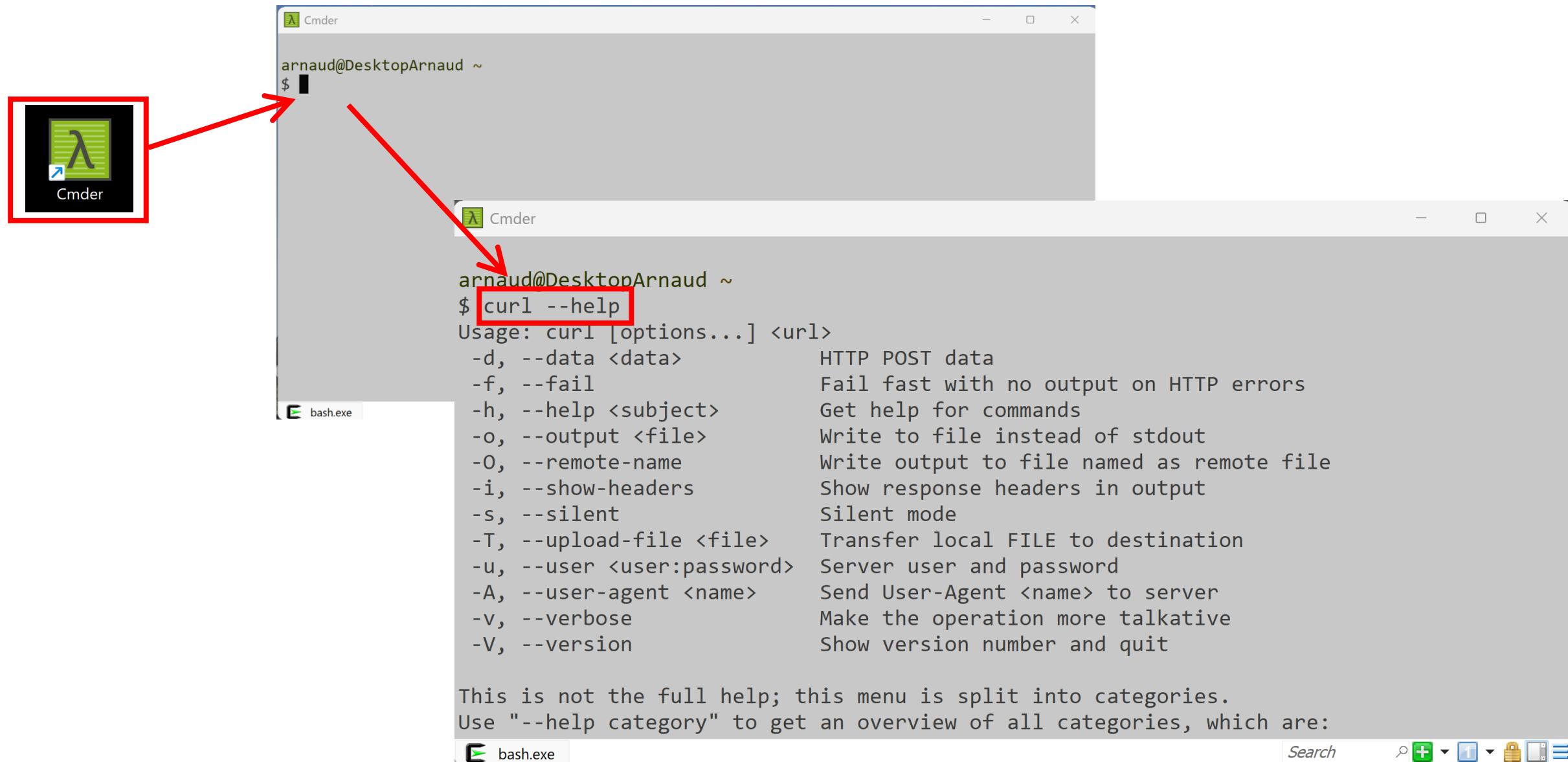
on mac => same (or use "brew install" ?)

on windows => recommended solution: install "cmder"

https://cmder.app/



Launch cmder console, test "curl --help"



curl main arguments

curl -v	<= for verbose
-k	<= accept all https certificate
-u "user:password"	<= basic user authentication
-x POST, PUT, DELETE	<= http verb
-H "header:value"	<= http header
url	<= http url
-d	<= http request body data

Test different curl requests..

`http://localhost:8080/index.html`

```
curl -v http://localhost:8080/index.html
```

```
curl -v -u user:password123 http://localhost:8080/index.html
```

```
curl -v -u user:password123 -H "accept: text/html" http://localhost:8080/index.html
```

```
curl -v -H "Cookie: JSESSIONID=..." \
      -H "accept: text/html" http://localhost:8080/index.html
```

Explain different Http response status

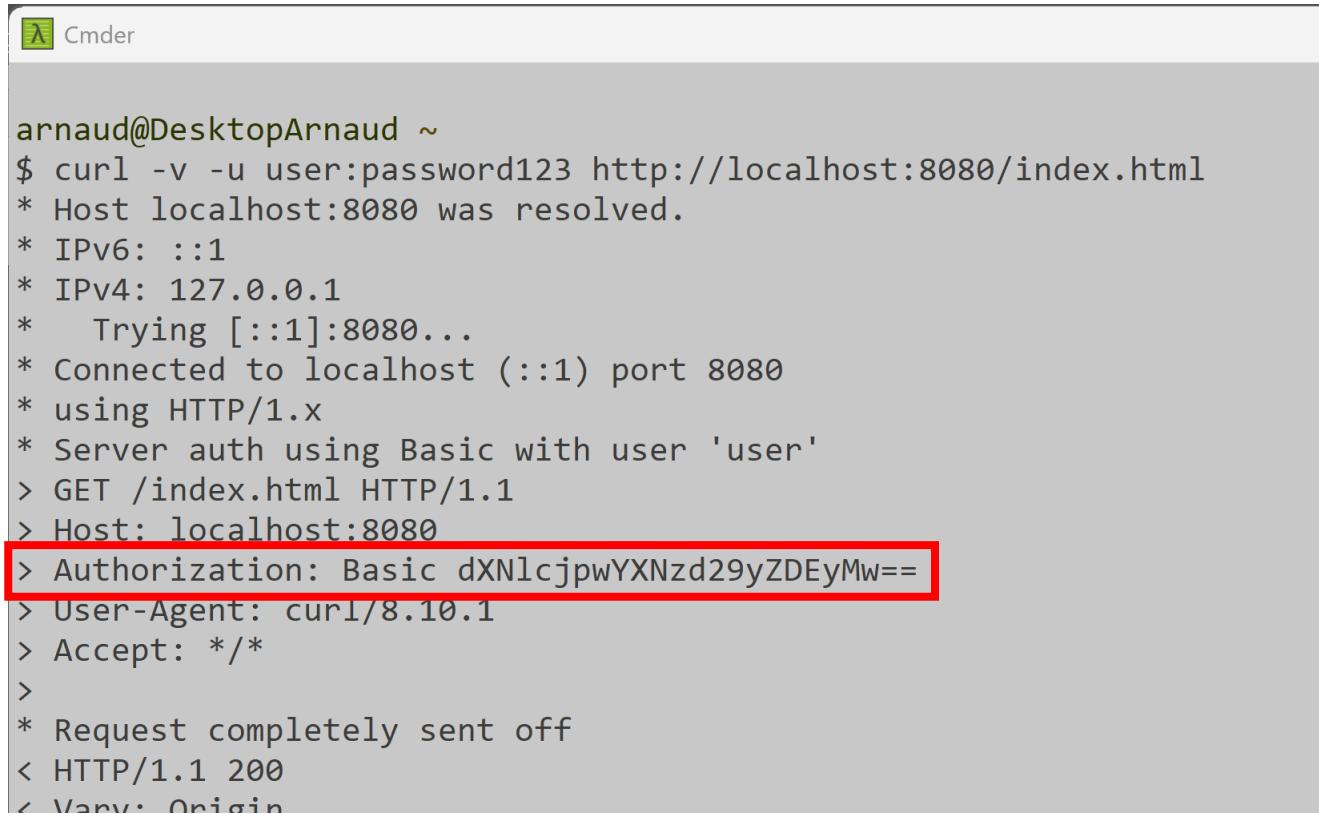
200, 301, 400, 403, 404, ...

example with http 200

```
Cmder
arnaud@DesktopArnaud ~
$ 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.x
* Server auth using Basic with user 'user'
> GET /index.html HTTP/1.1
> Host: localhost:8080
> Authorization: Basic dXNlcjpwYXNzd29yZDEyMw==
> User-Agent: curl/8.10.1
> 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: Thu, 30 Jan 2025 09:36:44 GMT
<
```



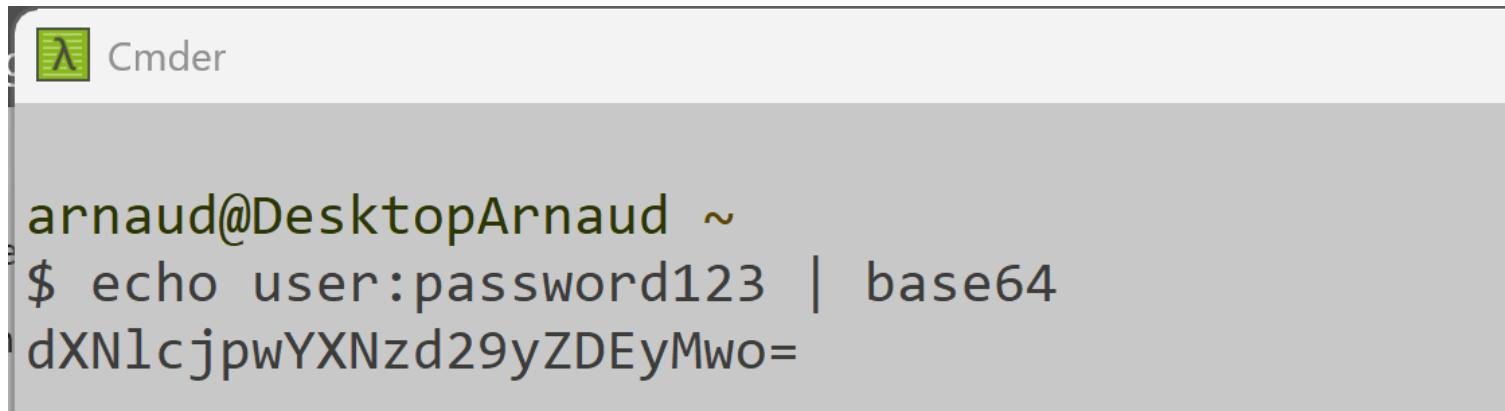
```
curl -v -u "user:password123"
=> see Http Header "Authorization: Basic ..."
```



```
arnaud@DesktopArnaud ~
$ 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.x
* Server auth using Basic with user 'user'
> GET /index.html HTTP/1.1
> Host: localhost:8080
> Authorization: Basic dXNlcjpwYXNzd29yZDEyMw==
> User-Agent: curl/8.10.1
> Accept: */*
>
* Request completely sent off
< HTTP/1.1 200
< Vary: Origin
```

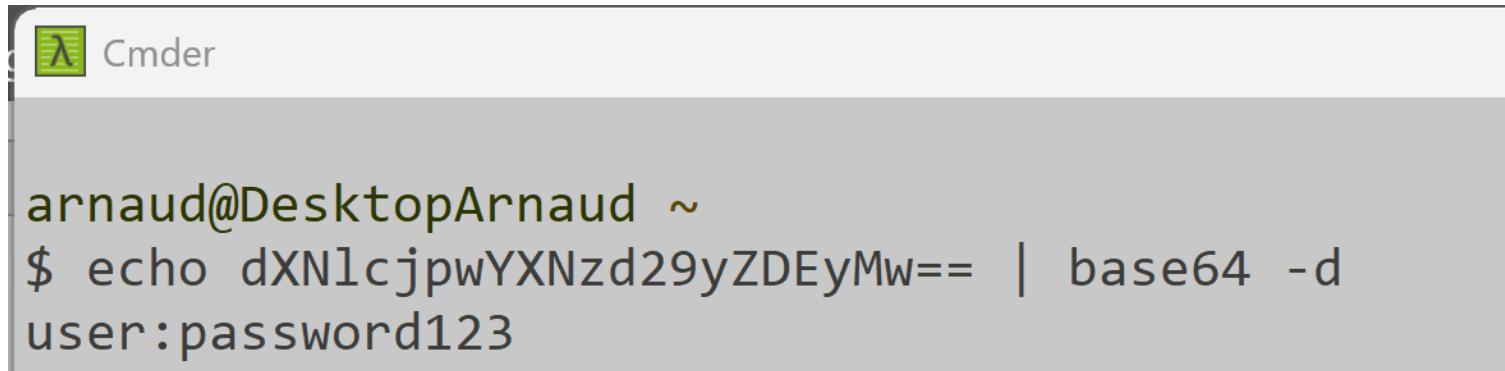
Encode / Decode base64 password, using "base64 -d"

```
echo user:password123 | base64
```



```
arnaud@DesktopArnaud ~
$ echo user:password123 | base64
dXNlcjpwYXNzd29yZDEyMwo=
```

```
echo dXNlcjpwYXNzd29yZDEyMw== | base64 -d
```



```
arnaud@DesktopArnaud ~
$ echo dXNlcjpwYXNzd29yZDEyMw== | base64 -d
user:password123
```

Test curl Basic Authorization Header (base64)

```
curl -v -H "Authorization: Basic dXNlcjpwYXNzd29yZDEyMw==" \  
      -H "accept: text/html" http://localhost:8080/index.html
```



Remember ...
When taking Http Request screenshots, or sharing screens to have IT support
... Do not reveal your "base64 encoded" password !!!

Outline

Authentication



Authorization - Permission

Confidentiality/Encryption

Backend-end: storing password in Database

Cryptographic "Hash" function, Salt

Who is "John The Ripper" ?

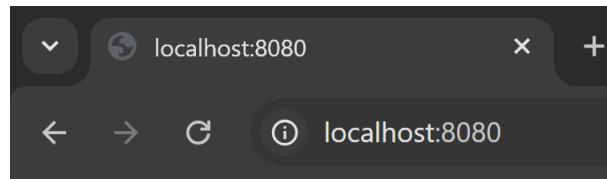
Target: pages*.html with different Roles

The screenshot shows a Java IDE interface with the following details:

- Project View:** Shows a tree structure of files and folders:
 - src
 - main
 - java
 - resources
 - static
 - index.html
 - page-admin.html
 - page-anonymous.html
 - page-authenticated.html
 - page-manager.html
 - page-user.html
 - templates
 - test
 - target
 - Structure View:** Shows a list of elements:
 - <H1> Home page
 - Code Editor:** Displays the content of the selected file, index.html:

```
1 <H1>Home page</H1>
2 <BR/>
3 <BR/>
4
5 <A href="/page-user.html">page-user</A>
6 <BR/>
7 <A href="/page-manager.html">page-manager</A>
8 <BR/>
9 <A href="/page-authenticated.html">page-authenticated</A>
10 <BR/>
11 <A href="/page-anonymous.html">page-anonymous</A>
12 <BR/>
13 <A href="/page-admin.html">page-admin</A>
14 <BR/>
15
16 <BR/>
17
18 <A href="/h2-console">h2-console</A> (ADMIN)
```

"index.html" menu with anonymous access



Home page

[page-user](#)
[page-manager](#)
[page-authenticated](#)
[page-anonymous](#)
[page-admin](#)

[h2-console](#) (ADMIN)
[logout](#)

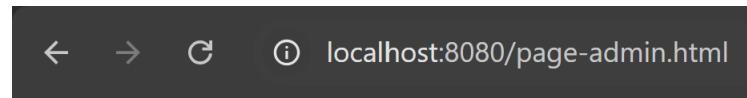
```
<H1>Home page</H1>
<BR/>
<BR/>

<A href="/page-user.html">page-user</A>
<BR/>
<A href="/page-manager.html">page-manager</A>
<BR/>
<A href="/page-authenticated.html">page-authenticated</A>
<BR/>
<A href="/page-anonymous.html">page-anonymous</A>
<BR/>
<A href="/page-admin.html">page-admin</A>
<BR/>
<BR/>

<A href="/h2-console">h2-console</A> (ADMIN)
<BR/>

<A href="/logout">logout</A>
```

Page-XYZ ... accessible only if user has role XYZ



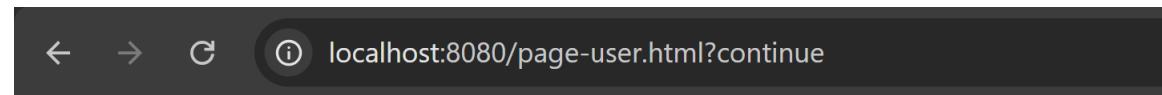
[Home](#)

... should see only if you have role "ADMIN"

<H1>Page Admin</H1>

Home

... should see only if you have role "ADMIN"



Whitelabel Error Page

This application has no explicit mapping for /error, so you are seeing this as a fallback.

Fri Jan 31 19:12:37 CET 2025

There was an unexpected error (type=Forbidden, status=403).

class SecurityConf [1/3]

to configure ... (and disable cors & csrf)

```
package com.example.demo;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.config.annotation.web.builders.HttpSecurity;
import org.springframework.security.web.SecurityFilterChain;
import org.springframework.web.cors.CorsConfiguration;
import org.springframework.web.cors.UrlBasedCorsConfigurationSource;
import org.springframework.web.filter.CorsFilter;
import java.util.List;

@Configuration
@EnableWebSecurity
public class SecurityConfig {
    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity http) throws Exception {
        return http
            .formLogin(withDefaults())
            .logout(withDefaults())
            .authorizeHttpRequests((requests) -> configureHttpAuth(requests))
            .cors(cors -> cors.configurationSource(corsConfigurationSource()))
            .csrf(csrf -> csrf
                .ignoringRequestMatchers("/h2-console/**")
                .disable()) // Disable CSRF for API requests
            .headers(headers -> headers.frameOptions(frame -> frame.sameOrigin()))
            .build();
    }
}
```

// ... continue next page

class SecurityConf [2/3]

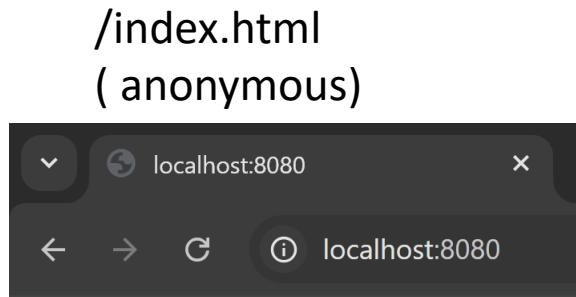
```
private static void configureHttpAuth(  
    AuthorizeHttpRequestsConfigurer<HttpSecurity>.AuthorizationManagerRequestMatcherRegistry requests) {  
    requests  
        .requestMatchers(HttpMethod.GET, "*.css", "*.js", "*.png").permitAll()  
        .requestMatchers("/", "/index.html").permitAll()  
        .requestMatchers("/page-admin.html").hasRole("ADMIN")  
        .requestMatchers("/page-user.html").hasRole("USER")  
        .requestMatchers("/page-manager.html").hasRole("MANAGER")  
        .requestMatchers("/page-anonymous.html").permitAll()  
        .requestMatchers("/page-authenticated.html").authenticated()  
        .requestMatchers("/h2-console/**")  
            .authenticated() // TODO .. should be .hasRole("ADMIN")  
            // .hasRole("ADMIN")  
        .requestMatchers("/api/**").authenticated()  
        .anyRequest().permitAll();  
}
```

class SecurityConf [3/3]

```
@Bean
public CorsFilter corsFilter() {
    CorsConfiguration config = new CorsConfiguration();
    config.setAllowCredentials(true);
    config.setAllowedOrigins(List.of("*"));
    config.setAllowedHeaders(List.of("*"));
    config.setAllowedMethods(List.of("GET", "POST", "PUT", "DELETE", "HEAD", "OPTIONS"));
    UrlBasedCorsConfigurationSource source = new UrlBasedCorsConfigurationSource();
    source.registerCorsConfiguration("/**", config);
    return new CorsFilter(source);
}

private UrlBasedCorsConfigurationSource corsConfigurationSource() {
    UrlBasedCorsConfigurationSource source = new UrlBasedCorsConfigurationSource();
    CorsConfiguration config = new CorsConfiguration();
    config.setAllowCredentials(true);
    config.setAllowedOrigins(List.of("*")); // Allow all origins
    config.setAllowedHeaders(List.of("*")); // Allow all headers
    config.setAllowedMethods(List.of("*")); // Allow all HTTP methods
    source.registerCorsConfiguration("/**", config);
    return source;
}
```

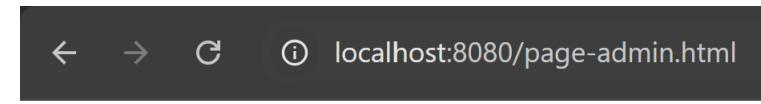
Relaunch & Test navigation



Home page

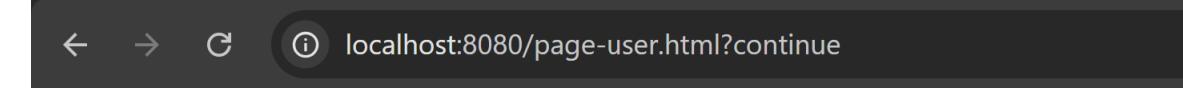
[page-user](#)
[page-manager](#)
[page-authenticated](#)
[page-anonymous](#)
[page-admin](#)

[h2-console](#) (ADMIN)
[logout](#)



Page Admin

[Home](#)
... should see only if you have role "ADMIN"

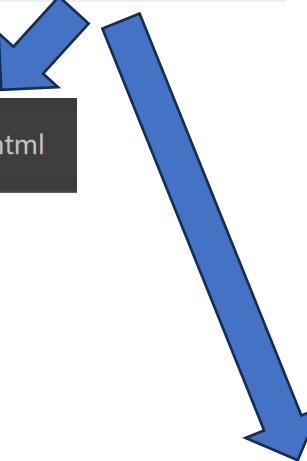
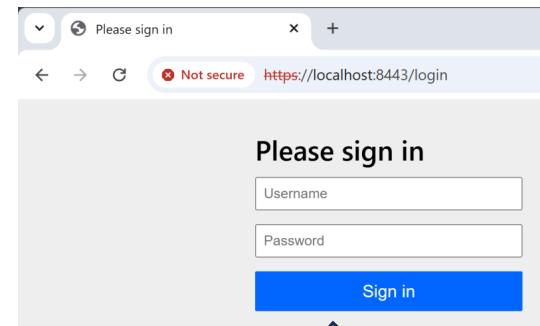


Whitelabel Error Page

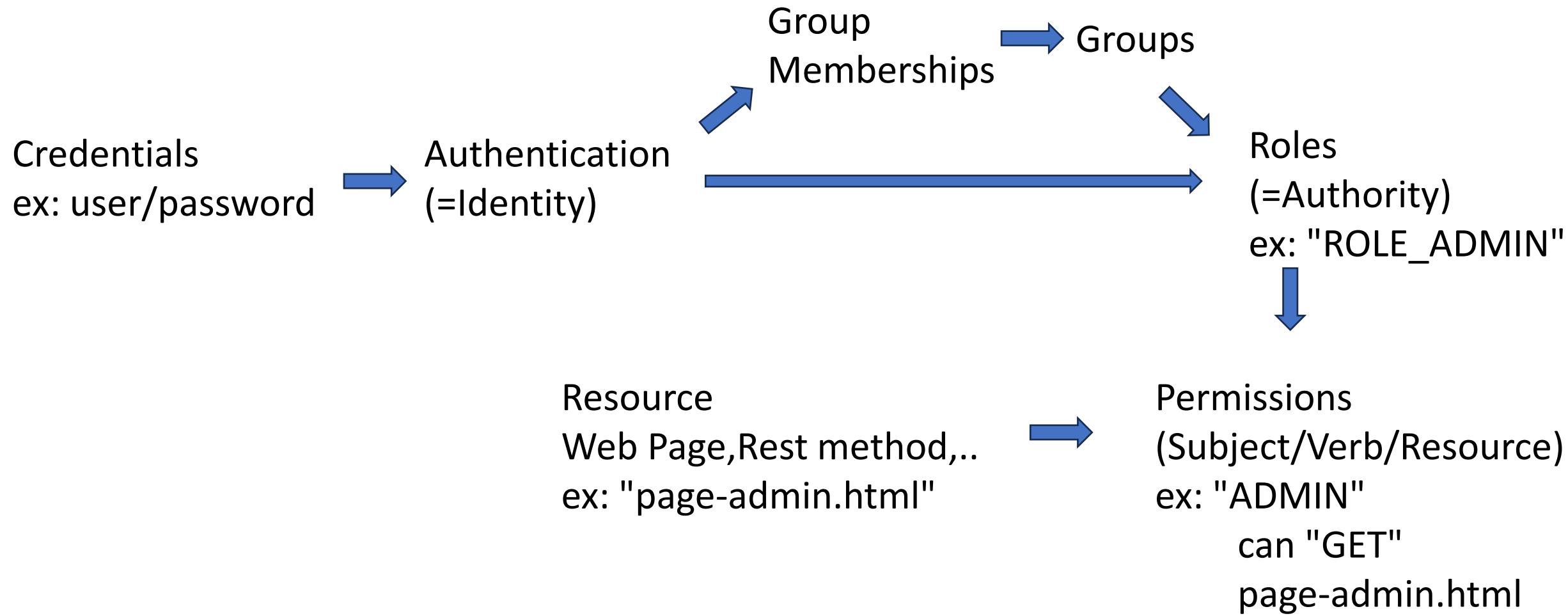
This application has no explicit mapping for /error, so you are seeing this as a fallback.

Fri Jan 31 19:12:37 CET 2025

There was an unexpected error (type=Forbidden, status=403).



Reminder..



Outline

Authentication

Authorization - Permission



Confidentiality/Encryption

Backend-end: storing password in Database

Cryptographic "Hash" function, Salt

Who is "John The Ripper" ?

Generate your Https self-signed certificate

```
keytool -genkeypair -alias mycert -keyalg RSA -keysize 2048 -storetype PKCS12 -keystore mycert.p12  
-validity 3650
```

Troubleshooting ?

"keytool" is a tool in your jdk/bin directory

check "where keytool" => it should be found in your "PATH" environment variable

check "echo \$PATH"

... to fix, use windows "env" to edit your environement variables, or export PATH="\$PATH:\$JAVA_HOME/bin"



```
arnaud@DesktopArnaud /cygdrive/c/security-course/demo  
$ where keytool  
C:\apps\jdk\jdk-20.0.1+9\bin\keytool.exe
```

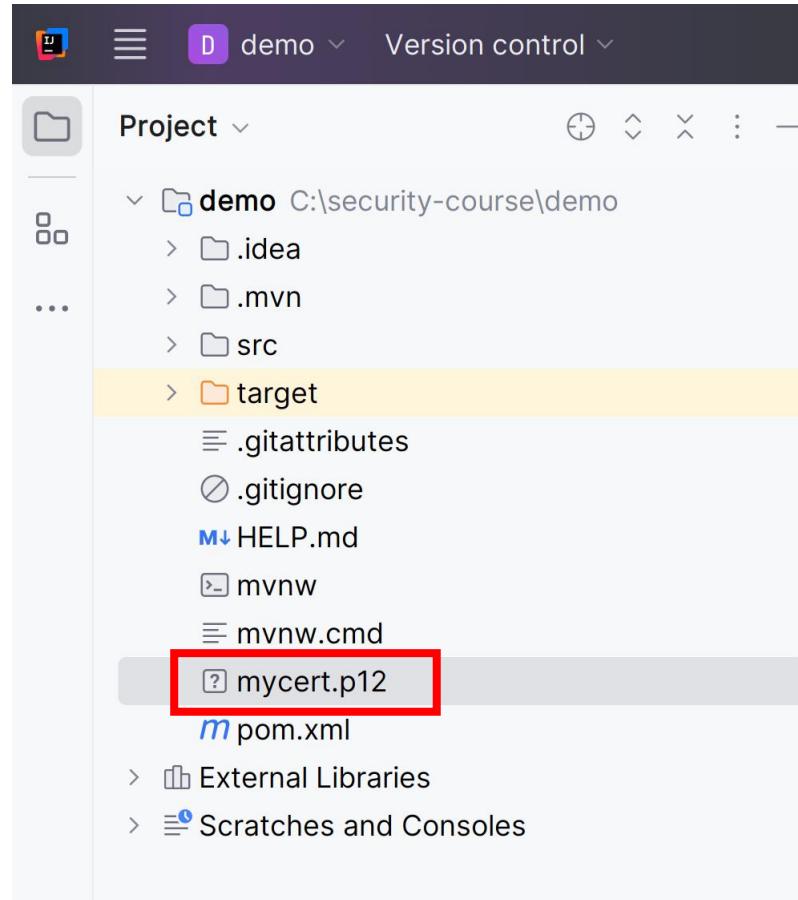
interactive answer to prompt (any is ok) lastly "Is it CN=.... ? " [no]: type yes !!

```
  Cmder
Quel est le nom de votre entreprise ?
[Unknown]:
Quel est le nom de votre ville de résidence ?
[Unknown]: Quel est le nom de votre état ou province ?
[Unknown]:
arnaud@DesktopArnaud /cygdrive/c/security-course/demo
$ keytool -genkeypair -alias mycert -keyalg RSA -keysize 2048 -storetype PKCS12 -keystore mycert.p12 -validity 3650
Entrez le mot de passe du fichier de clés :

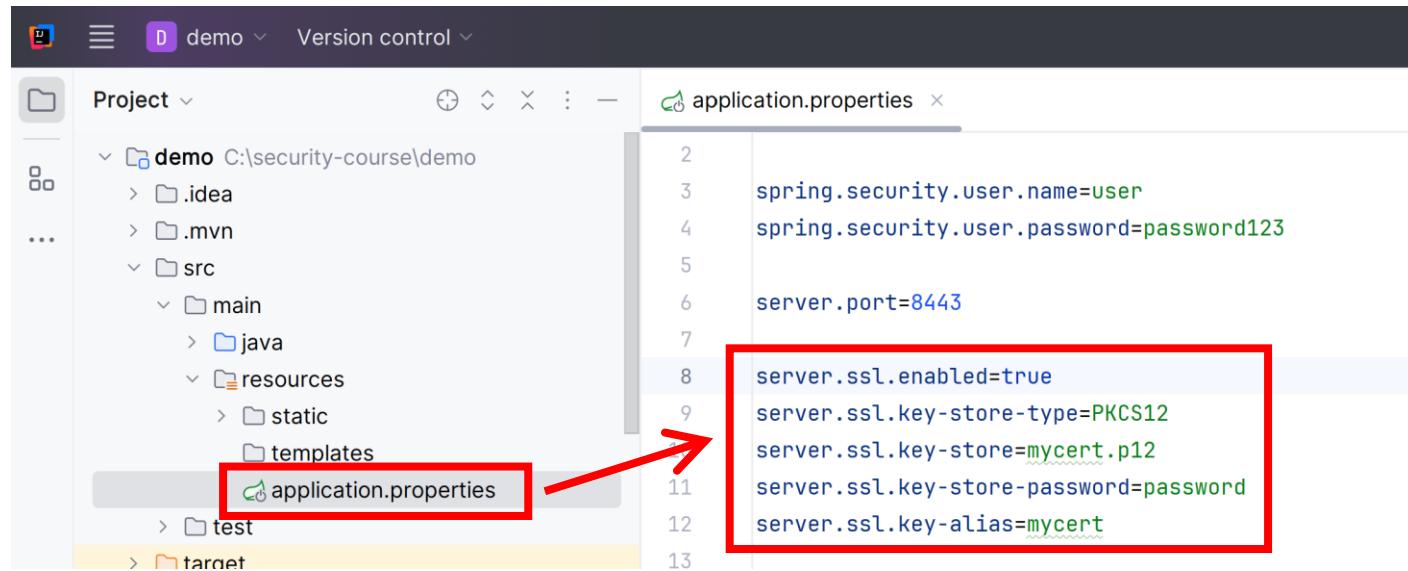
Ressaisissez le nouveau mot de passe :

Enter the distinguished name. Provide a single dot (.) to leave a sub-component empty or press
ENTER to use the default value in braces.
Quels sont vos nom et prénom ?
[Unknown]: Nauwynck Arnaud
Quel est le nom de votre unité organisationnelle ?
[Unknown]: esilv
Quel est le nom de votre entreprise ?
[Unknown]: sg
Quel est le nom de votre ville de résidence ?
[Unknown]: nanterre
Quel est le nom de votre état ou province ?
[Unknown]: fr
Quel est le code pays à deux lettres pour cette unité ?
[Unknown]: fr
Est-ce CN=Nauwynck Arnaud, OU=esilv, O=sg, L=nanterre, ST=fr, C=fr ?
[non]: oui
Génération d'une paire de clés RSA de 2048 bits et d'un certificat auto-signé (SHA384withRSA)
d'une validité de 3650 jours
pour : CN=Nauwynck Arnaud, OU=esilv, O=sg, L=nanterre, ST=fr, C=fr
```

Check created file "mycert.p12" in project
(or move it in your project)



Edit src/main/resources/application.properties to configure SSL



The screenshot shows a Java project structure in a code editor. The project is named 'demo' and contains a 'src' folder with 'main' and 'test' subfolders. Inside 'main', there are 'java', 'resources', 'static', and 'templates' folders. The 'application.properties' file is located in the 'resources' folder. A red box highlights the 'application.properties' file in the file tree. A red arrow points from this highlighted file to the content of the file in the main editor area. The file content is as follows:

```
spring.security.user.name=user
spring.security.user.password=password123
server.port=8443
server.ssl.enabled=true
server.ssl.key-store-type=PKCS12
server.ssl.key-store=mycert.p12
server.ssl.key-store-password=password
server.ssl.key-alias=mycert
```

server.port=8443

server.ssl.enabled=true

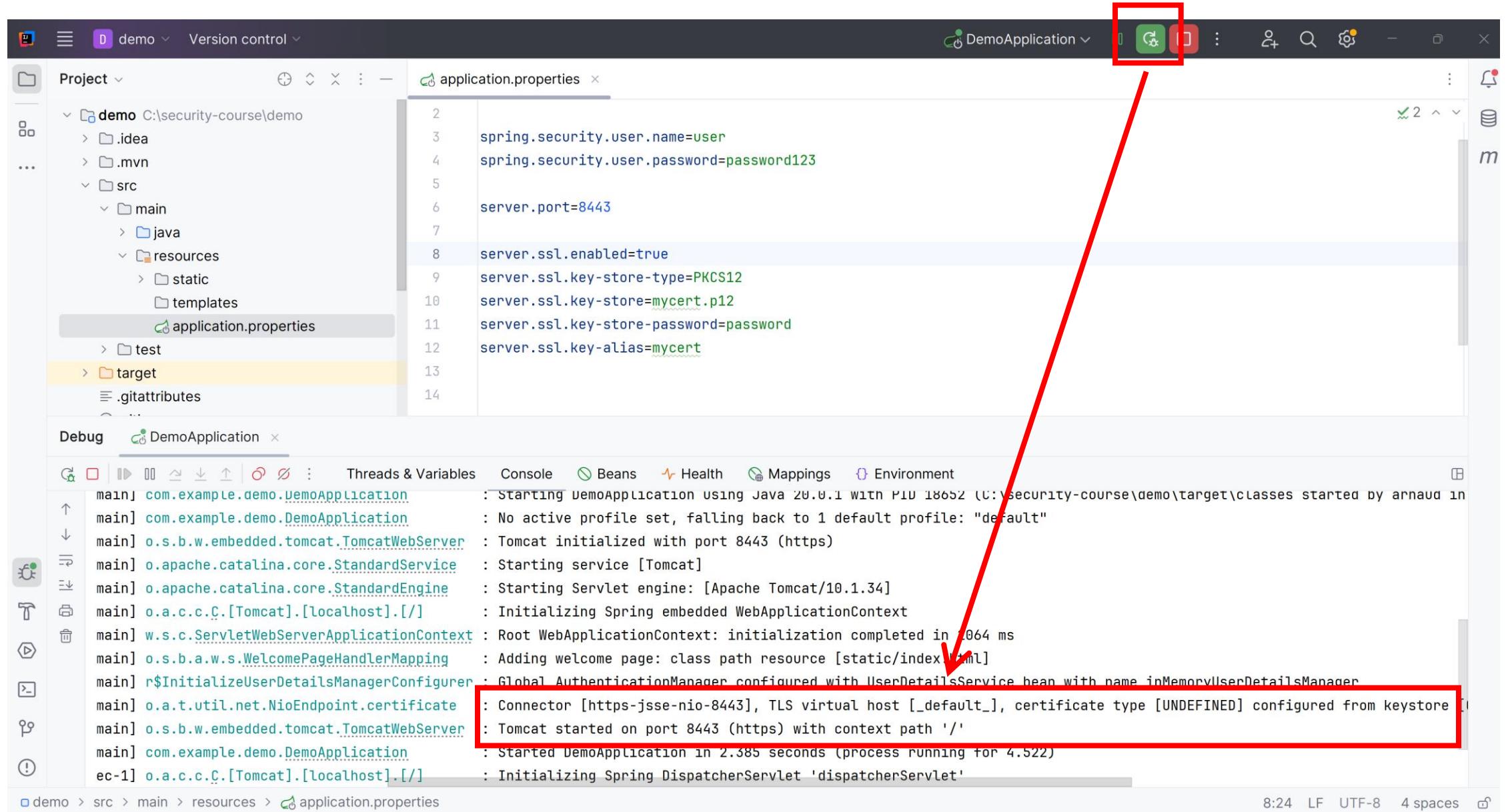
server.ssl.key-store-type=PKCS12

server.ssl.key-store=mycert.p12

server.ssl.key-store-password=password

server.ssl.key-alias=mycert

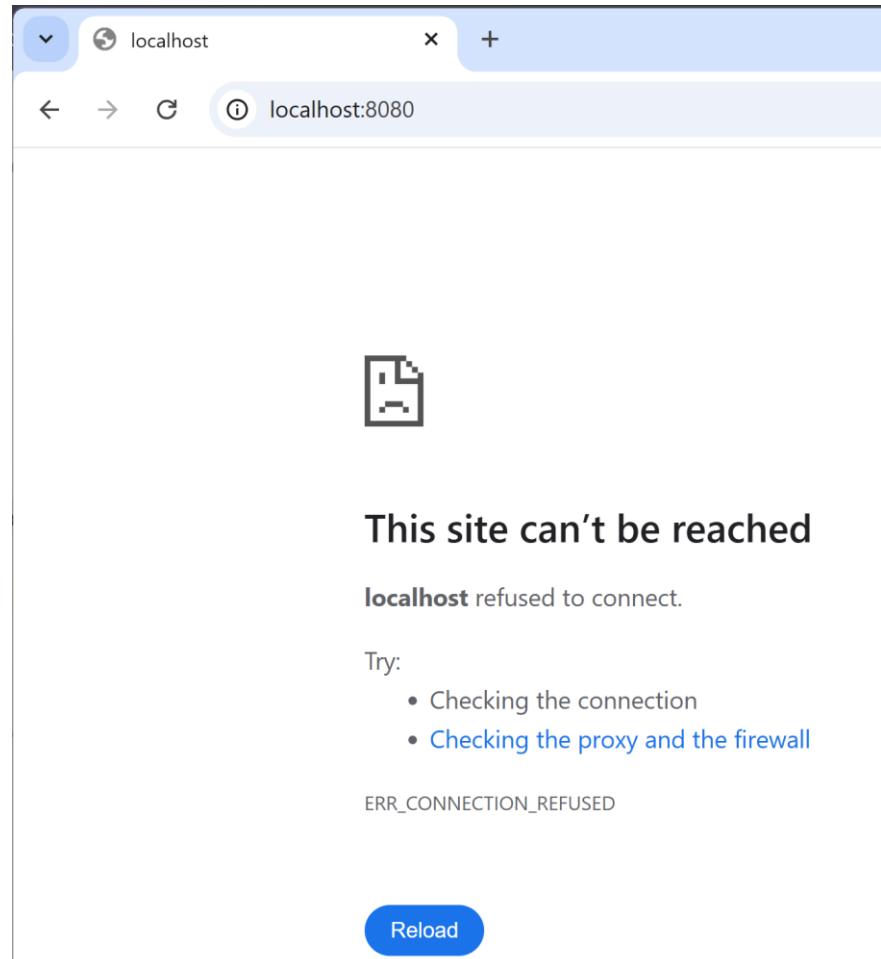
Relaunch server, check new console log



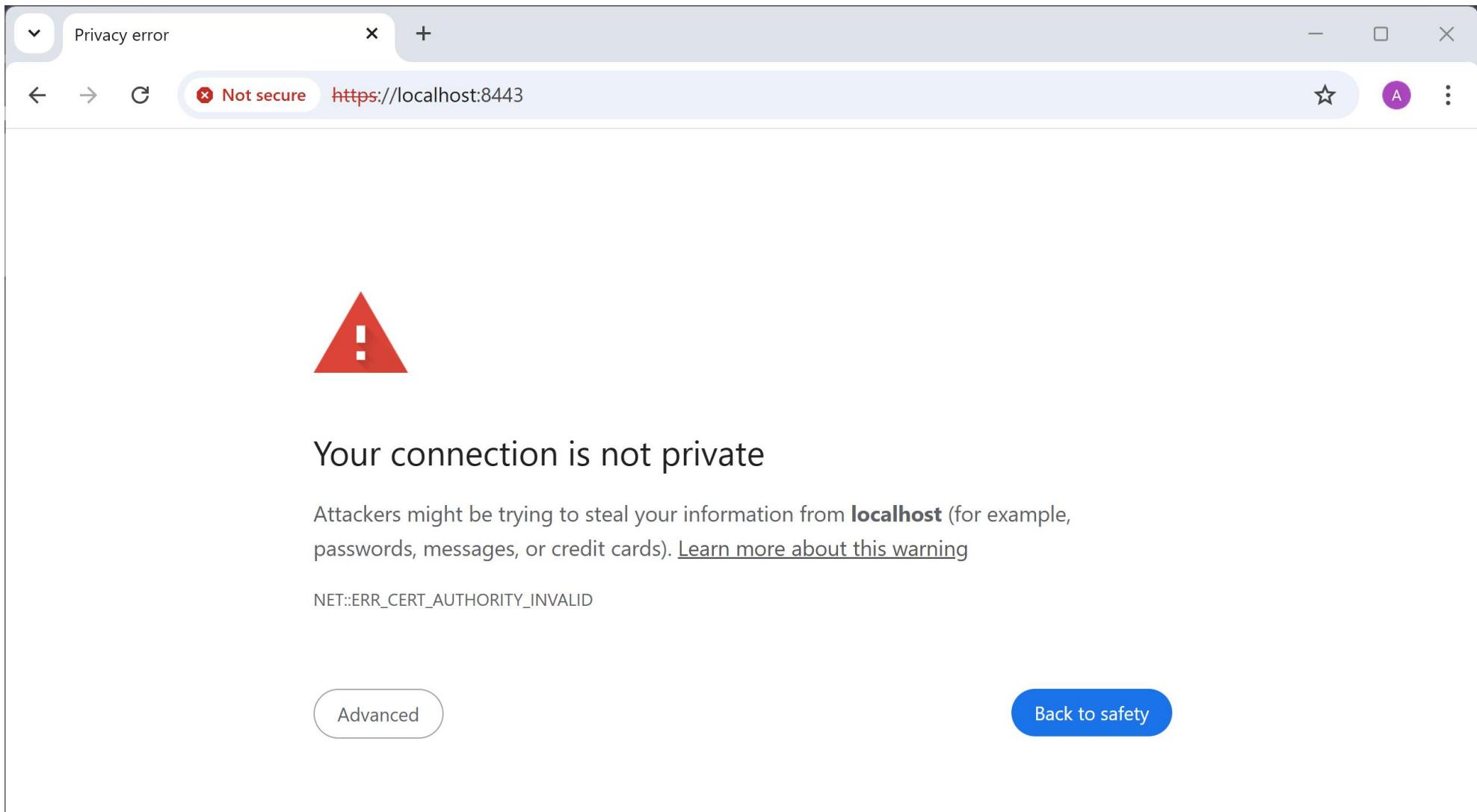
port changed 8080 (default for http)
-> 8443 (default for https)

on previous port 8080
=> "ERROR Connection refused"

OK, Normal !!



<https://localhost:8443> (self-signed certificate)
CERT_AUTHORITY_INVALID ... OK



Why ?

... because of self-signed certificate
NOT built-in recognized in your Chrome

Alternatives ?

=> pay 200 EURO/year to have a valid certificate ?

=> use GO-DADDY or other signing Authority on internet (recognized by Chrome)

=> recompile chrome to know "yourself" as valid authority

=> click on "ADVANCED" button to accept self-signed certificate

Advanced > Proceed to (unsafe)

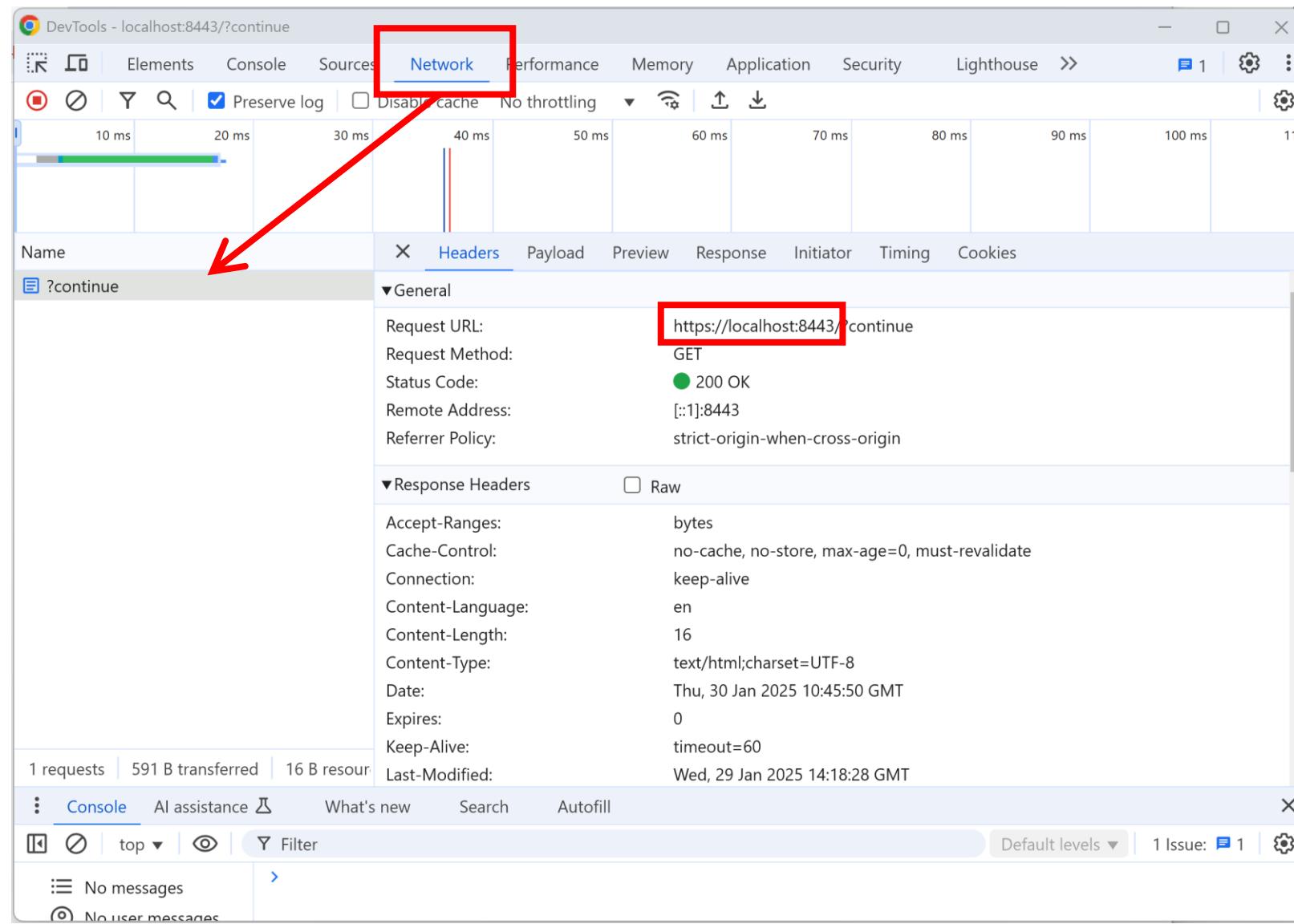
The image shows a web browser window with two tabs and a separate sign-in page.

Left Tab: The title bar says "Privacy error". The address bar shows "Not secure https://localhost:8443". The content area displays a warning message: "Your connection is not private" with a red exclamation mark icon. It states: "Attackers might be trying to steal your information from **localhost** (for example, passwords, messages, or credit cards). [Learn more about this warning](#)". Below this, the error code "NET::ERR_CERT_AUTHORITY_INVALID" is shown. A red box highlights the "Advanced" button, and a red arrow points from it to the "Proceed to localhost (unsafe)" link, which is also highlighted with a red box.

Right Tab: The title bar says "Please sign in". The address bar shows "Not secure https://localhost:8443/login". The content area displays a sign-in form with fields for "Username" and "Password", and a "Sign in" button. A red box highlights the "Not secure" status in the address bar.

Chrome DevTools Network requests

... same on "https://"



The screenshot shows the Chrome DevTools Network tab interface. A red arrow points from the title text to the 'Headers' section of the request details.

Request URL: <https://localhost:8443/?continue>

Request Method: GET

Status Code: 200 OK

Remote Address: [::1]:8443

Referrer Policy: strict-origin-when-cross-origin

Response Headers

Header	Value
Accept-Ranges	bytes
Cache-Control	no-cache, no-store, max-age=0, must-revalidate
Connection	keep-alive
Content-Language	en
Content-Length	16
Content-Type	text/html; charset=UTF-8
Date	Thu, 30 Jan 2025 10:45:50 GMT
Expires	0
Keep-Alive	timeout=60
Last-Modified	Wed, 29 Jan 2025 14:18:28 GMT

1 requests | 591 B transferred | 16 B resource

Console AI assistance What's new Search Autofill

Default levels ▾ 1 Issue: 1

Chrome DevTools "Security" tab

The screenshot shows the Chrome DevTools interface with the "Security" tab selected, indicated by a red box. The main content area displays a "Security overview" with three icons: a lock, an information circle, and a warning triangle. A prominent red message states: "This page is not secure (broken HTTPS)". Below this, two warning items are listed:

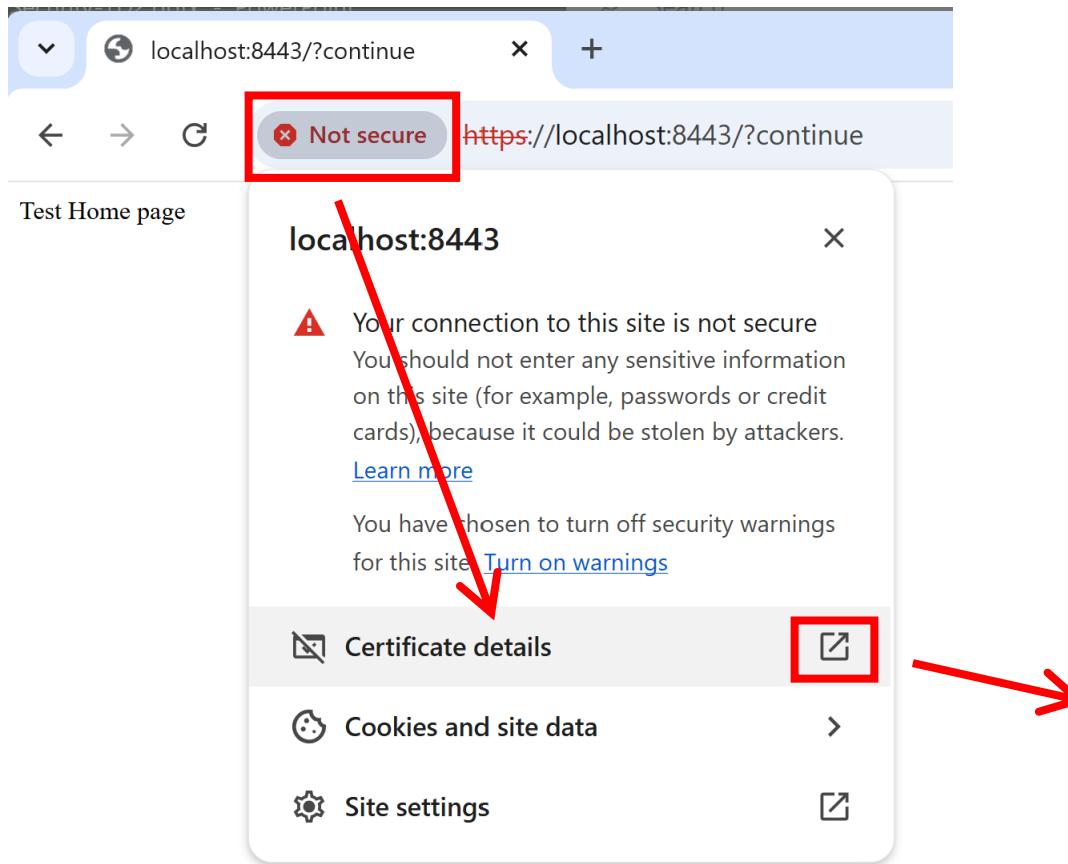
- ⚠ Certificate - Subject Alternative Name missing**: The certificate for this site does not contain a Subject Alternative Name extension containing a domain name or IP address. A "View certificate" button is provided.
- ⚠ Certificate - missing**: This site is missing a valid, trusted certificate (net::ERR_CERT_AUTHORITY_INVALID). A "View certificate" button is provided.

Two green success items are also listed:

- 🔒 Connection - secure connection settings**: The connection to this site is encrypted and authenticated using TLS 1.3, X25519, and AES_128_GCM.
- 🔒 Resources - all served securely**: All resources on this page are served securely.

The bottom of the DevTools window shows the standard toolbar with "Console" selected, along with "AI assistance", "What's new", "Search", and "Autofill". The bottom right corner shows the message "Default levels ▾ 1 Issue: 1" and a gear icon.

See Also Chrome -> Certificate Details



Suggestion ...
open view on a real web-site, "https://www.google.fr"
to see a valid signed certificate

Outline

Authentication

Authorization - Permission

Confidentiality/Encryption



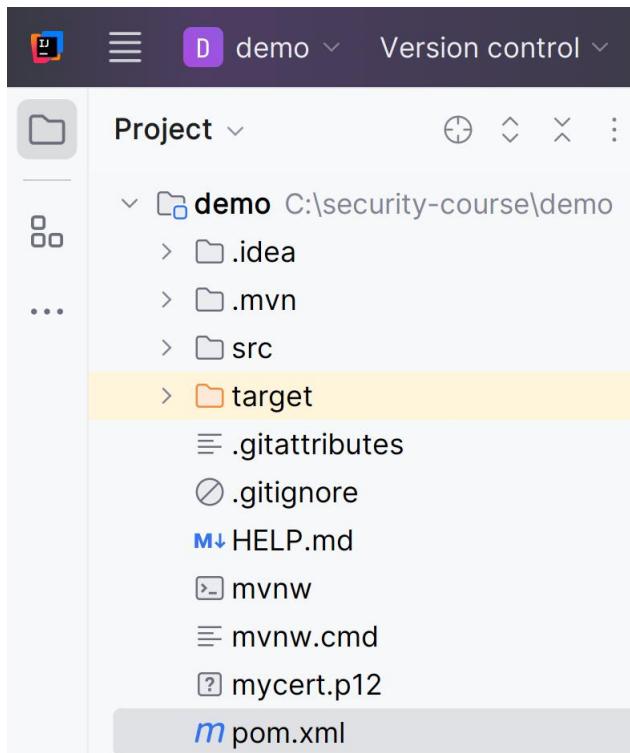
Backend-end: storing password in Database

Cryptographic "Hash" function, Salt

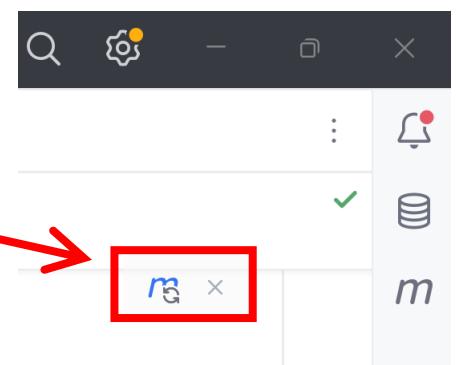
Who is "John The Ripper" ?

Edit pom.xml

Add maven dependency for Database
inside <dependencies> ..

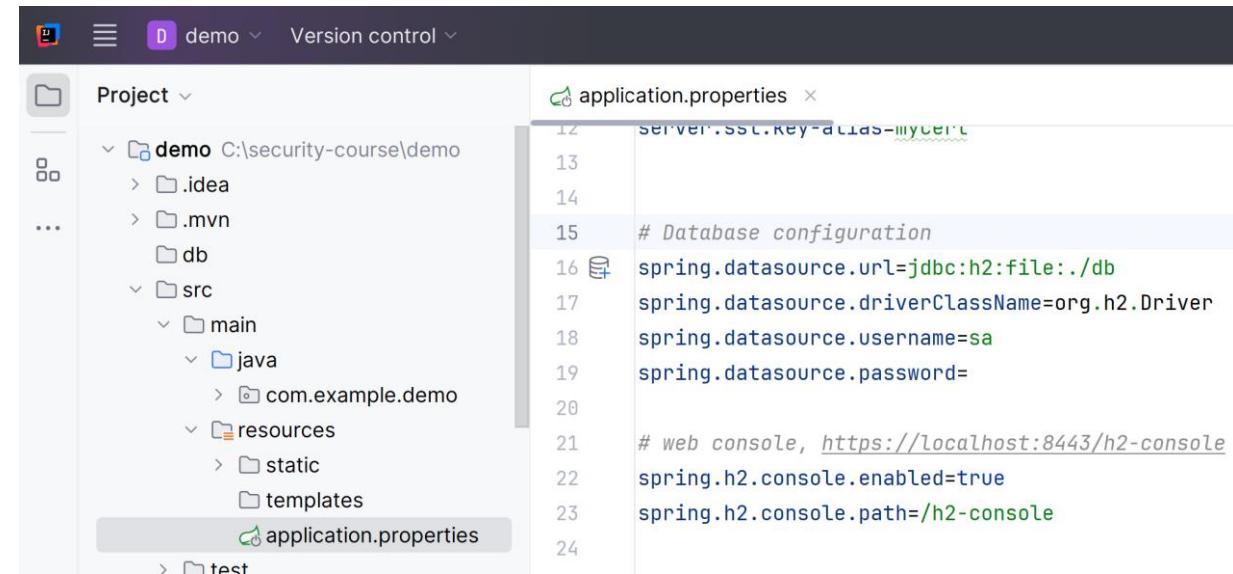


```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
    <groupId>com.h2database</groupId>
    <artifactId>h2</artifactId>
</dependency>
```



then click on refresh maven

Edit application.properties add H2 config



The screenshot shows a Java IDE interface with a dark theme. In the top bar, there are icons for file, edit, and version control, followed by the project name "demo". The left sidebar shows the project structure under "Project": "demo" (C:\security-course\demo) contains ".idea", ".mvn", "db", "src" (which includes "main" and "resources" folders), and "test". The "resources" folder contains "static" and "templates" subfolders, and the "application.properties" file is selected and highlighted with a grey background.

application.properties

```
12 SERVER.SSL.KEY=-----  
13  
14  
15 # Database configuration  
16 spring.datasource.url=jdbc:h2:file:./db  
17 spring.datasource.driverClassName=org.h2.Driver  
18 spring.datasource.username=sa  
19 spring.datasource.password=  
20  
21 # web console, https://localhost:8443/h2-console  
22 spring.h2.console.enabled=true  
23 spring.h2.console.path=/h2-console  
24
```

```
# Database configuration  
spring.datasource.url=jdbc:h2:file:./db  
spring.datasource.driverClassName=org.h2.Driver  
spring.datasource.username=sa  
spring.datasource.password=  
  
# web console, see https://localhost:8443/h2-console  
spring.h2.console.enabled=true  
spring.h2.console.path=/h2-console
```

Create java class SecurityDbConfiguration

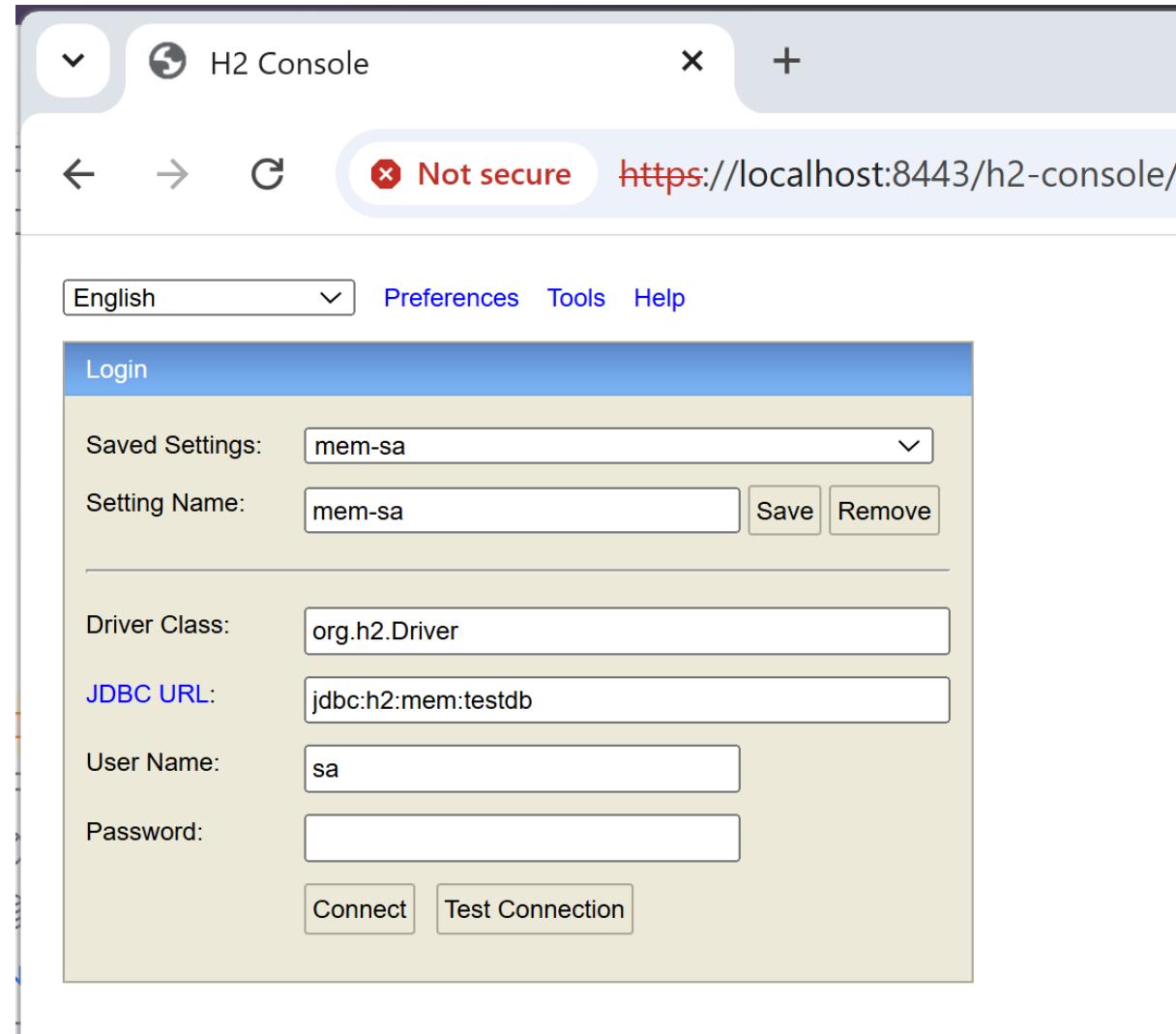
```
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.provisioning.JdbcUserDetailsManager;
import org.springframework.security.provisioning.UserDetailsManager;
import javax.sql.DataSource;

@Configuration
public class SecurityDbConfiguration {

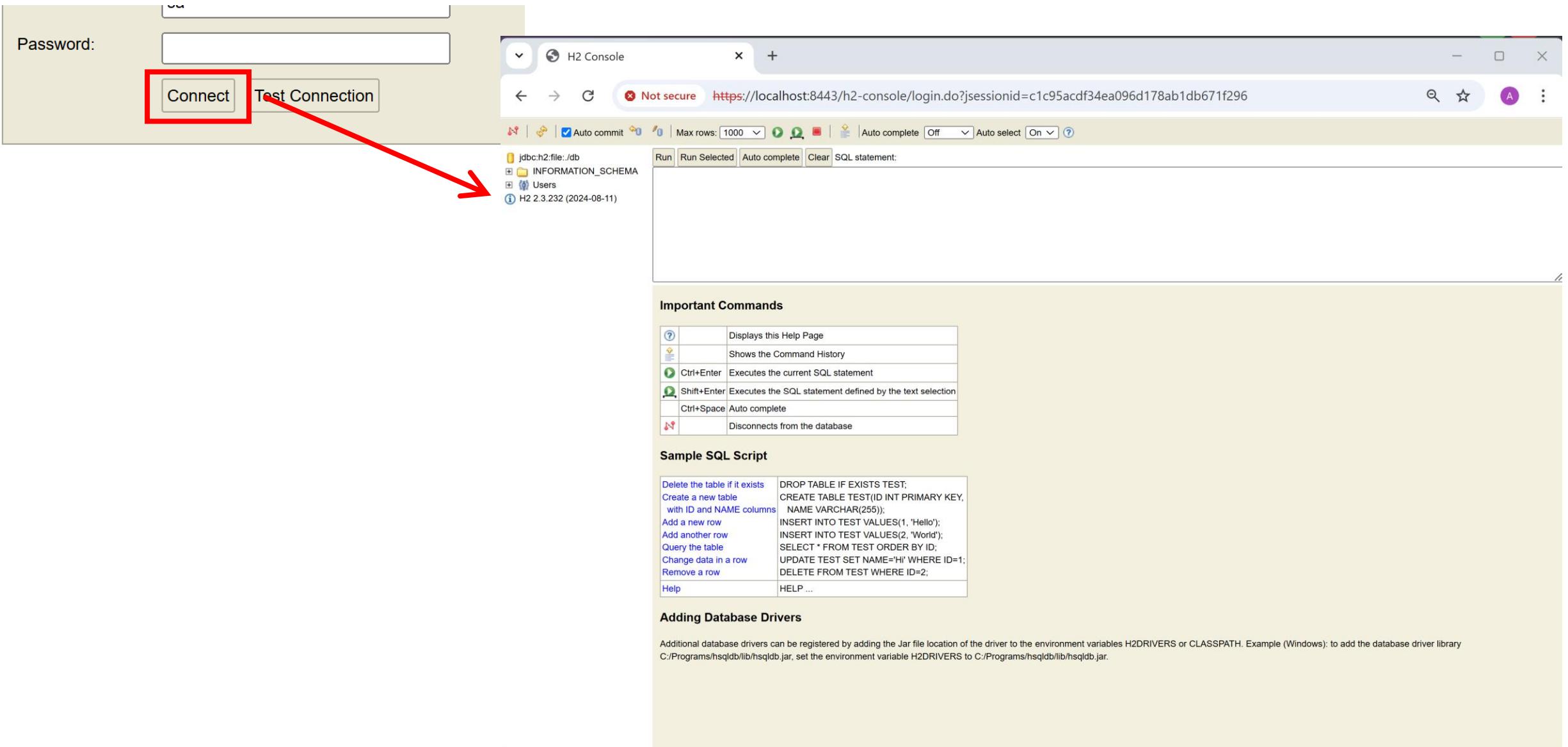
    @Bean
    UserDetailsManager users(DataSource dataSource) {
        JdbcUserDetailsManager users = new JdbcUserDetailsManager(dataSource);
        return users;
    }

}
```

Relaunch, Open <http://localhost:8443/h2-console>



Click "Connect"



Copy&Paste Create Tables DDL

```
create table users(
    username varchar_ignorecase(50) not null primary key,
    password varchar_ignorecase(500) not null,
    enabled boolean not null
);

create table authorities (
    username varchar_ignorecase(50) not null,
    authority varchar_ignorecase(50) not null,
    constraint fkAuthoritiesUsers foreign key(username) references
users(username)
);
create unique index ix_auth_username on authorities (username,authority);
```

cf doc

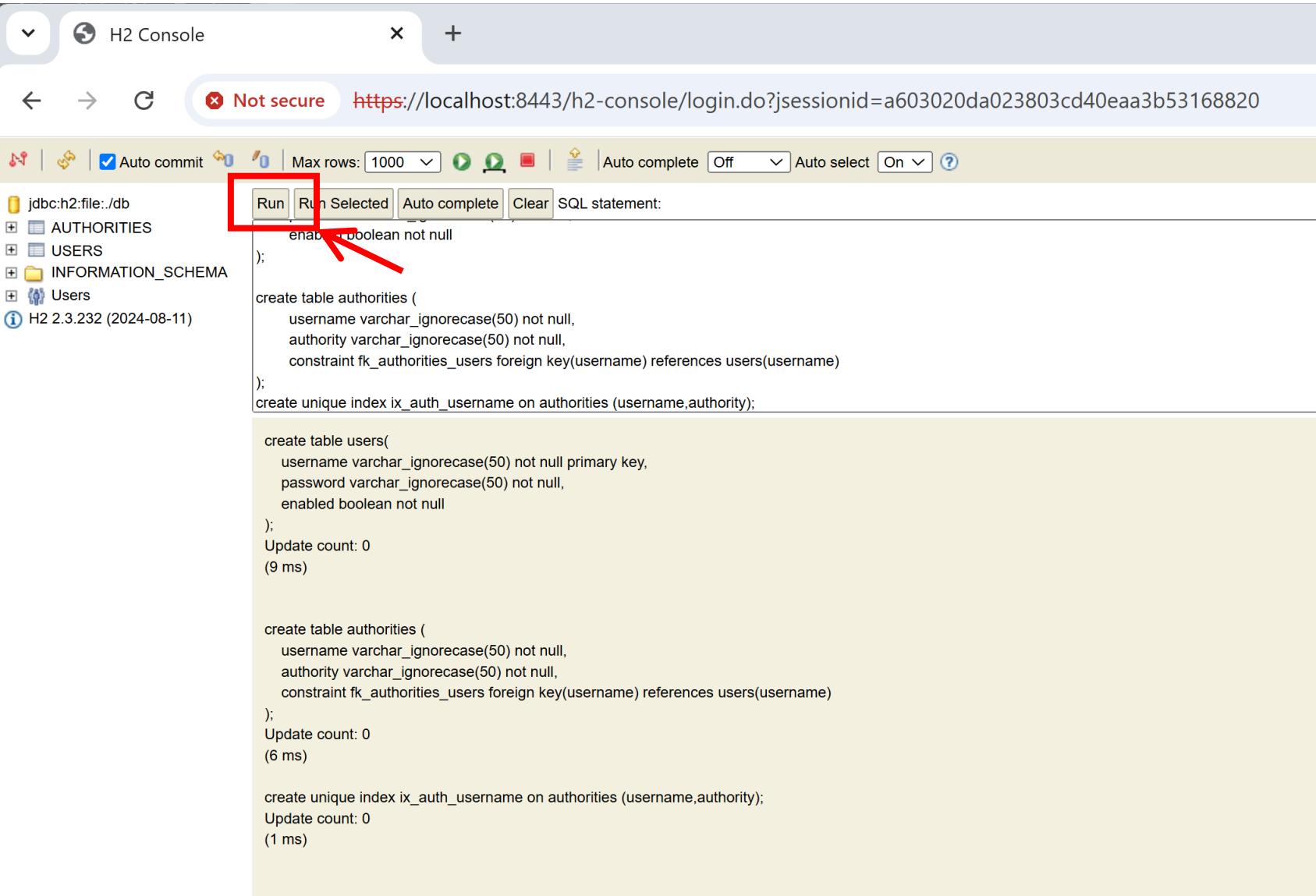
<https://docs.spring.io/spring-security/reference/servlet/authentication/passwords/jdbc.html>

The screenshot shows a web browser displaying the Spring Security JDBC Authentication documentation. The URL in the address bar is docs.spring.io/spring-security/reference/servlet/authentication/passwords/jdbc.html. The page title is "JDBC Authentication :: Spring Security". The left sidebar contains navigation links for "Spring Security 6.4.2", "Search", "Overview", "Prerequisites", "Community", "What's New", "Preparing for 7.0", "Migrating to 6.2", "Getting Spring Security", "Javadoc", "Features", "Project Modules", "Samples", "Servlet Applications", "Getting Started", and "Architecture". The main content area shows the "User Schema" section, which explains that `JdbcDaoImpl` requires tables to load the password, account status (enabled or disabled) and a list of authorities (roles) for the user. A "NOTE" box states that the default schema is exposed as a classpath resource named `org/springframework/security/core/userdetails/jdbc/users.ddl`. Below this, the "Default User Schema" is shown in SQL code:

```
create table users(
    username varchar_ignorecase(50) not null primary key,
    password varchar_ignorecase(500) not null,
    enabled boolean not null
);

create table authorities (
    username varchar_ignorecase(50) not null,
    authority varchar_ignorecase(50) not null,
    constraint fkAuthorities_users foreign key(username) references users(username)
);
```

Execute Create Table DDL



The screenshot shows the H2 Console interface with the following details:

- Title Bar:** H2 Console
- Address Bar:** Not secure https://localhost:8443/h2-console/login.do?jsessionid=a603020da023803cd40eaa3b53168820
- Toolbar Buttons:** Auto commit, Max rows: 1000, Auto complete: Off, Auto select: On.
- Tool Buttons:** Run, Run Selected, Auto complete, Clear, SQL statement: (text input field).
- Left Sidebar:** Database structure tree showing: jdbc:h2:file:/db, AUTHORITIES, USERS, INFORMATION_SCHEMA, and Users. Version: H2 2.3.232 (2024-08-11).
- SQL Editor Area:** Displays the execution of three Create Table DDL statements. The first statement creates the 'authorities' table with columns: enabled boolean not null, username varchar_ignorecase(50) not null, authority varchar_ignorecase(50) not null, and a constraint fkAuthorities_users foreign key(username) references users(username). The second statement creates a unique index ix_auth_username on the authorities table. The third statement creates the 'users' table with columns: username varchar_ignorecase(50) not null primary key, password varchar_ignorecase(50) not null, and enabled boolean not null. The fourth statement creates the same 'authorities' table again with the same structure. The fifth statement creates the same unique index ix_auth_username on the authorities table. All statements show an update count of 0 and execution times of 9 ms, 6 ms, and 1 ms respectively.

```
enable boolean not null
);
create table authorities (
    username varchar_ignorecase(50) not null,
    authority varchar_ignorecase(50) not null,
    constraint fkAuthorities_users foreign key(username) references users(username)
);
create unique index ix_auth_username on authorities (username,authority);

create table users(
    username varchar_ignorecase(50) not null primary key,
    password varchar_ignorecase(50) not null,
    enabled boolean not null
);
Update count: 0
(9 ms)

create table authorities (
    username varchar_ignorecase(50) not null,
    authority varchar_ignorecase(50) not null,
    constraint fkAuthorities_users foreign key(username) references users(username)
);
Update count: 0
(6 ms)

create unique index ix_auth_username on authorities (username,authority);
Update count: 0
(1 ms)
```

(Optionnal) DDL for Groups & Members

```
create table groups (
    id bigint generated by default as identity(start with 0) primary key,
    group_name varchar_ignorecase(50) not null
);

create table groupAuthorities (
    group_id bigint not null,
    authority varchar(50) not null,
    constraint fk_groupAuthorities_group foreign key(group_id) references groups(id)
);

create table groupMembers (
    id bigint generated by default as identity(start with 0) primary key,
    username varchar(50) not null,
    group_id bigint not null,
    constraint fk_groupMembers_group foreign key(group_id) references groups(id)
);
```

Insert users in database

Using H2 "New Row", edit, commit

The screenshot illustrates the process of inserting users into a database using the H2 "New Row", edit, commit feature. It consists of three panels:

- Panel 1 (Top Left):** Shows the database structure with the **USERS** table selected. A red box highlights the **USERS** table in the tree view. A red arrow points from this panel to the **Edit** button in Panel 2.
- Panel 2 (Bottom Left):** Displays the result of the `SELECT * FROM USERS;` query. It shows two rows: `user1`, `password1`, `TRUE` and `user2`, `password2`, `TRUE`. A red box highlights the **Edit** button at the bottom left of the results table.
- Panel 3 (Bottom Right):** Shows the `@edit SELECT * FROM USERS;` command with the results table. The first row (`user1`) has a delete icon (red X) in the Action column. The second row (`user2`) has a plus icon (+) in the Action column. A red box highlights the plus icon. A red arrow points from the plus icon in Panel 3 to the plus icon in Panel 2.

Resulting Data:

Action	USERNAME	PASSWORD	ENABLED
✓ ✗	user1	{noop}password1	TRUE
✚	user2	{noop}password2	TRUE

NOTICE: should prefix password with `{noop}`
(see next lesson for Hash and Salting)



INSERT INTO users... using SQL

INSERT INTO users (username, password, enabled) VALUES ('user2', '{noop}password2', true)

The screenshot shows the H2 Console interface running in a web browser at <https://localhost:8443/h2-console/login.do?jsessionid=bb660cf37e72b9dcb4dd56287dd>. The interface includes a sidebar with database objects like AUTHORITIES, USERS, INFORMATION_SCHEMA, and Users. The main area contains a toolbar with various icons, a dropdown for 'Max rows: 1000', and buttons for 'Run', 'Run Selected', 'Auto complete', and 'Clear'. A text input field labeled 'SQL statement:' contains the SQL command: `INSERT INTO users (username, password, enabled) VALUES ('user2', '{noop}password2', true)`. Below the input field is a large text area showing the results of the query.

```
SELECT * FROM users
```

Also add users a "role" (= AUTHORITY)

The screenshot shows the H2 Database Console interface. The left sidebar lists databases (jdbc:h2:file:./db), schemas (AUTHORITIES, USERS, INFORMATION_SCHEMA), and tables (Users). The right pane contains a SQL editor with the query `SELECT * FROM AUTHORITIES`. Below the editor is a table with two rows:

Action	USERNAME	AUTHORITY
	user1	ROLE_USER
	user2	ROLE_MANAGER

(2 rows, 1 ms)

```
INSERT INTO authorities (username, authority) VALUES ('user1', 'ROLE_USER')
```

```
INSERT INTO authorities (username, authority) VALUES ('user2', 'ROLE_MANAGER')
```



If NO role => user treated as 'Not Found' !

cf source code

<https://github.com/spring-projects/spring-security/blob/main/core/src/main/java/org/springframework/security/core/userdetails/jdbc/JdbcDaoImpl.java#L200>

The screenshot shows a GitHub code editor interface. The left sidebar displays the project structure under 'spring-security/core/src/main/java'. The main panel shows the file 'JdbcDaoImpl.java' with line numbers 181 to 206. Line 200 is highlighted with a yellow background. The code snippet is as follows:

```
181  
182     @Override  
183     public UserDetails loadUserByUsername(String username) throws UsernameNotFoundException {  
184         List<UserDetails> users = loadUsersByUsername(username);  
185         if (users.isEmpty()) {  
186             this.logger.debug("Query returned no results for user '" + username + "'");  
187             throw new UsernameNotFoundException(this.messages.getMessage("JdbcDaoImpl.notFound",  
188                     new Object[] { username }, "Username {0} not found"));  
189         }  
190         UserDetails user = users.get(0); // contains no GrantedAuthority[]  
191         Set<GrantedAuthority> dbAuthsSet = new HashSet<>();  
192         if (this.enableAuthorities) {  
193             dbAuthsSet.addAll(loadUserAuthorities(user.getUsername()));  
194         }  
195         if (this.enableGroups) {  
196             dbAuthsSet.addAll(loadGroupAuthorities(user.getUsername()));  
197         }  
198         List<GrantedAuthority> dbAuths = new ArrayList<>(dbAuthsSet);  
199         addCustomAuthorities(user.getUsername(), dbAuths);  
200         if (dbAuths.isEmpty()) {  
201             this.logger.debug("User '" + username + "' has no authorities and will be treated as 'not found'"  
202             throw new UsernameNotFoundException(this.messages.getMessage("JdbcDaoImpl.noAuthority",  
203                     new Object[] { username }, "User {0} has no GrantedAuthority"));  
204         }  
205         return createUserDetails(username, user, dbAuths);  
206     }
```



Springboot refuse password without "Encoder" ... or use explicitly "{noop}"

Screenshot of a Java IDE (IntelliJ IDEA) showing a stack trace in the Debug tool window. The stack trace indicates a failure due to a missing password encoder.

```
at org.springframework.security.config.annotation.web.configuration.WebMvcSecurityConfiguration$CompositeFilterChainProxy.doFilter(WebMvcSecurityConfiguration.java:410)
2025-01-30T16:13:13.137+01:00 ERROR 21412 --- [demo] [io-8443-exec-10] o.a.c.c.C.[.].[dispatcherServlet] : Servlet.service() for servlet [dispatcherServlet] in
java.lang.IllegalArgumentException Create breakpoint : Given that there is no default password encoder configured, each password must have a password encoding prefix. Please
at org.springframework.security.crypto.password.DelegatingPasswordEncoder$UnmappedIdPasswordEncoder.matches(DelegatingPasswordEncoder.java:307) ~[spring-security-crypto-6.4.2.jar:6.4.2]
at org.springframework.security.crypto.password.DelegatingPasswordEncoder.matches(DelegatingPasswordEncoder.java:248) ~[spring-security-crypto-6.4.2.jar:6.4.2]
at org.springframework.security.authentication.dao.DaoAuthenticationProvider.additionalAuthenticationChecks(DaoAuthenticationProvider.java:90) ~[spring-security-core-6.4.2.jar:6.4.2]
at org.springframework.security.authentication.dao.AbstractUserDetailsAuthenticationProvider.authenticate(AbstractUserDetailsAuthenticationProvider.java:147) ~[spring-security-core-6.4.2.jar:6.4.2]
at org.springframework.security.authentication.ProviderManager.authenticate(ProviderManager.java:182) ~[spring-security-core-6.4.2.jar:6.4.2]
at org.springframework.security.authentication.ProviderManager.authenticate(ProviderManager.java:201) ~[spring-security-core-6.4.2.jar:6.4.2]
at org.springframework.security.web.authentication.UsernamePasswordAuthenticationFilter.attemptAuthentication(UsernamePasswordAuthenticationFilter.java:85) ~[spring-security-web-6.4.2.jar:6.4.2]
```

Given that there is no default password encoder configured, each password must have a password encoding prefix.
Please either prefix this password with '{noop}' or set a default password encoder in `DelegatingPasswordEncoder`.



Check (update) password to be "{noop}password"

Screenshot of a database management tool showing the 'USERS' table.

Toolbar: Auto commit checked, Max rows: 1000, Run, Run Selected, Auto complete, Clear, SQL statement.

Table structure:

Action	USERNAME	PASSWORD	ENABLED
X	user1	{noop}password1	TRUE
X	user2	{noop}password2	TRUE

(2 rows, 0 ms)

Screenshot of a database management tool showing the 'USERS' table.

Toolbar: Run, Run Selected, Auto complete, Clear, SQL statement.

Table structure:

Action	USERNAME	PASSWORD	ENABLED
X	user1	{noop}password1	TRUE
X	user2	{noop}password2	TRUE

Screenshot of a database management tool showing the 'USERS' table.

Toolbar: Run, Run Selected, Auto complete, Clear, SQL statement.

Table structure:

Action	USERNAME	PASSWORD	ENABLED
✓ X	user1	{noop}password1	TRUE
✓ X	user2	{noop}password2	TRUE

(2 rows, 0 ms)

Relaunch & Test ..

https://localhost:8443/login

Please sign in

user1

.....

Sign in

next lesson ... for activating the "HASH" + "Salting"

Outline

Authentication

Authorization - Permission

Confidentiality/Encryption

Backend-end: storing password in Database



Cryptographic "Hash" function, Salt

Who is "John The Ripper" ?

Re-Salting an already saved {noop}password

by default, springboot "BCryptPasswordEncoder" already generates a random Salt for each encoding

It encode both salt + hashed password in field "password"

(no need to save in a different column in database table)

Adding java class for Updating Password [1/2]

```
package com.example.demo;

import org.springframework.security.core.userdetails.User;
import org.springframework.security.core.userdetails.UserDetails;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.security.provisioning.UserDetailsManager;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

@RestController
@RequestMapping("/api/v1/users")
public class UserUpdateRestController {

    private final UserDetailsManager userDetailsManager;

    private final PasswordEncoder passwordEncoder = new BCryptPasswordEncoder();
    private static final String ENCODE_PREFIX = "{bcrypt}"; // cf PasswordEncoderFactories
```

(...next)

```
public UserUpdateRestController(UserDetailsManager userDetailsManager) {  
    this.userDetailsManager = userDetailsManager;  
    // for debug (uncomment to execute once, or use curl  
    // updateResaltPassword(new UserPasswordRequestDTO("user2", "password2"));  
}  
  
record UserPasswordRequestDTO(String username, String newPassword) {}  
  
@PostMapping("/update-password")  
public void updatePassword(@RequestBody UserPasswordRequestDTO req) {  
    UserDetails userDetails = userDetailsManager.loadUserByUsername(req.username);  
    String oldPassword = userDetails.getPassword();  
  
    String hashedPassword = ENCODE_PREFIX + passwordEncoder.encode(req.newPassword);  
  
    System.out.println("updating user password: old hashed '" + oldPassword + "' => new '" + hashedPassword + "'");  
    UserDetails updatedUser = User.withUserDetails(userDetails).password(hashedPassword).build();  
    userDetailsManager.updateUser(updatedUser);  
}  
}
```

Execute update password

Either Launch ONCE in debugger with un-commented line

```
19 public UserUpdateRestController(UserDetailsManager userDetailsManager) {  
20     this.userDetailsManager = userDetailsManager;  
21     // for debug (uncomment to execute once, or use curl  
22     //updatePassword(new UserPasswordRequestDTO("user2", "password2"));  
23 }
```

```
19 public UserUpdateRestController(UserDetailsManager userDetailsManager) {  
20     this.userDetailsManager = userDetailsManager;  
21     // for debug (uncomment to execute once, or use curl  
22     updatePassword(new UserPasswordRequestDTO(username: "user2", newPassword: "password2"));  
23 }
```

Or using Rest api

```
curl -v -k -u "user1:password1" -H 'Content-Type: application/json' \  
-X POST https://localhost:8443/api/v1/users/update-resalt-password \  
-d '{"username": "user1", "newPassword": "password1"}'
```

check HTTP 200 + Check in Debug Console :

Showing encoded password

The screenshot shows the H2 Database Browser interface. The left sidebar displays the database schema with tables: AUTHORITIES, USERS, INFORMATION_SCHEMA, and Users. The main area shows the results of a SQL query:

```
SELECT * FROM USERS;
```

USERNAME	PASSWORD	ENABLED
user2	{bcrypt}\${2a\$10\$BNAhHEMM9ihsq/SKTdmX7.yR9vXA2ZKzxhOYvHbkNohskr3BOstnC	TRUE
user1	{noop}password1	TRUE

(2 rows, 3 ms)

[Edit](#)

ANNEXE... Screenshot Debugging "encoders"

The screenshot shows a Java debugger interface with the following details:

- Structure** tab selected.
- Debug DemoApplication** session.
- Threads & Variables** tab selected.
- Evaluate expression** field: `String id = extractId(prefixEncoderassword), prefixEncoderassword. $2u$1$`
- Code changed:** `m`
- Breakpoints:** One breakpoint at `if (StringUtils.hasText(id)) {`.
- Stack Trace:** Shows multiple stack frames related to Spring Security's password encoding logic, such as `matches:299, DelegatingPasswordEncoder$UnmappedIdPasswordEncoder`, `matches:248, DelegatingPasswordEncoder`, and various authentication filters.
- Variables:** A detailed view of the `this` variable of type `DelegatingPasswordEncoder$UnmappedIdPasswordEncoder`. It includes:
 - `this$0`: `DelegatingPasswordEncoder`
 - `idPrefix`: `{`
 - `idSuffix`: `}`
 - `idForEncode`: `bcrypt`
 - `passwordEncoderForEncode`: `BCryptPasswordEncoder`
 - `idToPasswordEncoder`: `HashMap` with size 14, containing entries like:
 - `argon2` → `Argon2PasswordEncoder`
 - `pbkdf2` → `Pbkdf2PasswordEncoder`
 - `SHA-1` → `MessageDigestPasswordEncoder`
 - `sha256` → `StandardPasswordEncoder`
 - `pbkdf2@SpringSecurity_v5_8` → `Pbkdf2PasswordEncoder`
 - `scrypt@SpringSecurity_v5_8` → `SCryptPasswordEncoder`
 - `bcrypt` → `BCryptPasswordEncoder`
 - `noop` → `NoOpPasswordEncoder`
 - `ldap` → `LdapShaPasswordEncoder`
 - `SHA-256` → `MessageDigestPasswordEncoder`
 - `argon2@SpringSecurity_v5_8` → `Argon2PasswordEncoder`
 - `MD4` → `Md4PasswordEncoder`
 - `scrypt` → `SCryptPasswordEncoder`
 - `MD5` → `MessageDigestPasswordEncoder`

ANNEXE Screenshot Debugging authentication

The screenshot shows a Java code editor and a debugger interface.

Code Editor: The main window displays the file `AbstractUserDetailsAuthenticationProvider.java`. The code implements the `authenticate` method, which checks if the provided authentication token is an instance of `UsernamePasswordAuthenticationToken`. If so, it retrieves the user from the cache and returns the authentication object. If the user is null, it sets `cacheWasUsed` to `false`.

```
public abstract class AbstractUserDetailsAuthenticationProvider
    @Override
    public Authentication authenticate(Authentication authentication) throws AuthenticationException {
        Assert.isInstanceOf(UsernamePasswordAuthenticationToken.class, authentication, "Authentication must be an instance of UsernamePasswordAuthenticationToken");
        String username = determineUsername(authentication);
        boolean cacheWasUsed = true;
        UserDetails user = this.userCache.getUserFromCache(username);
        if (user == null) {
            cacheWasUsed = false;
        }
        try {
            user = retrieveUser(username, (UsernamePasswordAuthenticationToken) authentication);
        }
    }
}
```

Project Explorer: On the left, the project structure shows the `spring-security-core-6.4.2.jar` library root containing various packages like `META-INF`, `org.springframework.security.access`, `aot.hint`, `authentication`, and `dao`. The `AbstractUserDetailsAuthenticationProvider` class is selected in the `dao` package.

Debug View: The bottom section shows the debugger interface with the tab `Debug DemoApplication` selected. It lists threads and variables. A tooltip for the `authenticate` method shows the current state of variables:

- `this = {DaoAuthenticationProvider@11037}`
- `authentication = {UsernamePasswordAuthenticationToken@11036} "UsernamePasswordAuthenticationToken [Principal=use...]`
- `this.messages = {MessageSourceAccessor@11039}`

The status bar at the bottom indicates the file is `spring-security-core-6.4.2.jar`, the line number is `124:1`, and the encoding is `UTF-8`.

Outline

Authentication

Authorization - Permission

Confidentiality/Encryption

Backend-end: storing password in Database

Cryptographic "Hash" function, Salt



Who is "John The Ripper" ?

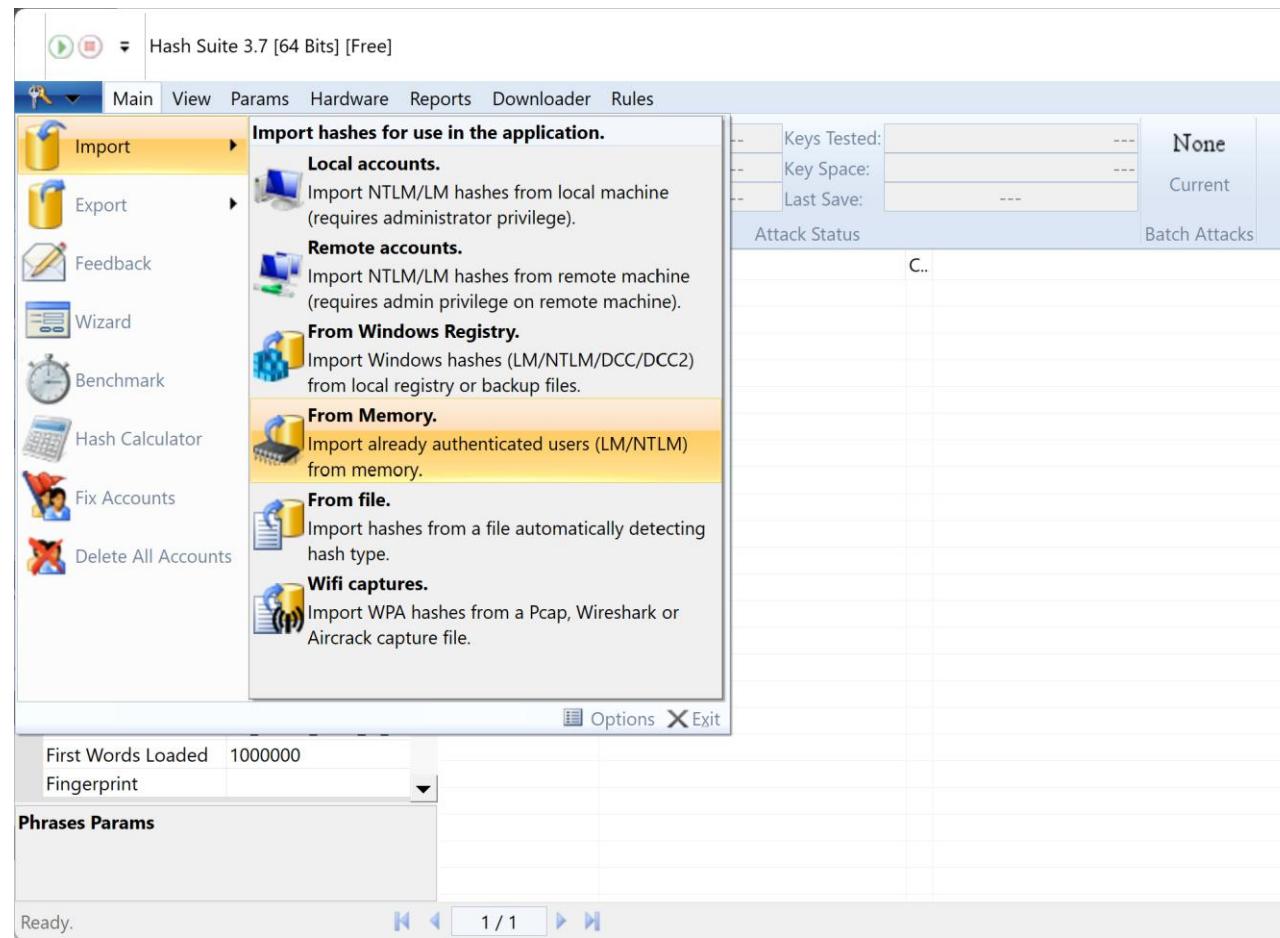
Download John The Ripper / HashSuite

- John the Ripper Pro for Linux
- John the Ripper Pro for macOS
- On Windows, consider Hash Suite (developed by Openwall)
- On Android, consider Hash Suite Droid

<https://hashsuite.openwall.net>

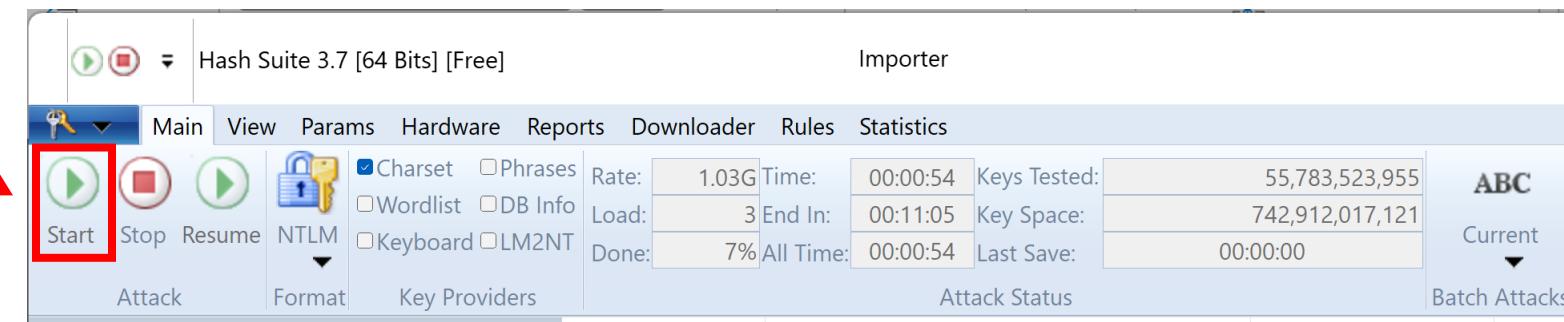
The screenshot shows the Hash Suite website at <https://hashsuite.openwall.net>. The main navigation bar includes Home, FAQ, Tutorial, Performance, Comparison, Download, Android, Changelog, License, and Contact. A banner at the top right encourages following @HashSuite on Twitter for new release announcements. Below the banner, a sub-header reads "Fast, powerful, simple". The central content area features a "Home" section with a brief description of Hash Suite's objectives and a list of its features: Fast, Simple and modern, Smart, Powerful, and Scalable. To the right, a "News" sidebar highlights two recent releases: "Hash Suite Droid 1.6" (October 28, 2021) and "Hash Suite 3.7, Droid 1.5.1" (March 19, 2021). Both news items mention the addition of support for Android 11 scoped storage enforcement. Below the news, a status bar displays "Hash Suite 3.7 [64 Bits] [Pro]" and provides real-time attack metrics: Rate: 16.9G, Time: 00:00:44, Keys Tested: 739, Load: 739, End In: 00:00:00, Key Space: 100%, Done: 100%, All Time: 00:00:44, Last Save: 00:00:44. A table below shows "Key Provider Params" for "Charset Params" with entries for "hcookis" (Hash: F83C01861FDD23B4354465FE6D7F6402, Cleartext: nurse), "emcnees" (Hash: C1A8D439E0906BBF241EBBC04BDB424C, Cleartext: ?????????????????????), "hpasceri" (Hash: 486EA753DEF361967B1A5E9C5D65EC18, Cleartext: ?????????????????????), and "ddauria" (Hash: 34643FD04B90614EF2E9A3E1DB738986, Cleartext: ?????????????????????).

Step 1/ Import Your Windows Hashed Password From Memory



Step 2/ Test your password strength

Run 5-10mn (default settings)



Step 3/ Edit "word.lst" containing some words (maybe known internet password databases) RE-Run 5-10mn with custom rules + wordlist

Restrict searches
with hypothesis

Add known words
in dictionary

The screenshot shows the Hash Suite 3.7 [64 Bits] [Free] application interface. The main window displays a table of users with their corresponding hashes and cleartexts. A red box highlights the 'Use rules' checkbox under 'Charset Params'. Another red box highlights the 'Wordlist Params' section under 'Key Provider Params'. A large red arrow points from the 'Wordlist Params' section in the left panel to the 'Wordlist' field in the right panel. The right panel shows the configuration for 'Wordlist Params' with 'Minimum Size' set to 1, 'Maximum Size' set to 6, 'Use rules' checked, and 'Wordlist' set to 'wordlist_small.lst (21.8 KB)'. The left panel also includes sections for 'Keyboard Params', 'Phrases Params', 'DB Info Params', 'LM2NT Params', and 'Rules'.

Username	Hash	Cleartext
Administrateur	31	????????????????????????...
arnaud	49	6 ?????????????????????...
DefaultAccount	31	0 ?????????????????????...
Invité	31	0 ?????????????????????...
WDAGUtilityAcc...	26	4 ?????????????????????...

Key Provider Params

- Charset Params
 - Minimum Size: 0
 - Maximum Size: 6
 - Use rules (highlighted)
- Wordlist Params (highlighted)
- Keyboard Params
- Phrases Params
- DB Info Params
- LM2NT Params
- Rules

Imported Data Summary:

Category	Value
Users Added:	5
Lines Skipped:	0
Completion:	100%
Added:	0
Disable:	5
Exist:	0
Added:	3
Disable:	0
Exist:	2

Importer

Key Provider Params

- Symbol
 - Symbol
 - !@#\$%^&*()_-+=~`[]{};:"...
- Wordlist Params
 - Minimum Size: 1
 - Maximum Size: 6
 - Use rules (highlighted)
 - Wordlist: wordlist_small.lst (21.8 KB) (highlighted)
- Keyboard Params
- Phrases Params
- DB Info Params
- LM2NT Params
- Rules

Summary

Authentication	 Transmit Password to Http
Authorization - Permission	password=>authentication=>role
Confidentiality/Encryption	password not in clear over internet
Backend-end: storing password in Database	storing password ...
Cryptographic "Hash" function, Salt	... storing hashed+salted password
Who is "John The Ripper" ?	finding password from hashed

Questions ?

arnaud.nauwynck@gmail.com