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

Authorization - Permission

Confidentiality/Encryption

Backend-end: storing password in Database

Cryptographic "Hash" function, Salt

Who is "John The Ripper" ?

Transmit Password to Http

password=>authentication=>role

password not in clear over internet

storing password ...

... storing hashed+salted password

finding password from hashed

Outline

Authentication

Authorization - Permission

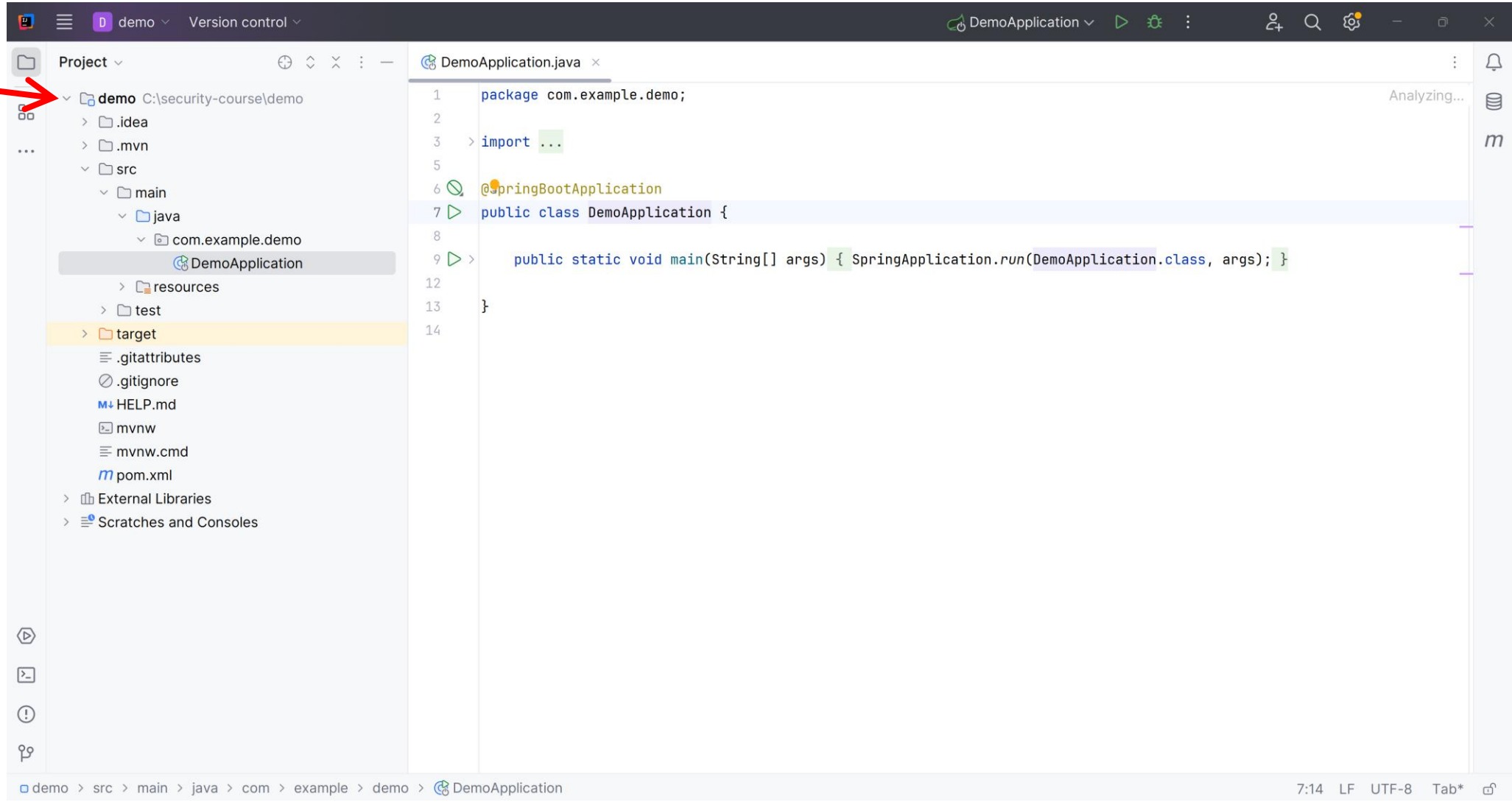
Confidentiality/Encryption

Backend-end: storing password in Database

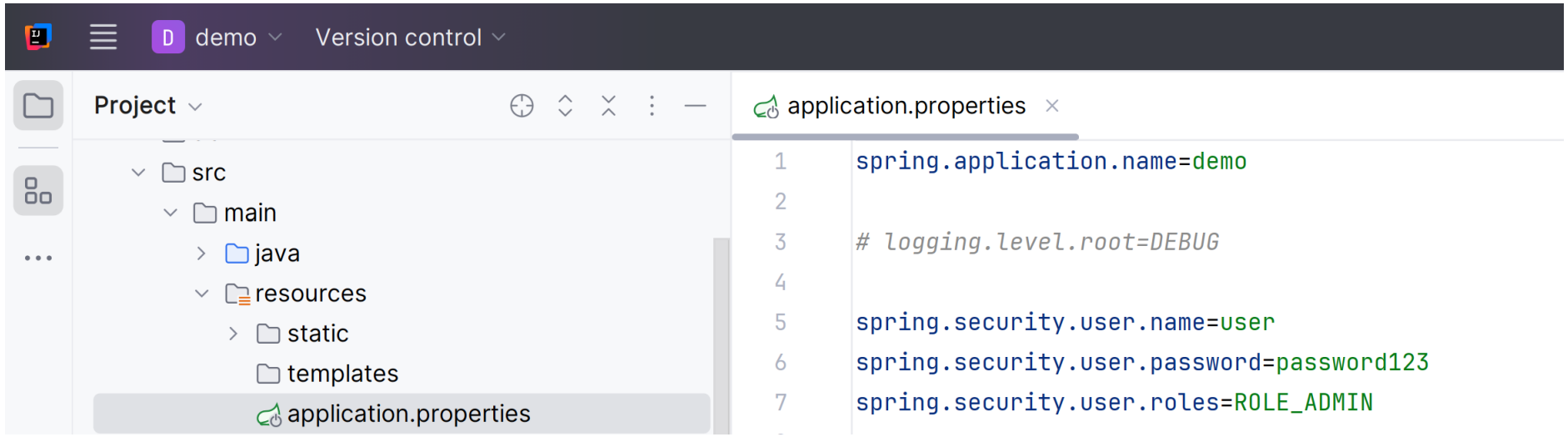
Cryptographic "Hash" function, Salt

Who is "John The Ripper" ?

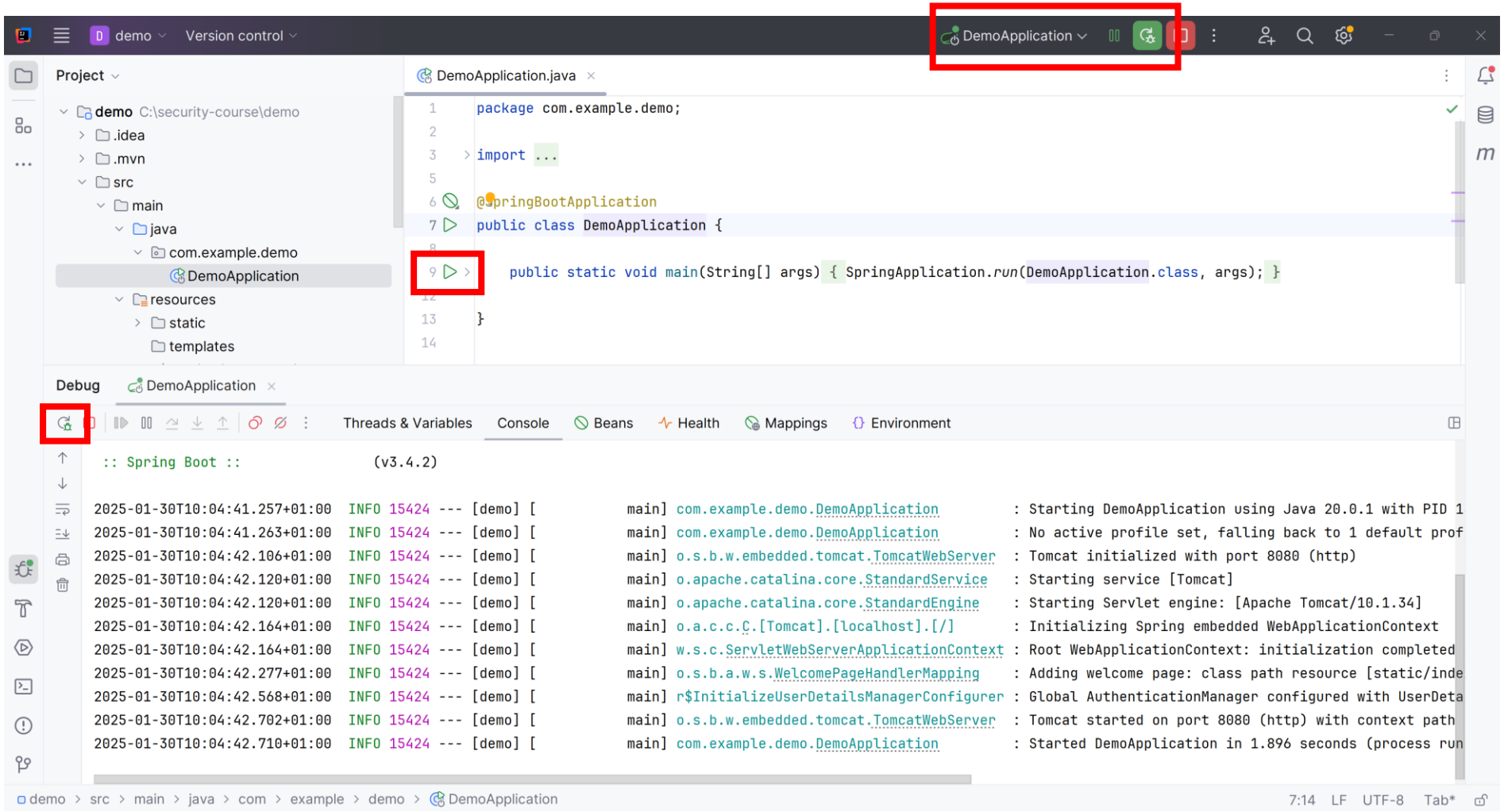
Pre-Requisite Step1: relaunch your IntelliJ IDE reopen project c:\security-course\demo



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



Pre-Requisite Step 3: relaunch webapp server

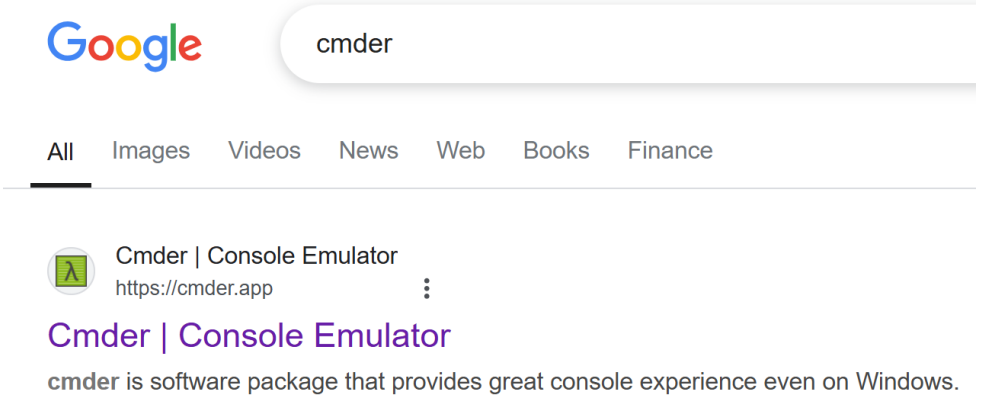


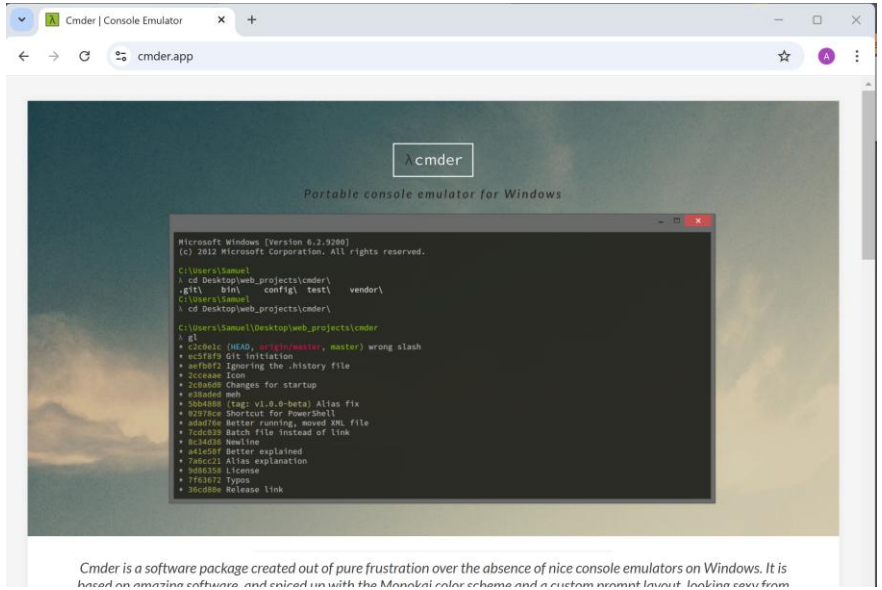
Pre-Requisite : Install (unix) shell tools "curl" and "base64"

on linux => built-in tools on system

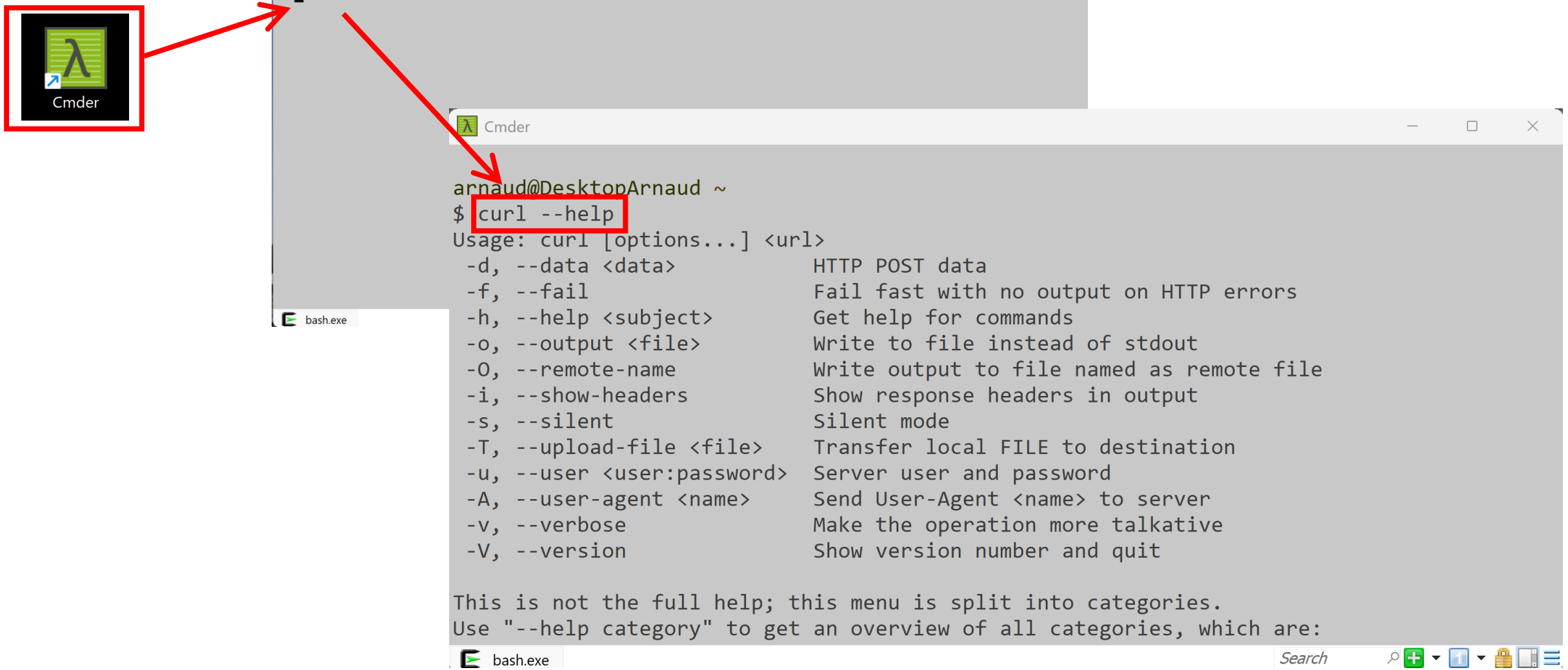
on mac => same (or use "brew install" ?)

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, ...

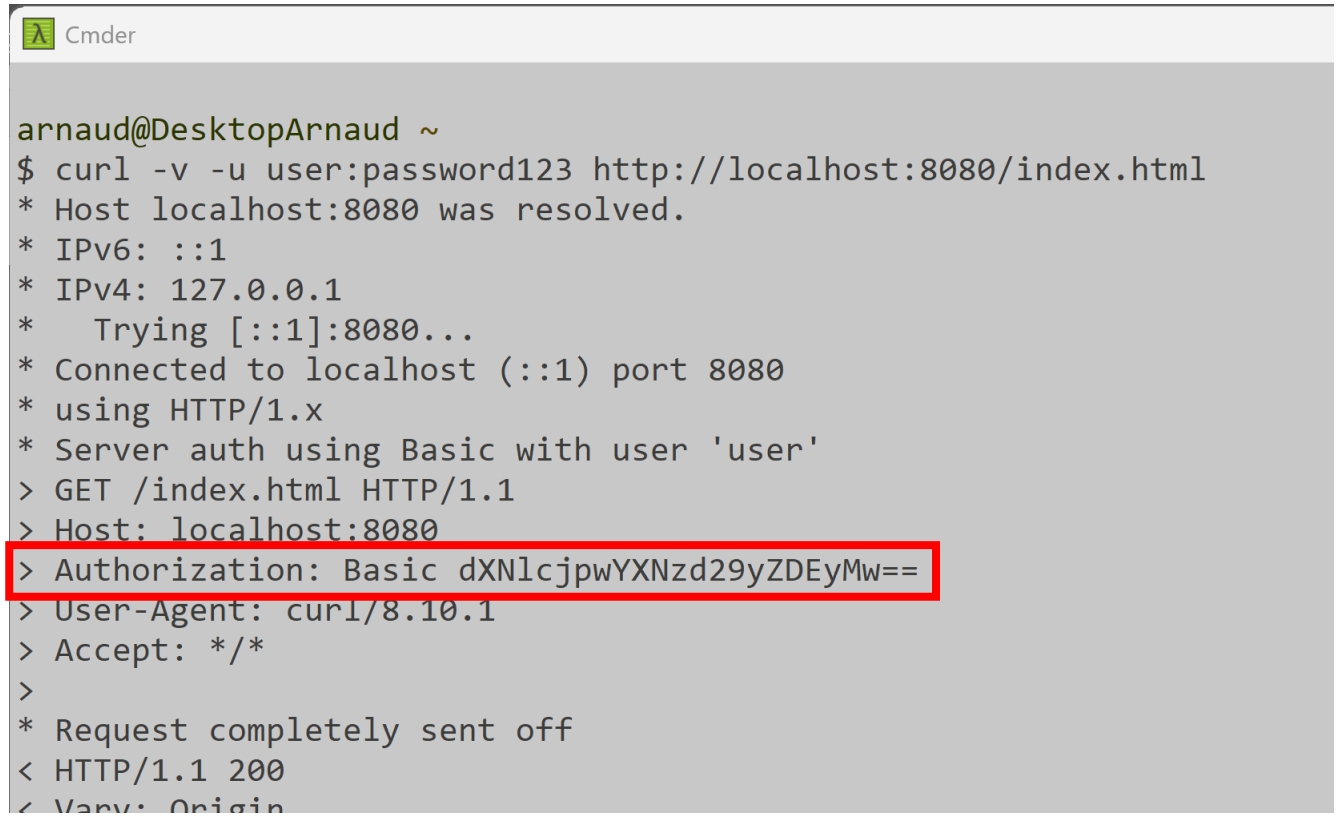
example with http **200**

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
<
```

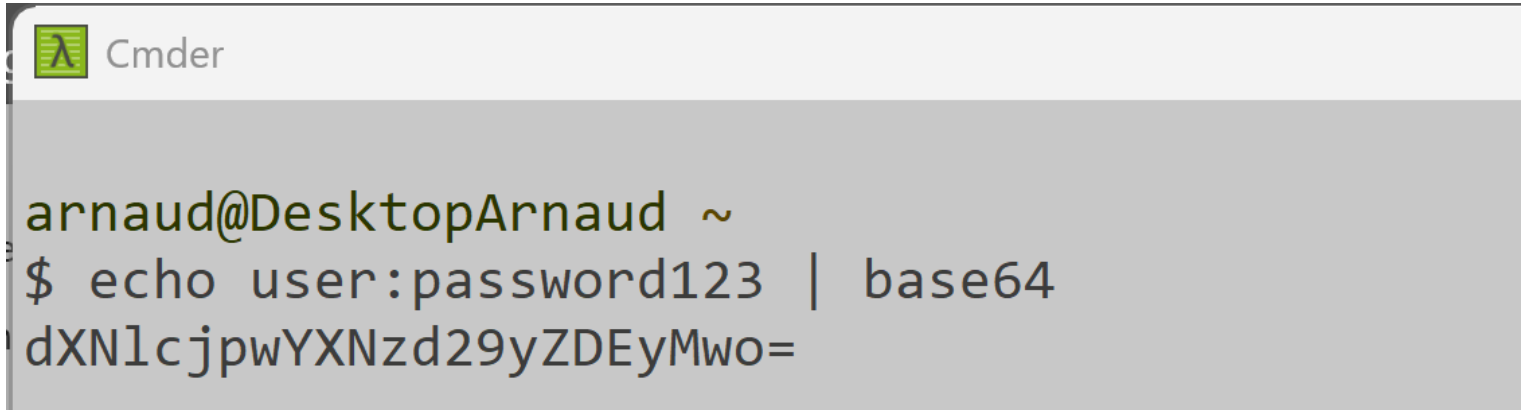
bash.exe

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

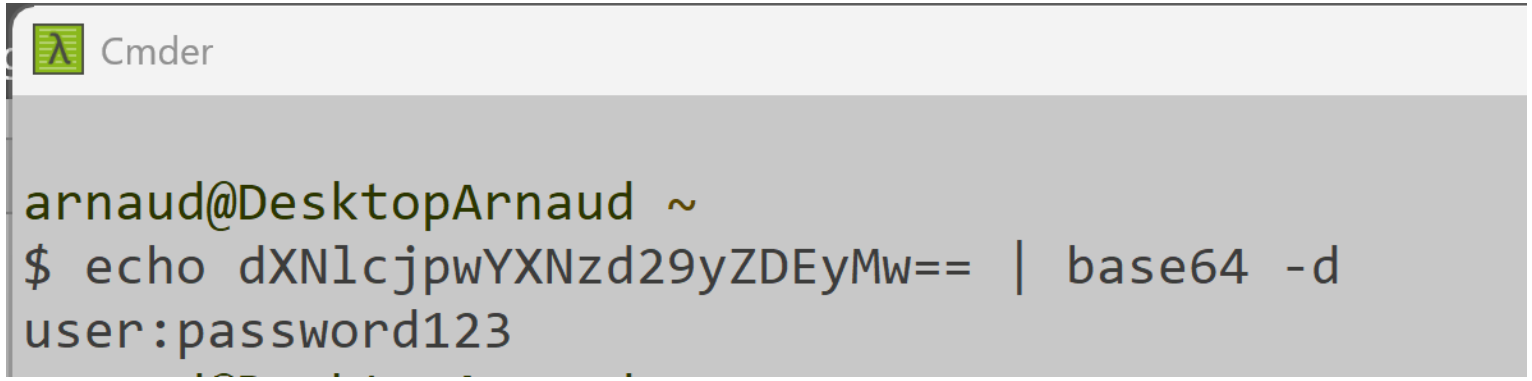


Encode / Decode base64 password, using "base64 -d"

echo user:password123 | base64



echo dXNlcjpwYXNzd29yZDEyMw== | base64 -d



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

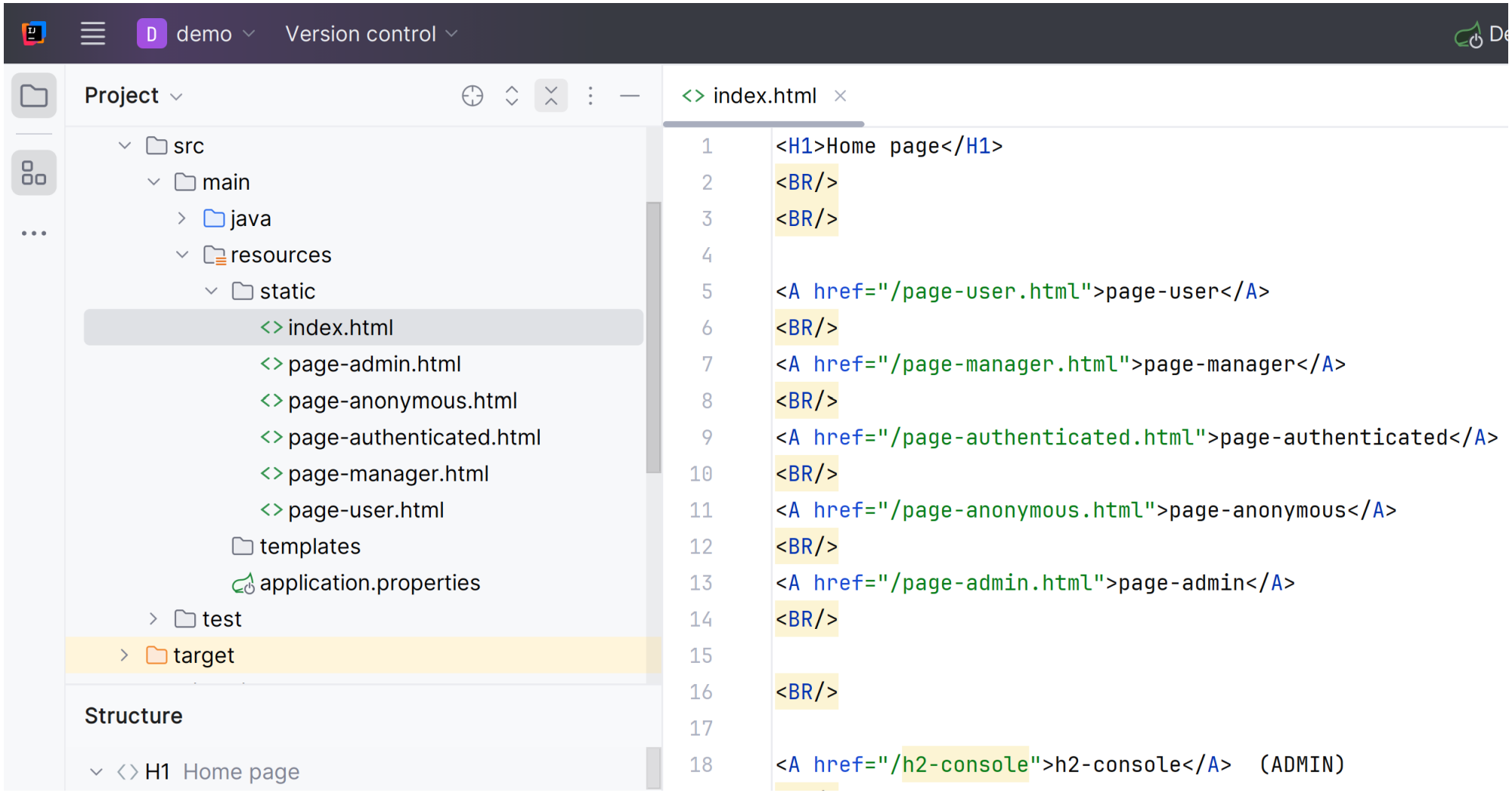
Confidentiality/Encryption

Backend-end: storing password in Database

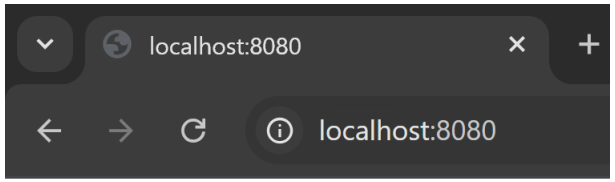
Cryptographic "Hash" function, Salt

Who is "John The Ripper" ?

Target: pages*.html with different Roles

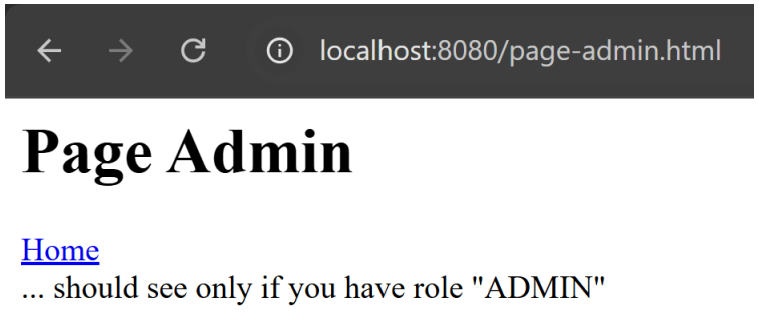


"/index.html" menu with anonymous access

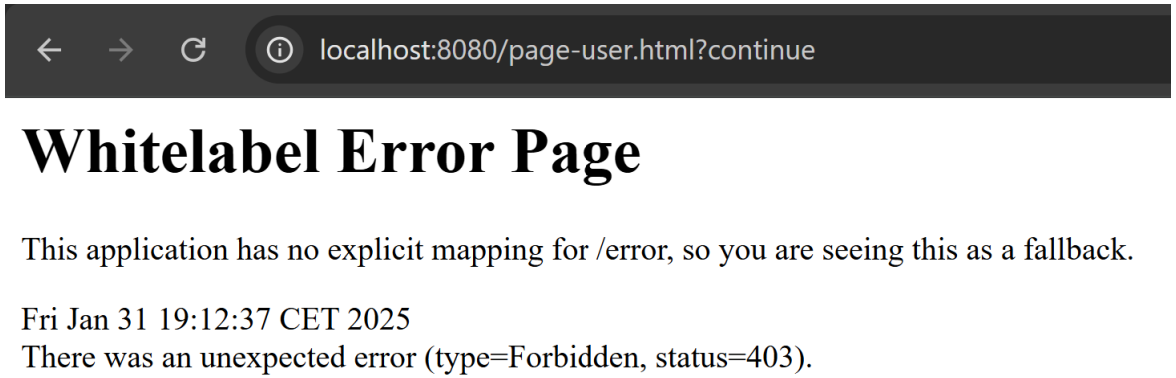



```
<H1>Home page</H1>
<BR/>
<BR/>

<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




```
<H1>Page Admin</H1>

<A href="/">Home</A>
<BR/>

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



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]

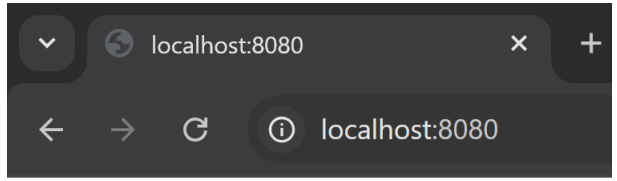
```
private static void configureHttpAuth(
        AuthorizeHttpRequestsConfigurer<HttpSecurity>.AuthorizationManagerRequestMatcherRegistry requests) {
    requests
        .requestMatchers(HttpMethod.GET, "*.css", "*.js", "*.png").permitAll()
        .requestMatchers("/", "/index.html").permitAll()
        .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

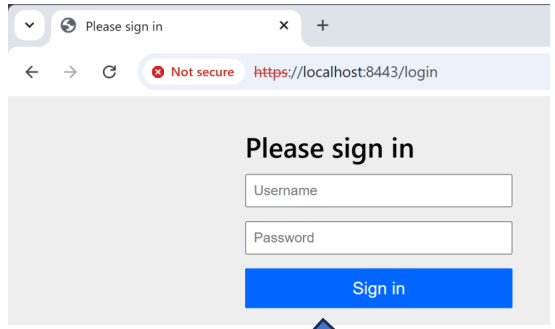
/index.html
(anonymous)

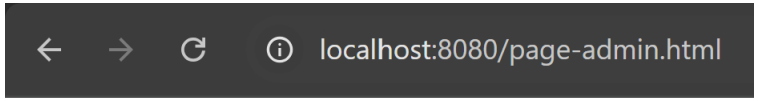


Home page

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

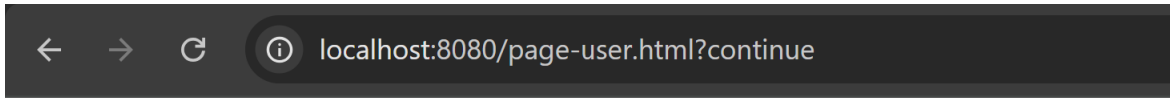
[h2-console](#) (ADMIN)
[logout](#)





Page Admin

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

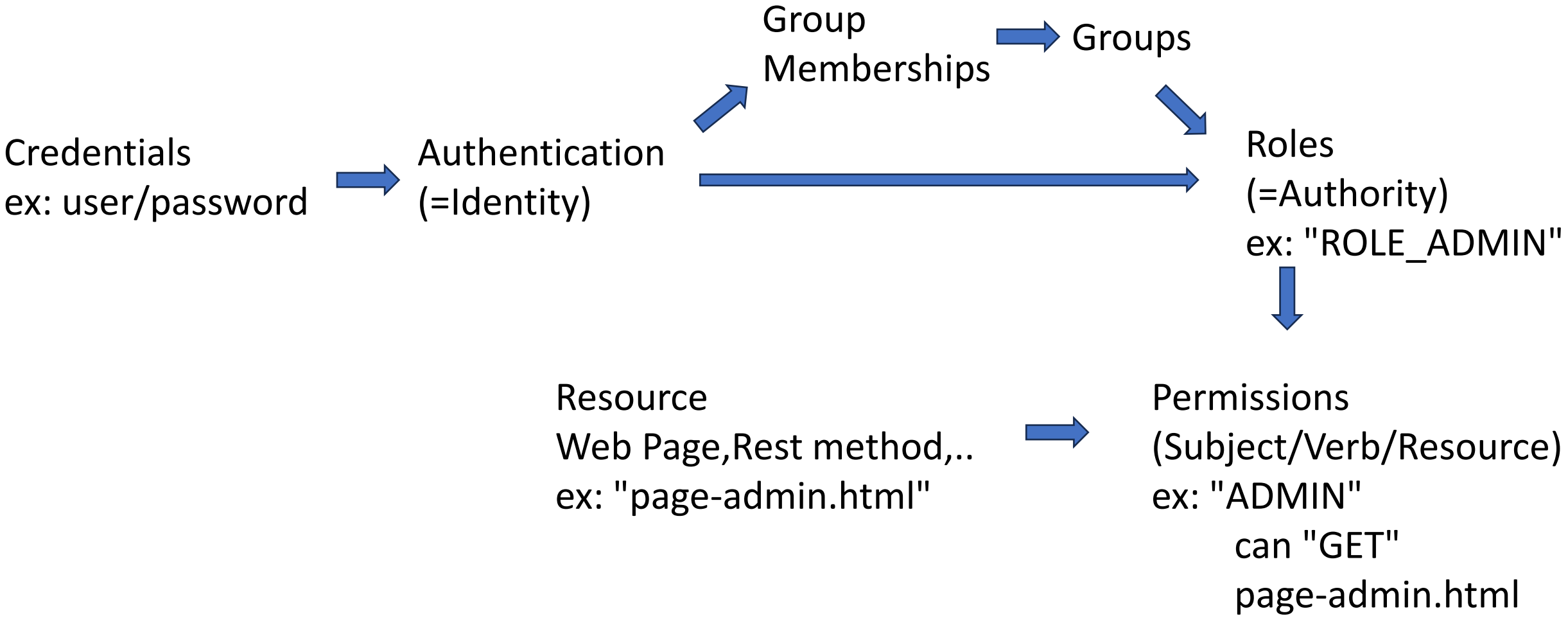


Whitelabel Error Page

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..


Credentials
ex: user/password
Authentication
(=Identity)
Group
Memberships
Groups
Roles
(=Authority)
ex: "ROLE_ADMIN"
Resource
Web Page,Rest method,..
ex: "page-admin.html"
Permissions
(Subject/Verb/Resource)
ex: "ADMIN"
can "GET"
page-admin.html

Outline

Authentication

Authorization - Permission

Confidentiality/Encryption

Backend-end: storing password in Database

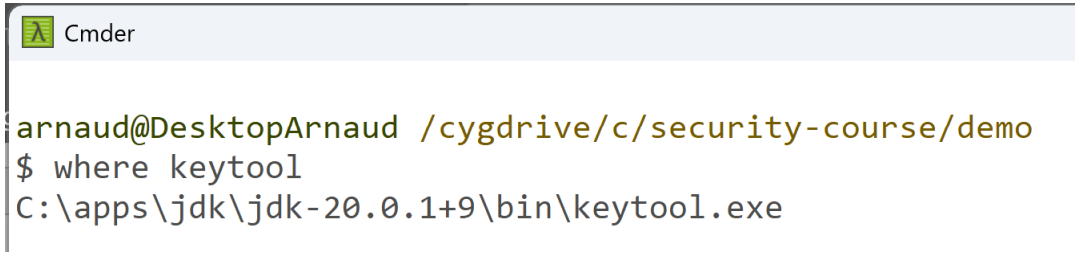
Cryptographic "Hash" function, Salt

Who is "John The Ripper" ?

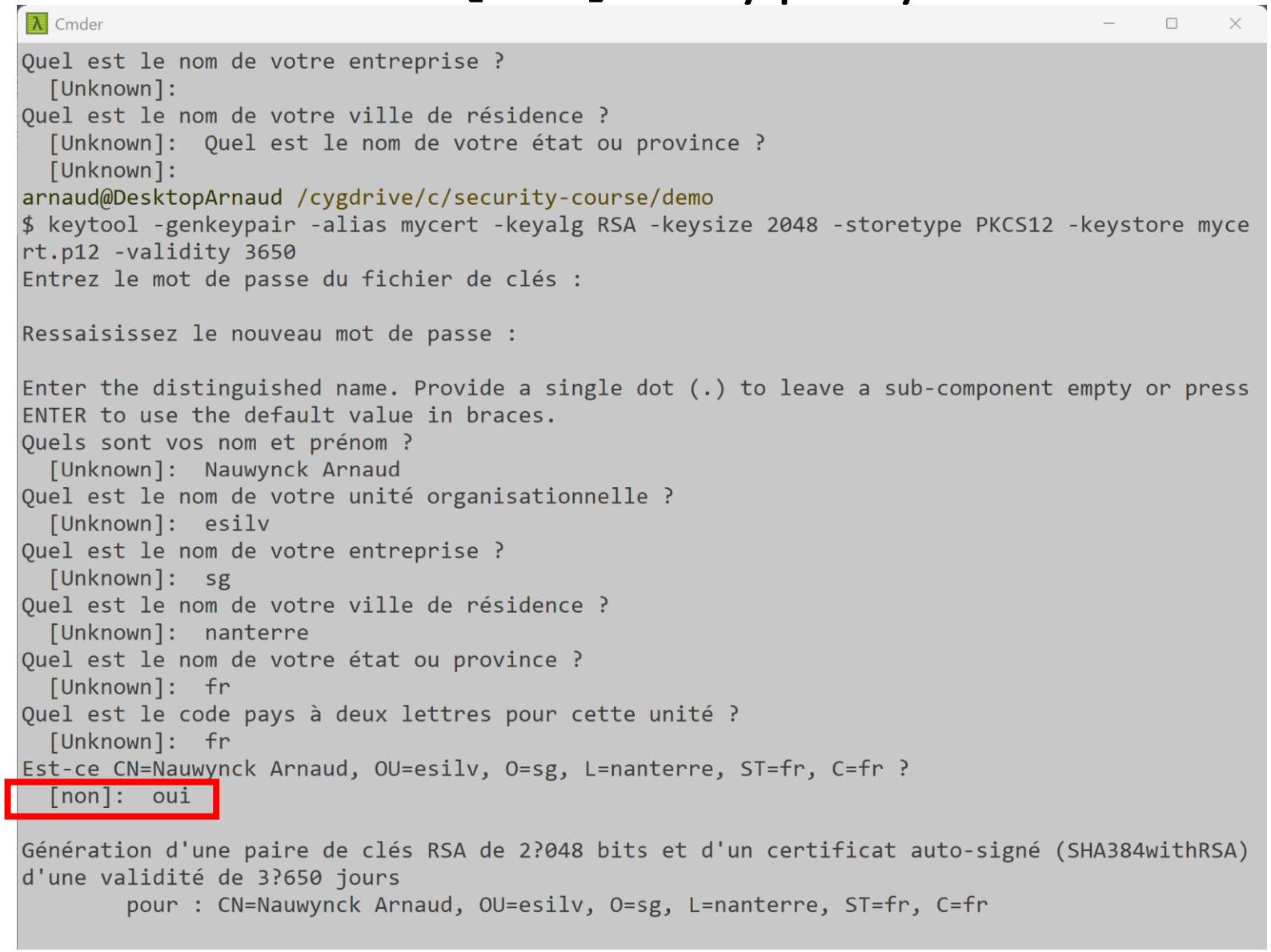
Generate your Https self-signed certificate

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

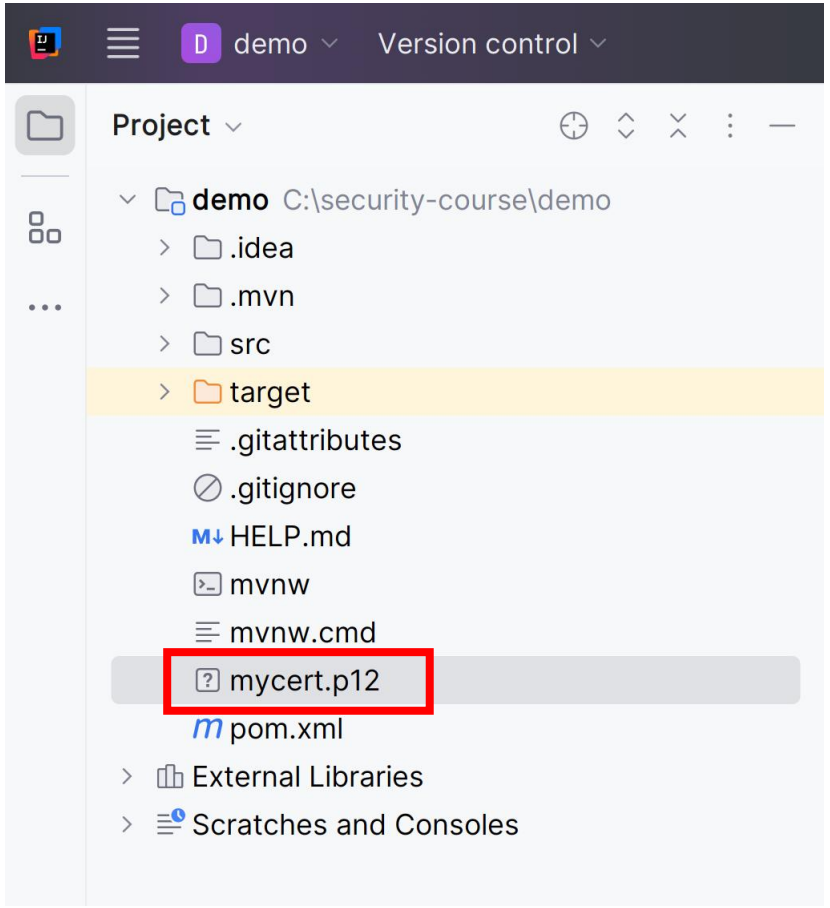
Troubleshooting ?
"keytool" is a tool in your jdk/bin directory
check "where keytool" => it should be found in your "PATH" environment variable
check "echo $PATH"
... to fix, use windows "env" to edit your environement variables, or export PATH="$PATH:$JAVA_HOME/bin"



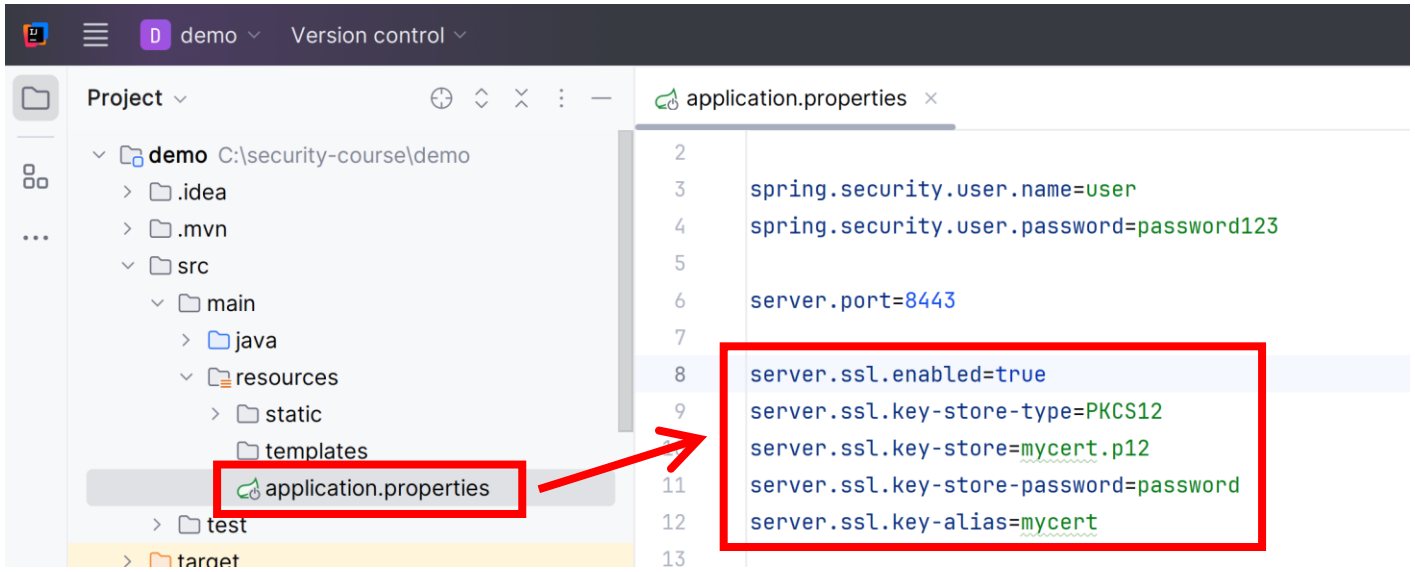
interactive answer to prompt (any is ok) lastly "Is it CN=.... ? " [no]: type yes !!



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



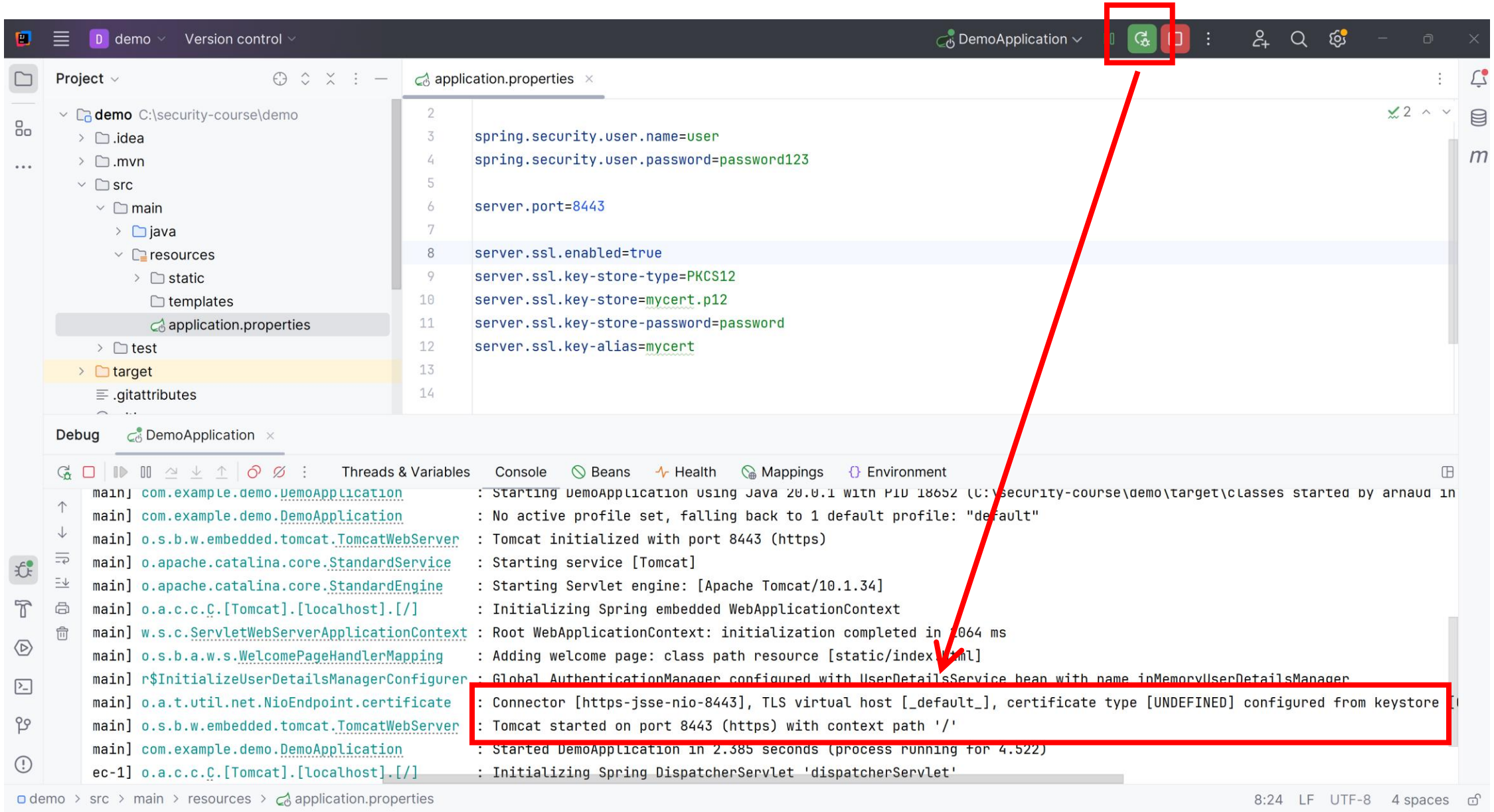
Edit src/main/resources/application.properties to configure SSL



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

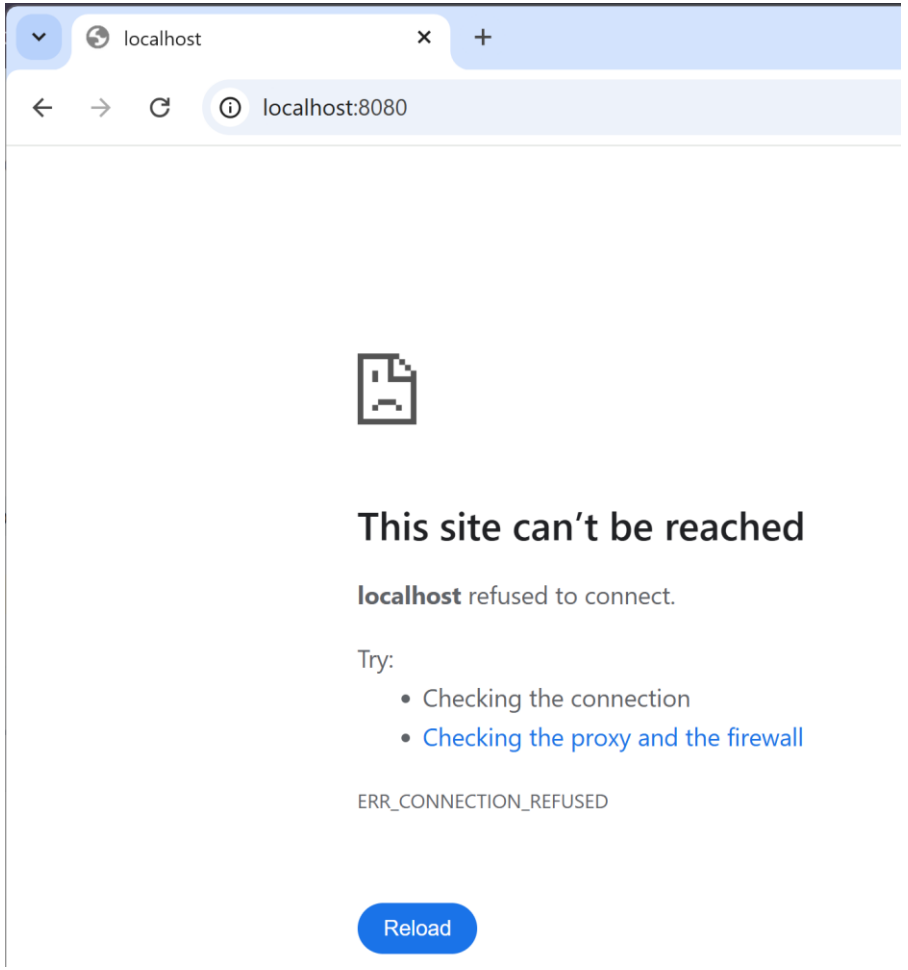
Relaunch server, check new console log



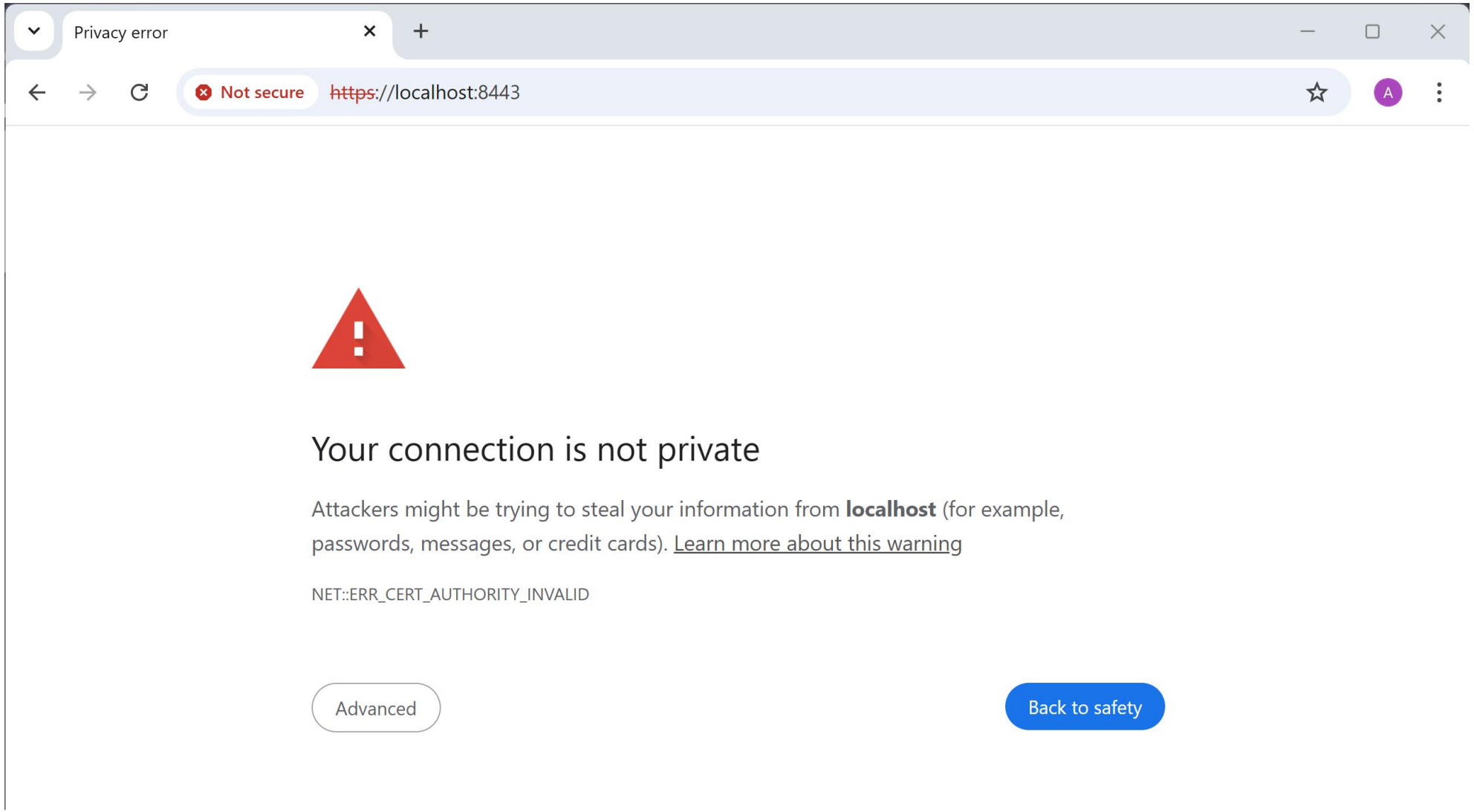
port changed 8080 (default for http) -> 8443 (default for https)

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 ?

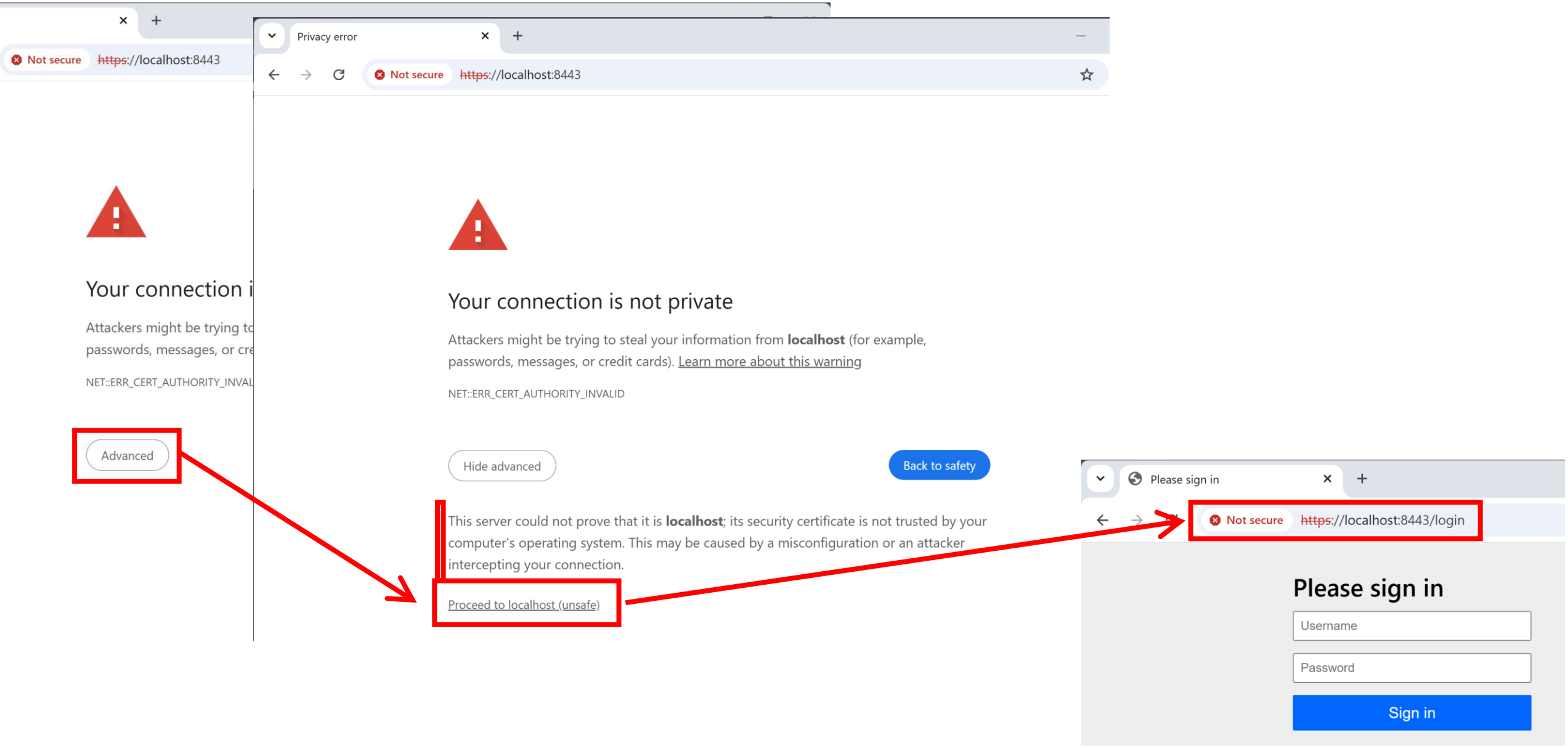
=> pay 200 EURO/year to have a valid certificate ?

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

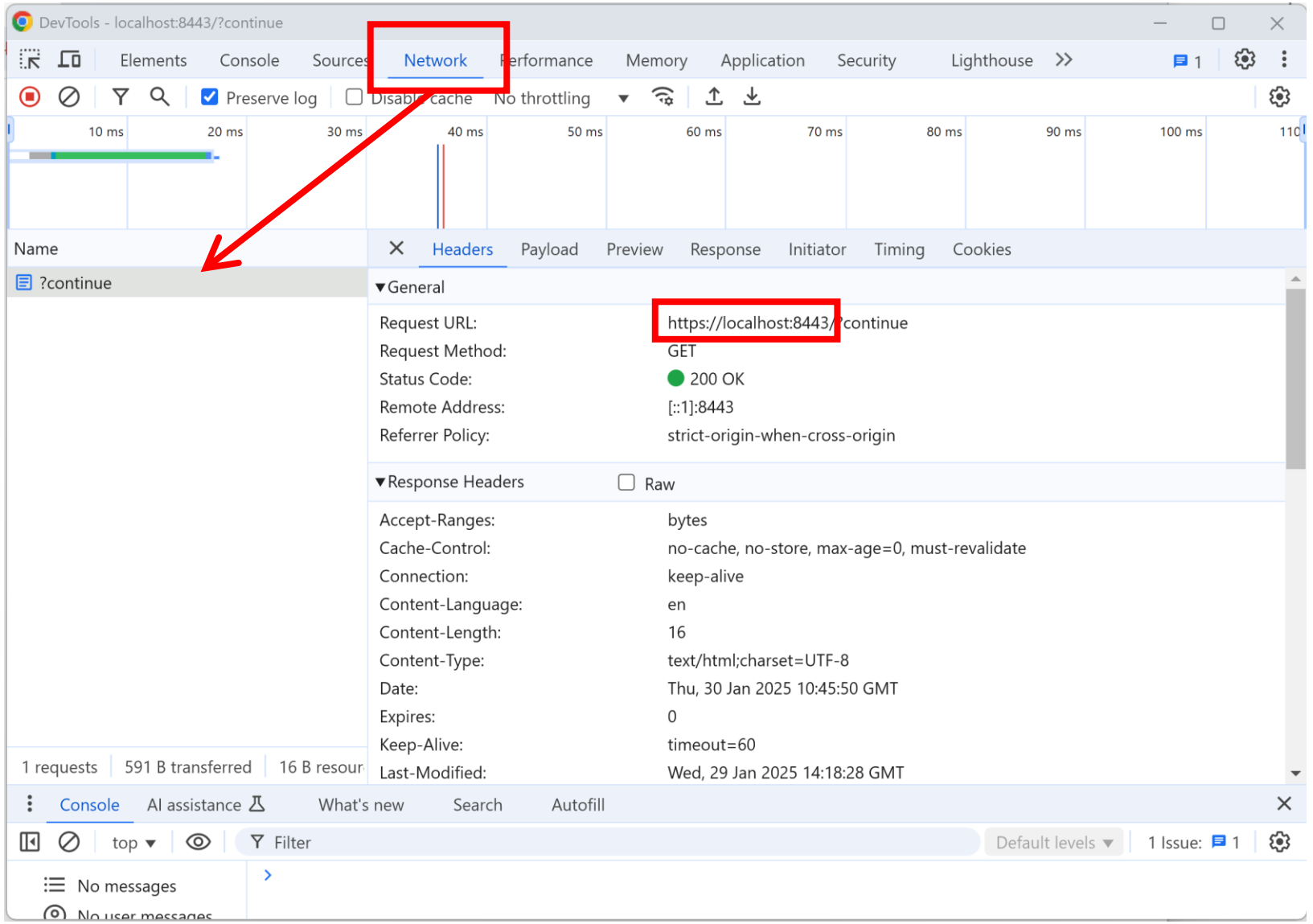
=> recompile chrome to know "yourself" as valid authority

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

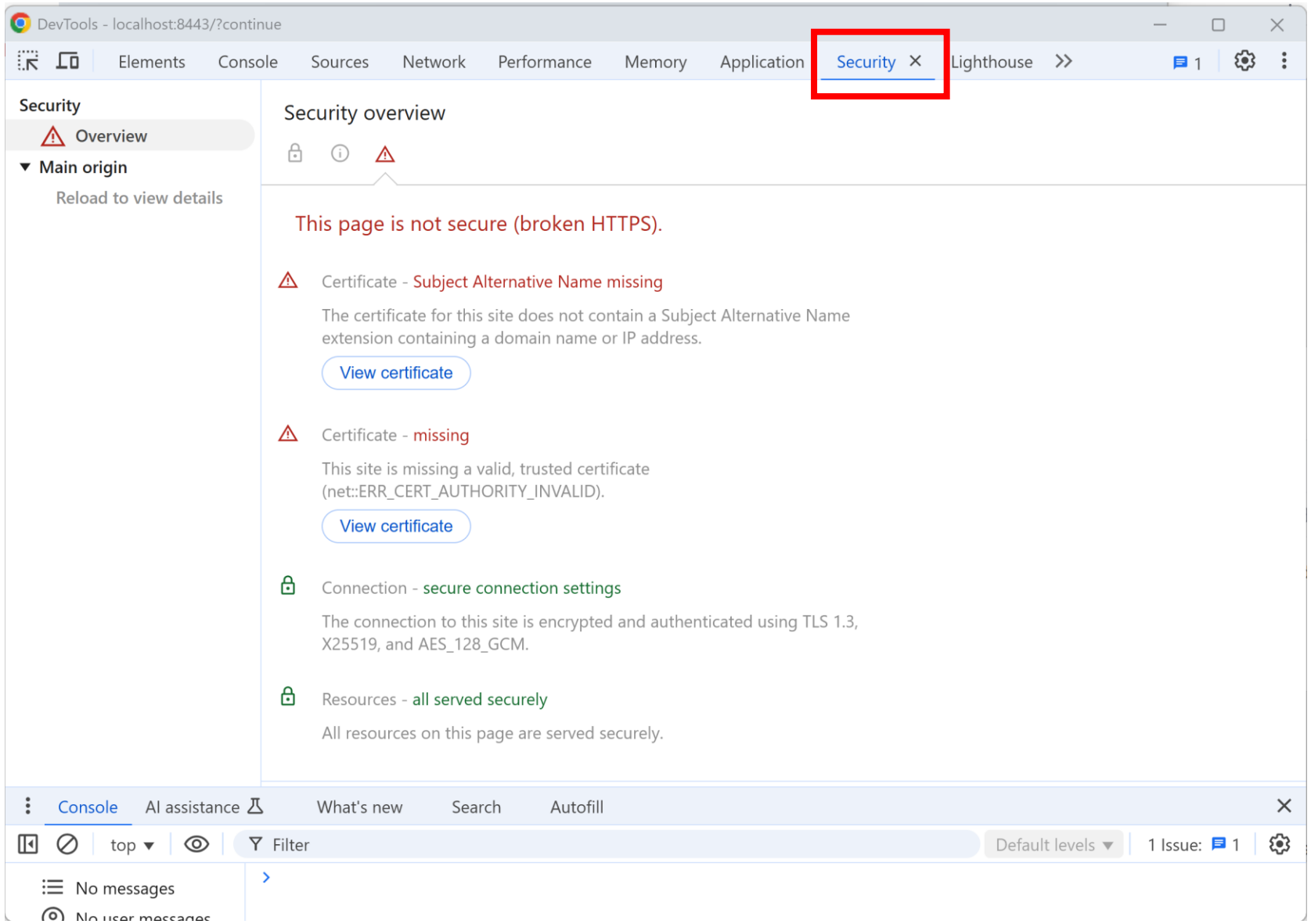
Advanced > Proceed to (unsafe)



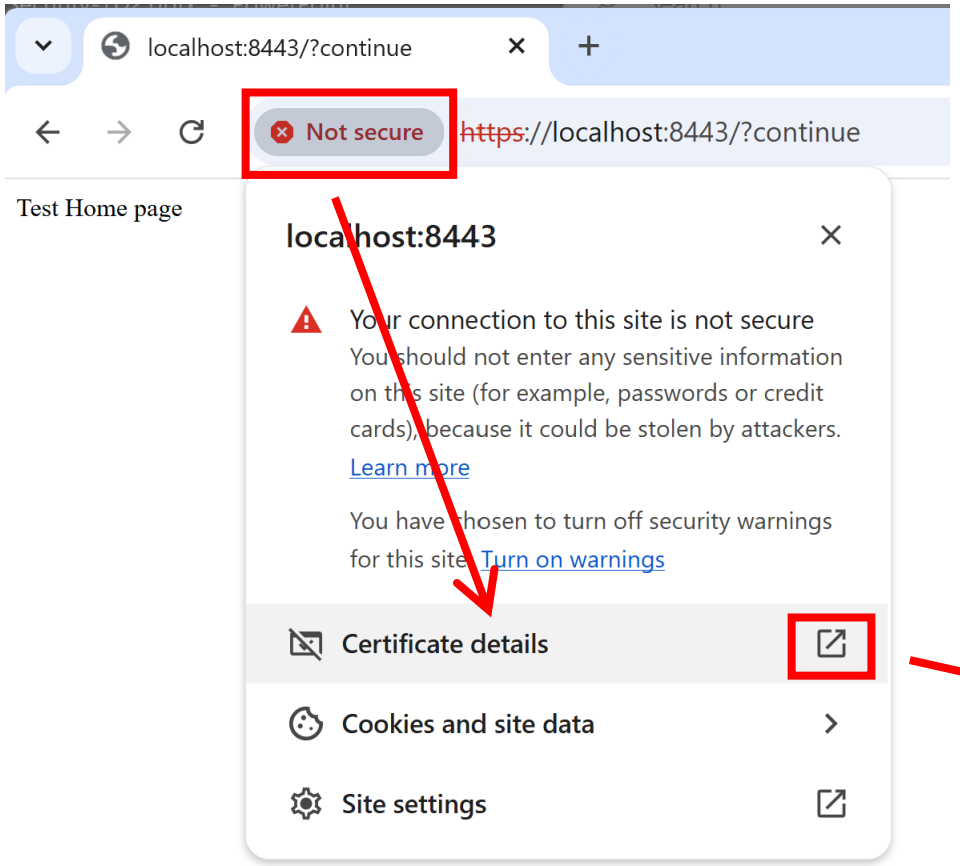
Chrome DevTools Network requests
... same on "https://"



Chrome DevTools "Security" tab



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

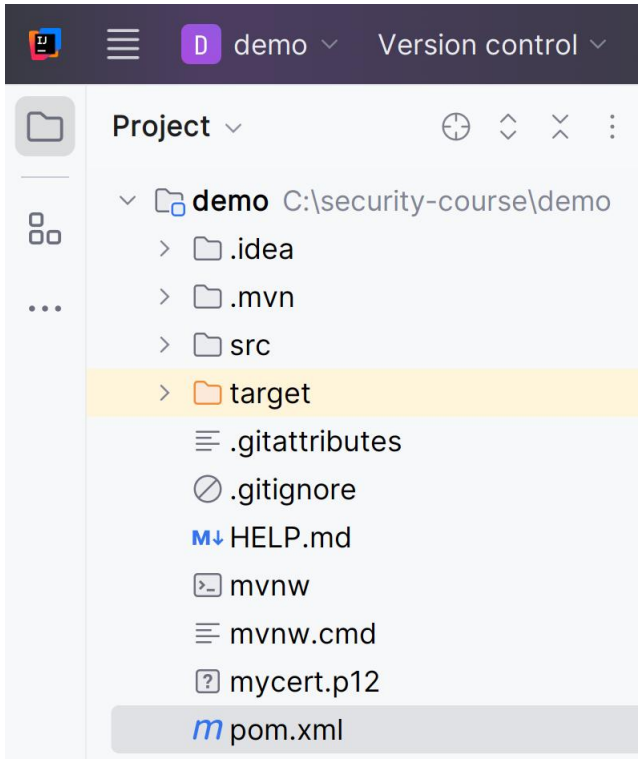
Confidentiality/Encryption

Backend-end: storing password in Database

Cryptographic "Hash" function, Salt

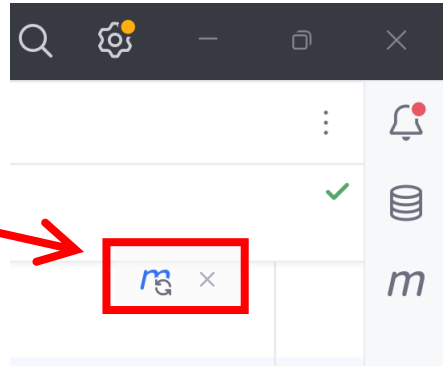
Who is "John The Ripper" ?

Edit pom.xml
Add maven dependency for Database inside <dependencies> ..

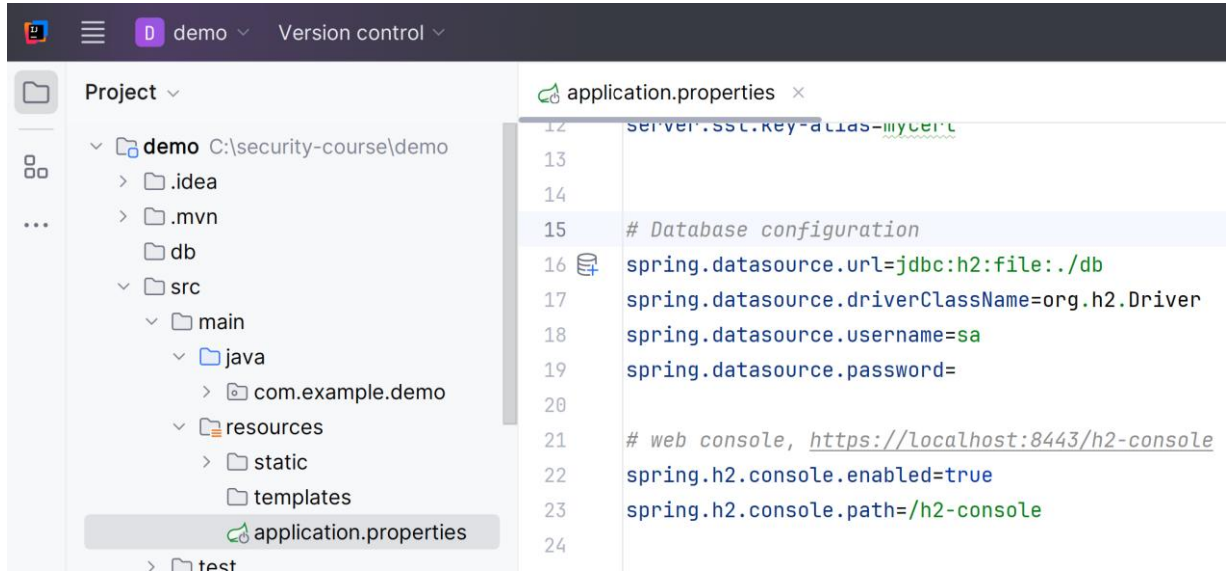



```
<dependency>
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
<dependency>
        <groupId>com.h2database</groupId>
        <artifactId>h2</artifactId>
</dependency>
```

then click on refresh maven



Edit application.properties add H2 config

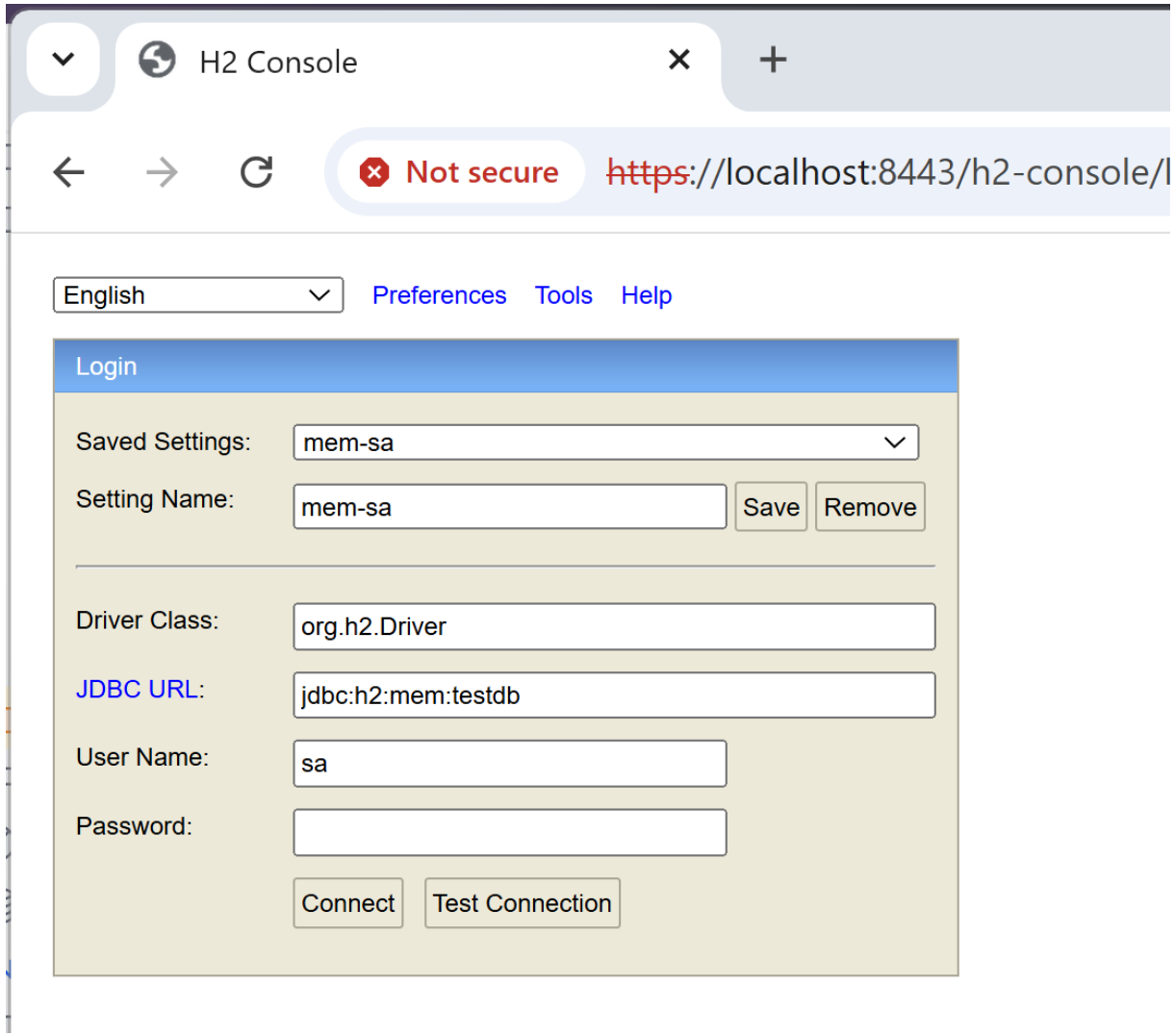



```
# Database configuration
spring.datasource.url=jdbc:h2:file:./db
spring.datasource.driverClassName=org.h2.Driver
spring.datasource.username=sa
spring.datasource.password=

# web console, see https://localhost:8443/h2-console
spring.h2.console.enabled=true
spring.h2.console.path=/h2-console
```

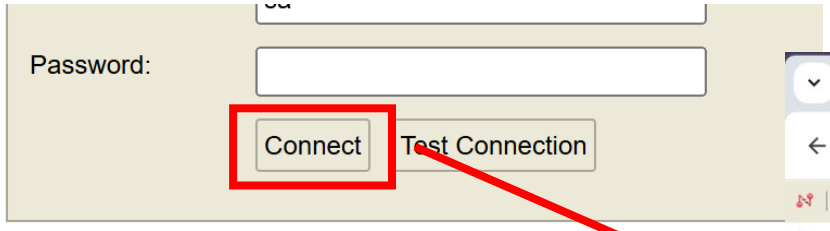
Create java class SecurityDbConfiguration

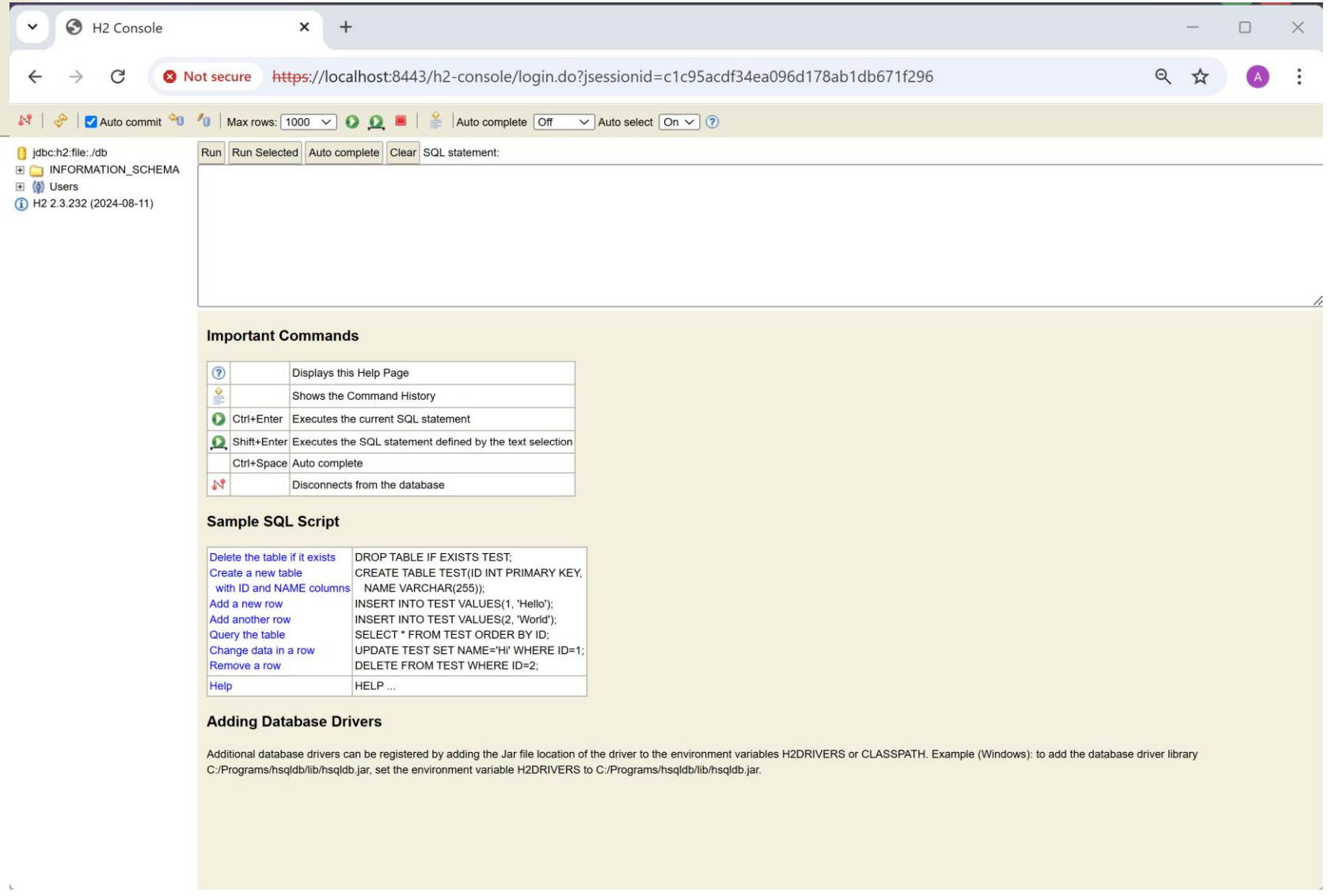
```
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.provisioning.JdbcUserDetailsManager;
import org.springframework.security.provisioning.UserDetailsManager;
import javax.sql.DataSource;

@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"



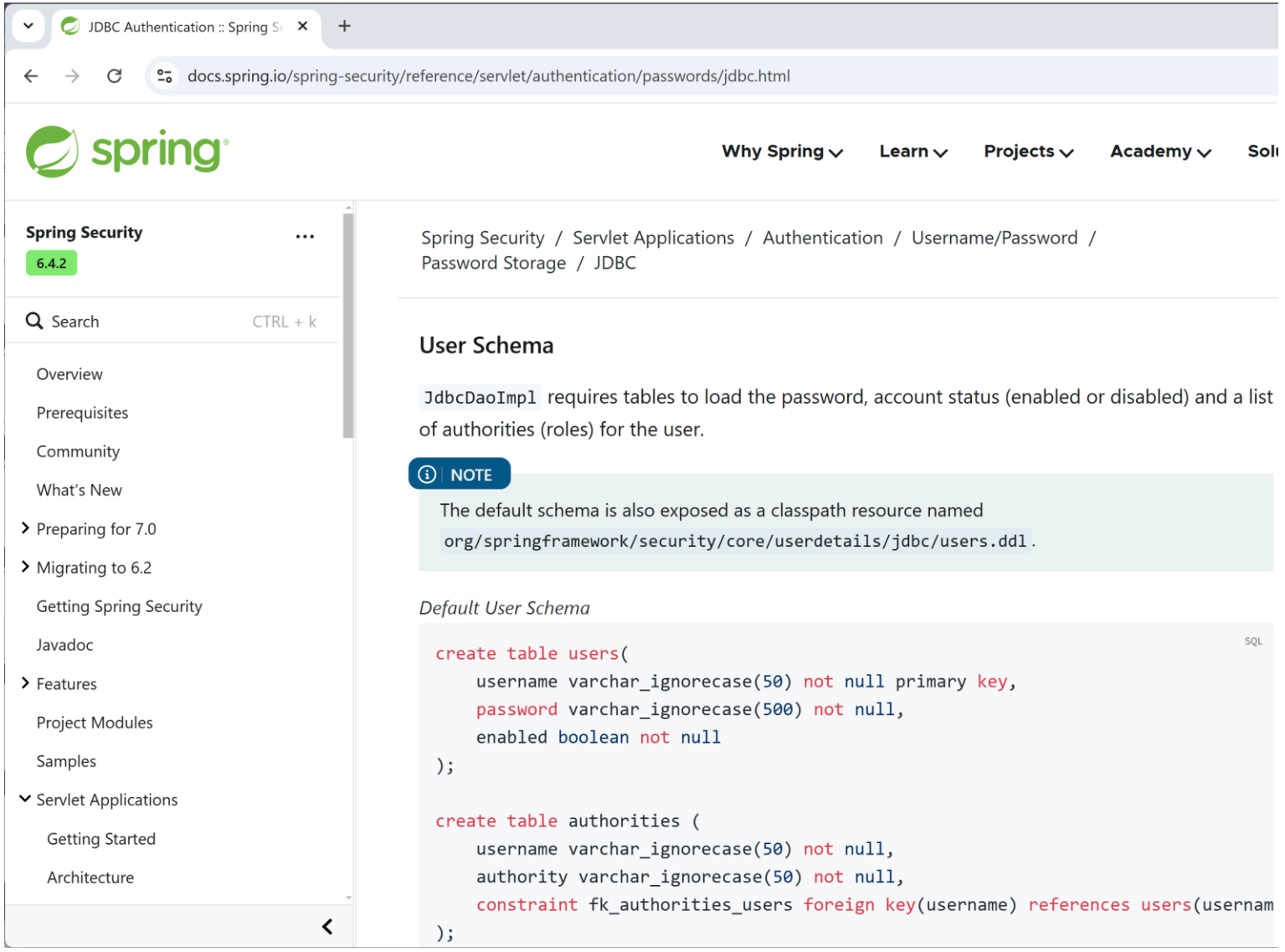


Copy&Paste Create Tables DDL

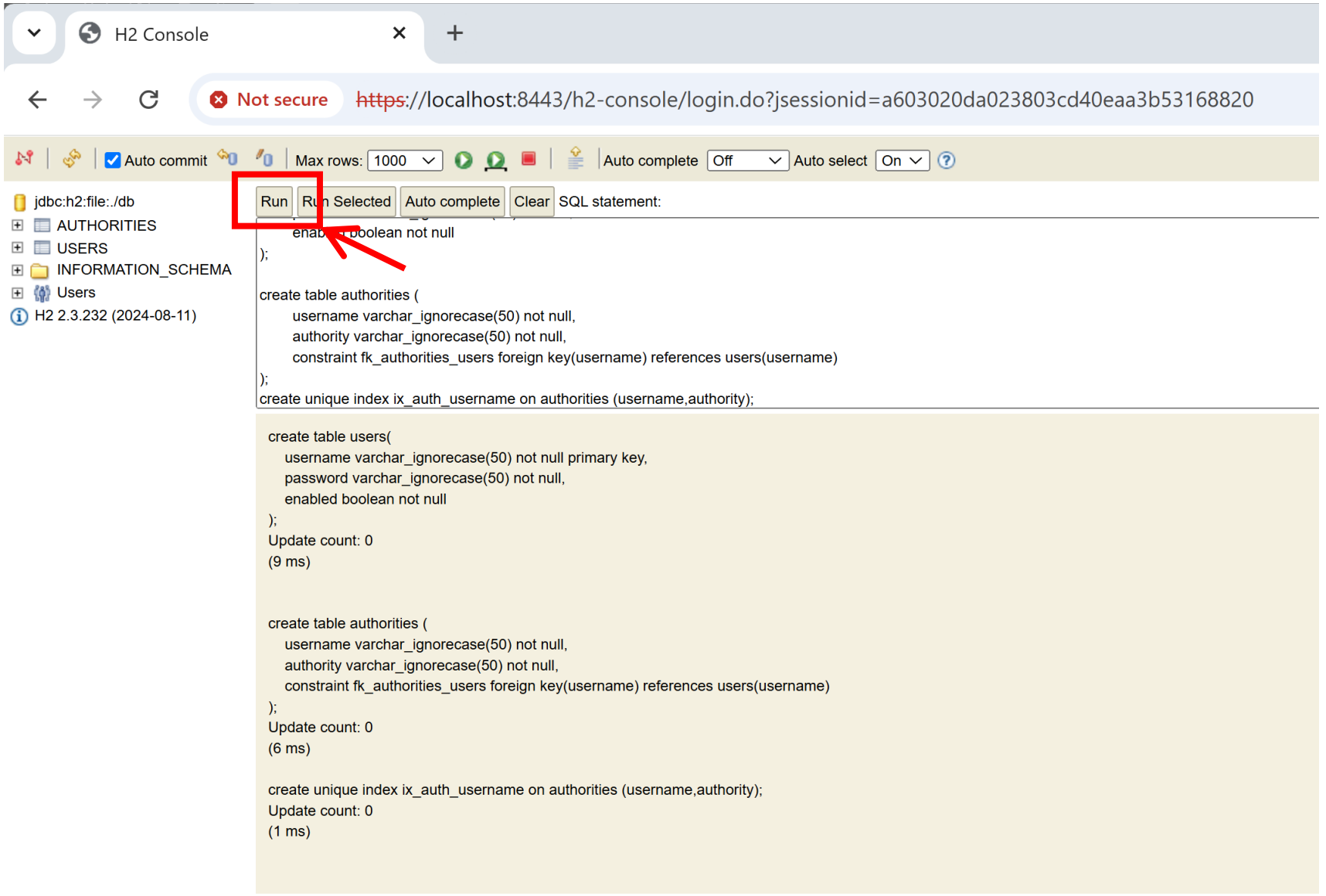
```
create table users(
        username varchar_ignorecase(50) not null primary key,
        password varchar_ignorecase(500) not null,
        enabled boolean not null
);

create table authorities (
        username varchar_ignorecase(50) not null,
        authority varchar_ignorecase(50) not null,
        constraint fk_authorities_users 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



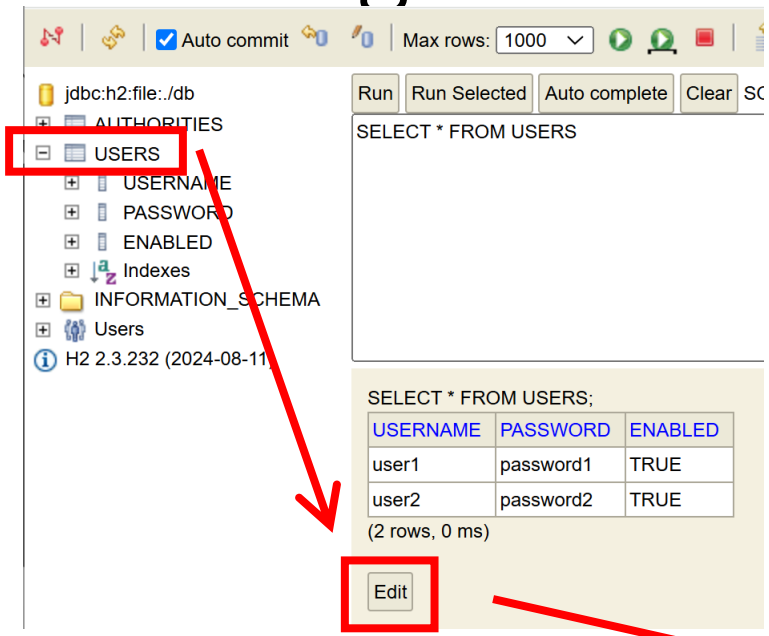
Execute Create Table DDL

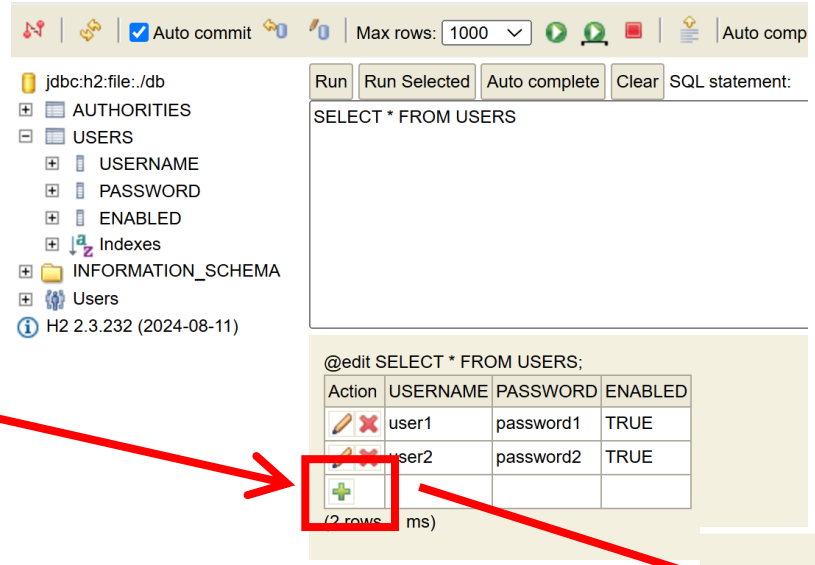


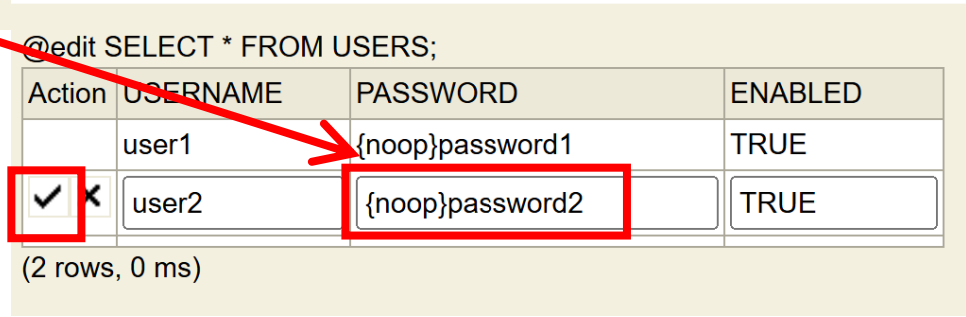
(Optionnal) DDL for Groups & Members

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

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

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

Insert users in database
Using H2 "New Row", edit, commit



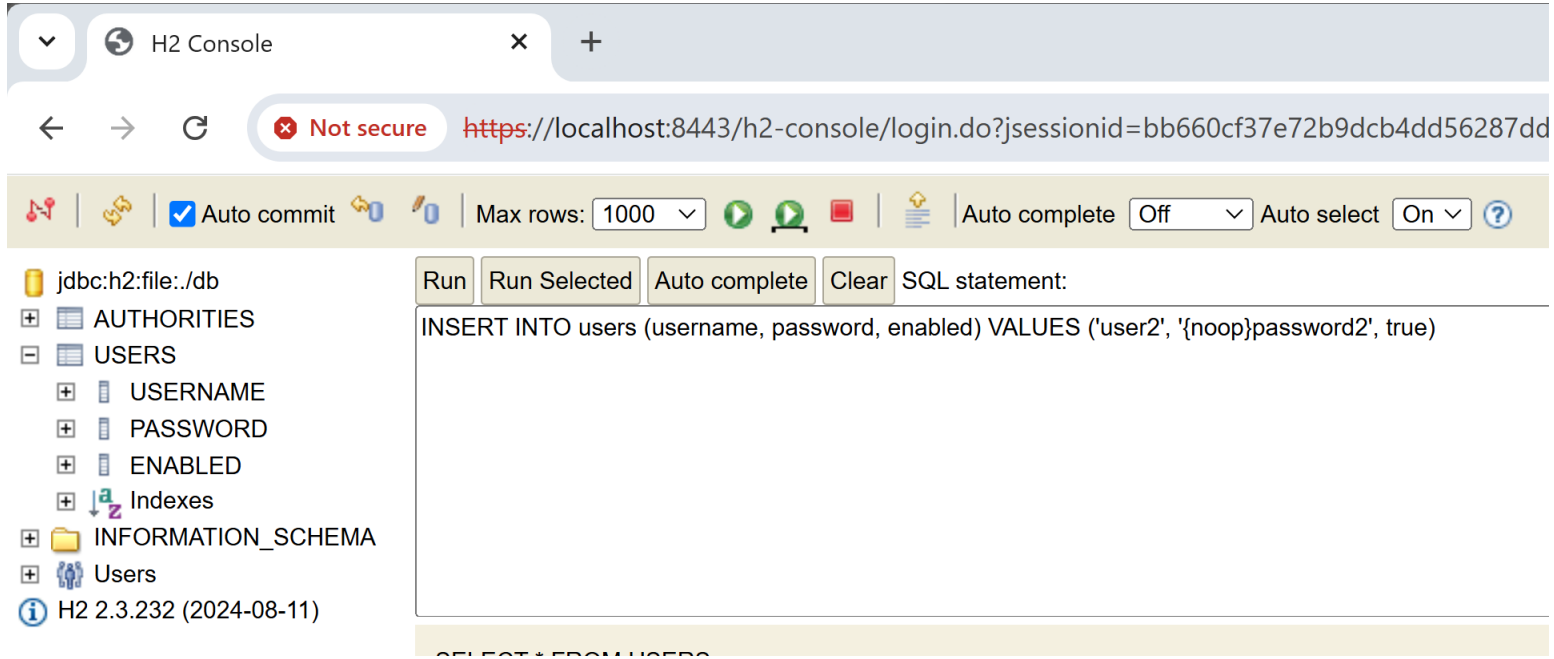




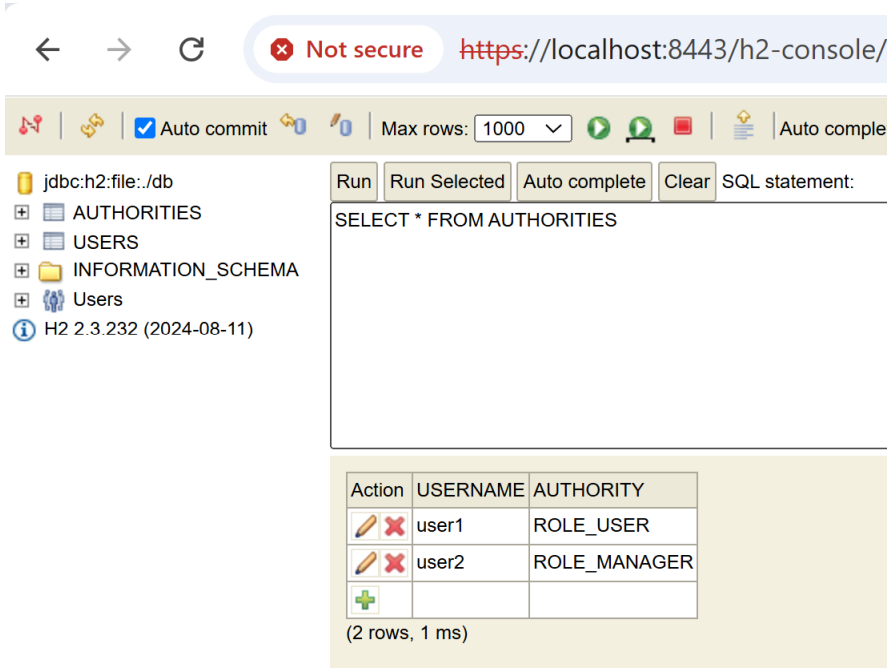
NOTICE: should prefix password with **{noop}**
(see next lesson for Hash and Salting)

INSERT INTO users... using SQL

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



Also add users a "role" (= AUTHORITY)

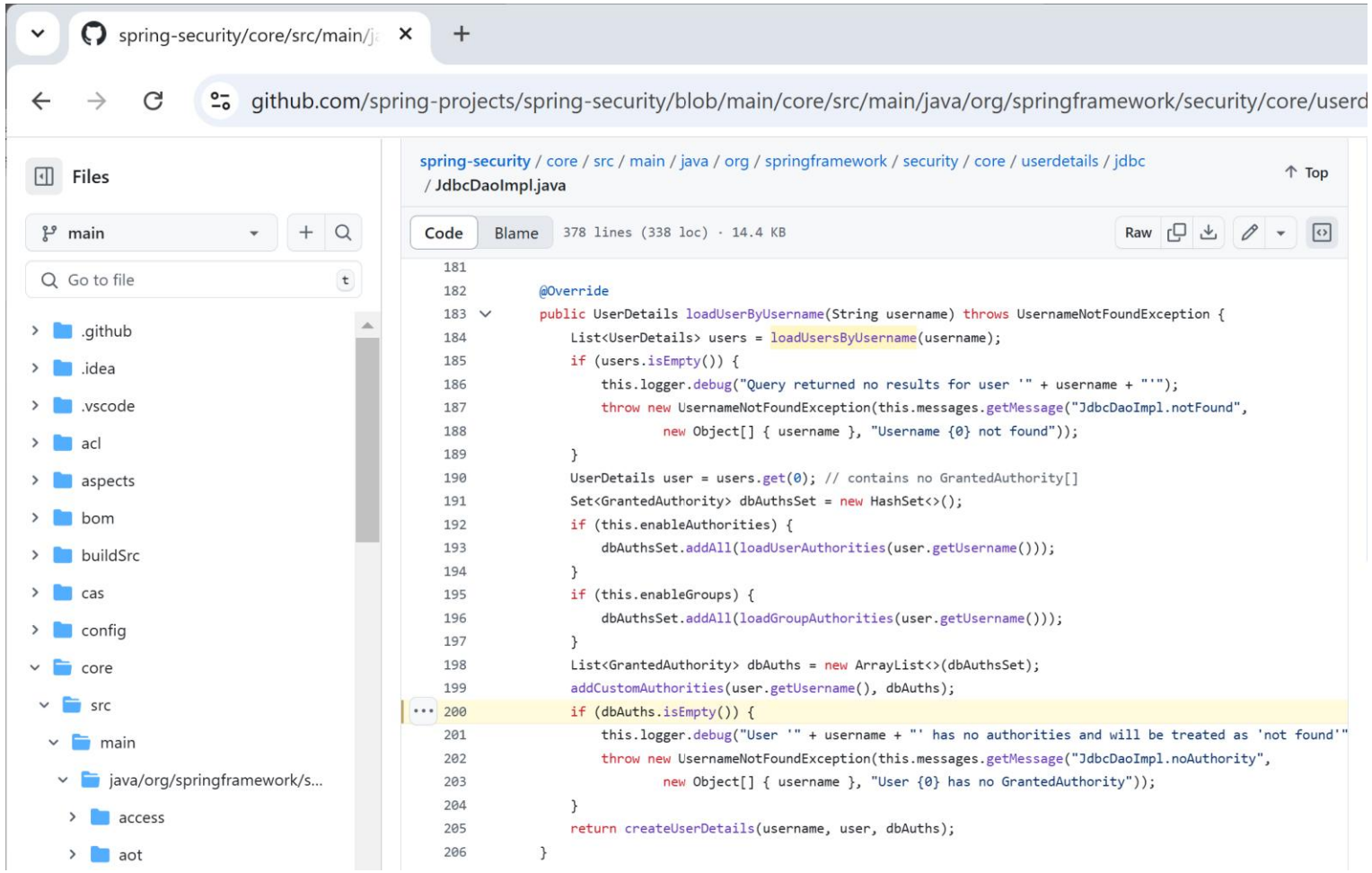


INSERT INTO authorities (username, authority) VALUES ('user1', 'ROLE_USER')

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



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

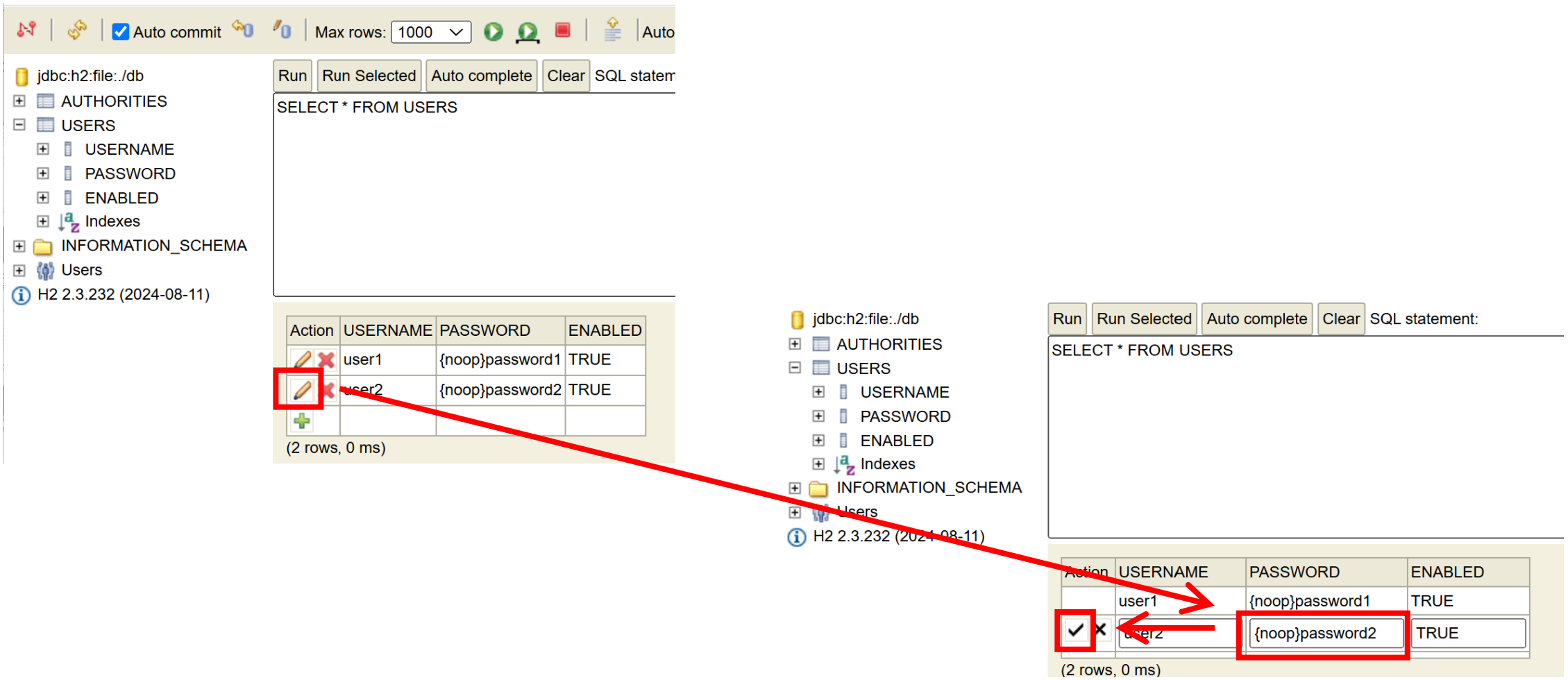
Debug DemoApplication

Threads & Variables | Console | Beans | Health | Mappings | Environment

```
    at org.springframework.security.config.annotation.web.configuration.WebMvcSecurityConfiguration$CompositeFilterChainProxy.doFilter(WebMvcSecurityConfiguration.java

2025-01-30T16:26:13.137+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. Pleas
    at org.springframework.security.crypto.password.DelegatingPasswordEncoder$UnmappedIdPasswordEncoder.matches(DelegatingPasswordEncoder.java:307) ~[spring-security-c
    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-co
    at org.springframework.security.authentication.dao.AbstractUserDetailsAuthenticationProvider.authenticate(AbstractUserDetailsAuthenticationProvider.java:147) ~[spr
    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) ~[sprin
```

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"



Relaunch & Test ..

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    }
```

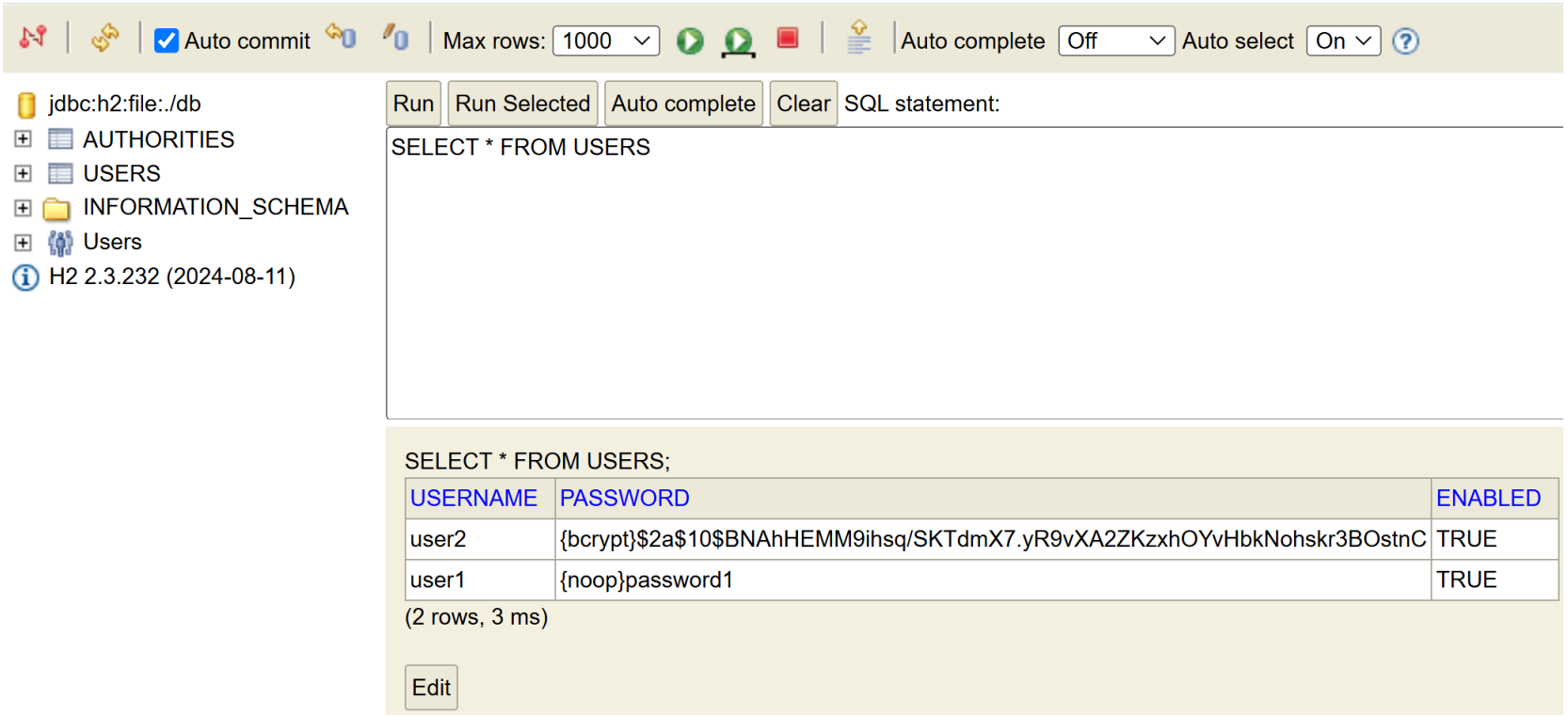
```
19    public UserUpdateRestController(UserDetailsManager userDetailsManager) {
20        this.userDetailsManager = userDetailsManager;
21        // for debug (uncomment to execute once, or use curl
22        updatePassword(new UserPasswordRequestDTO( username: "user2",  newPassword: "password2"));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

Or using Rest api

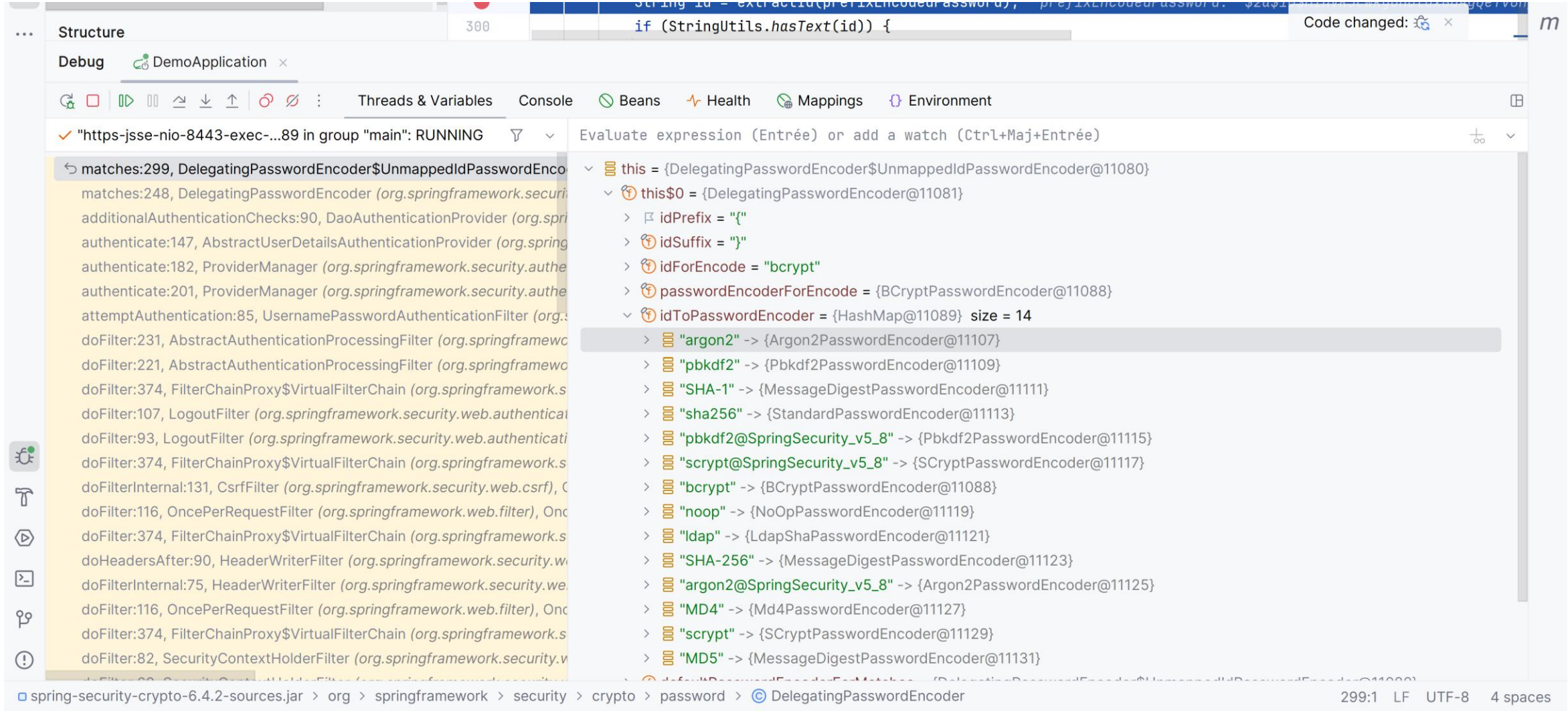
```
curl -v -k -u "user1:password1" -H 'Content-Type: application/json' \
     -X POST https://localhost:8443/api/v1/users/update-resalt-password \
     -d '{"username": "user1", "newPassword": "password1"}'
```

check HTTP 200 + Check in Debug Console :

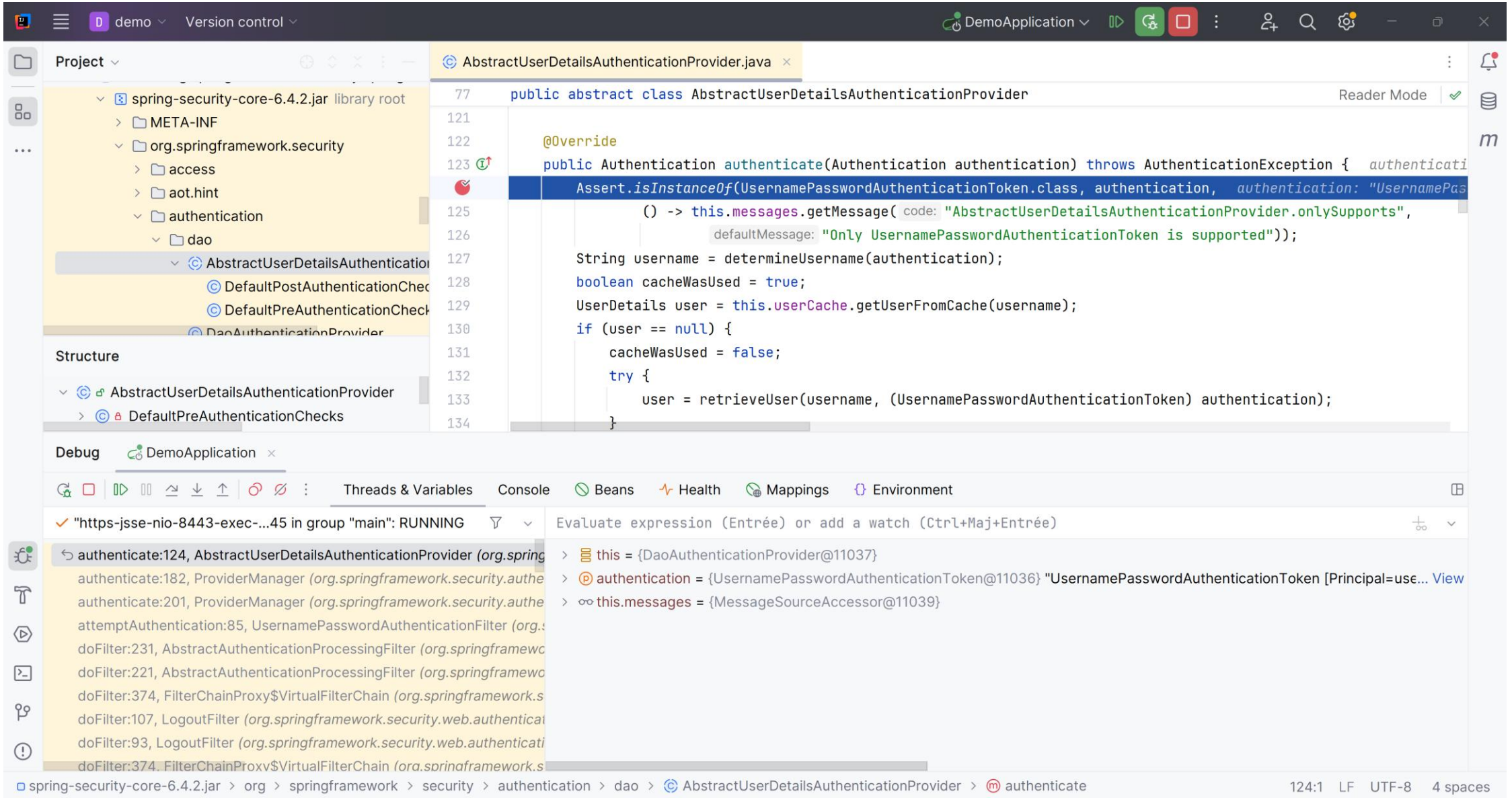
Showing encoded password



ANNEXE... Screenshot Debugging "encoders"



ANNEXE Screenshot Debugging authentication



Outline

Authentication

Authorization - Permission

Confidentiality/Encryption

Backend-end: storing password in Database

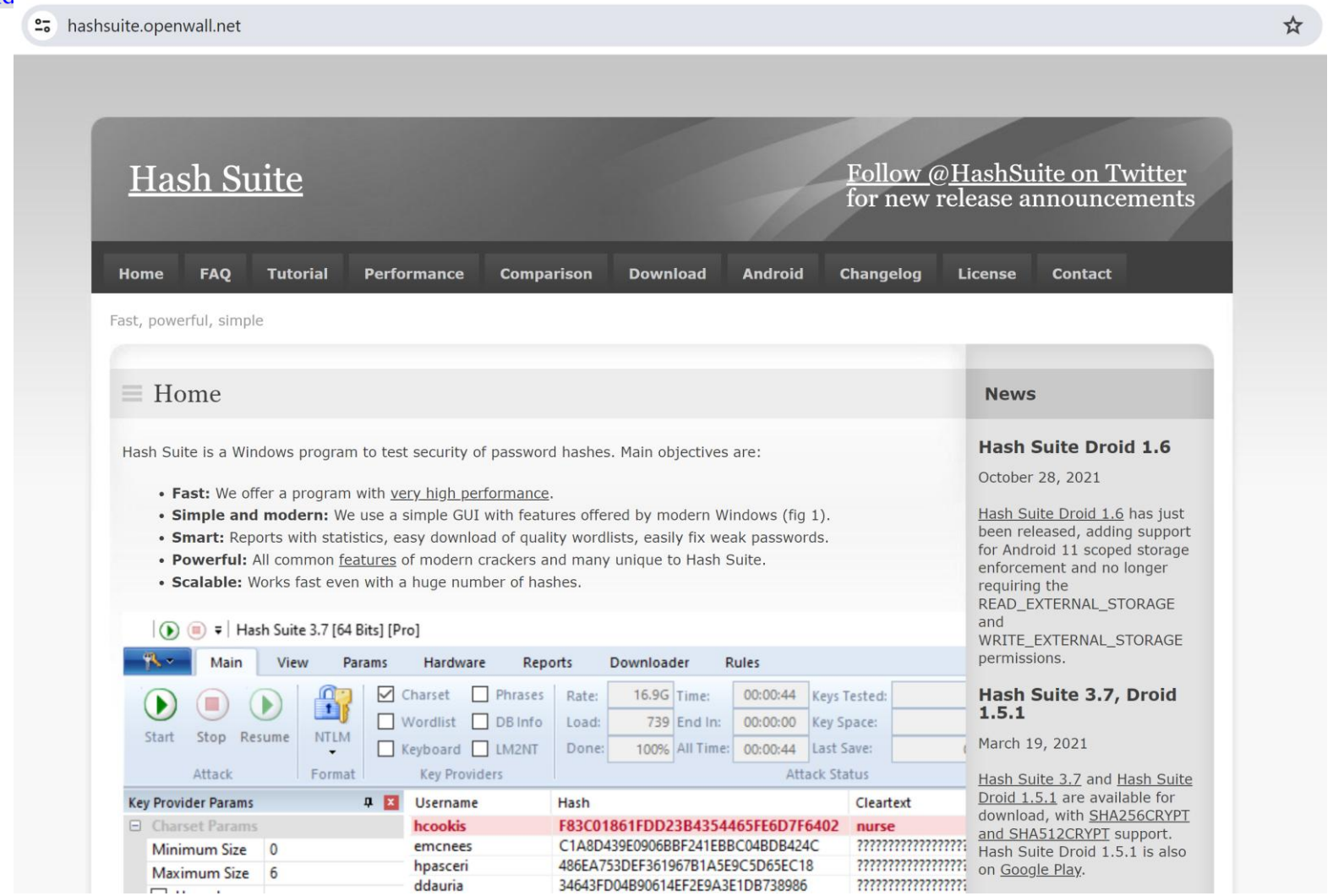
Cryptographic "Hash" function, Salt

Who is "John The Ripper" ?

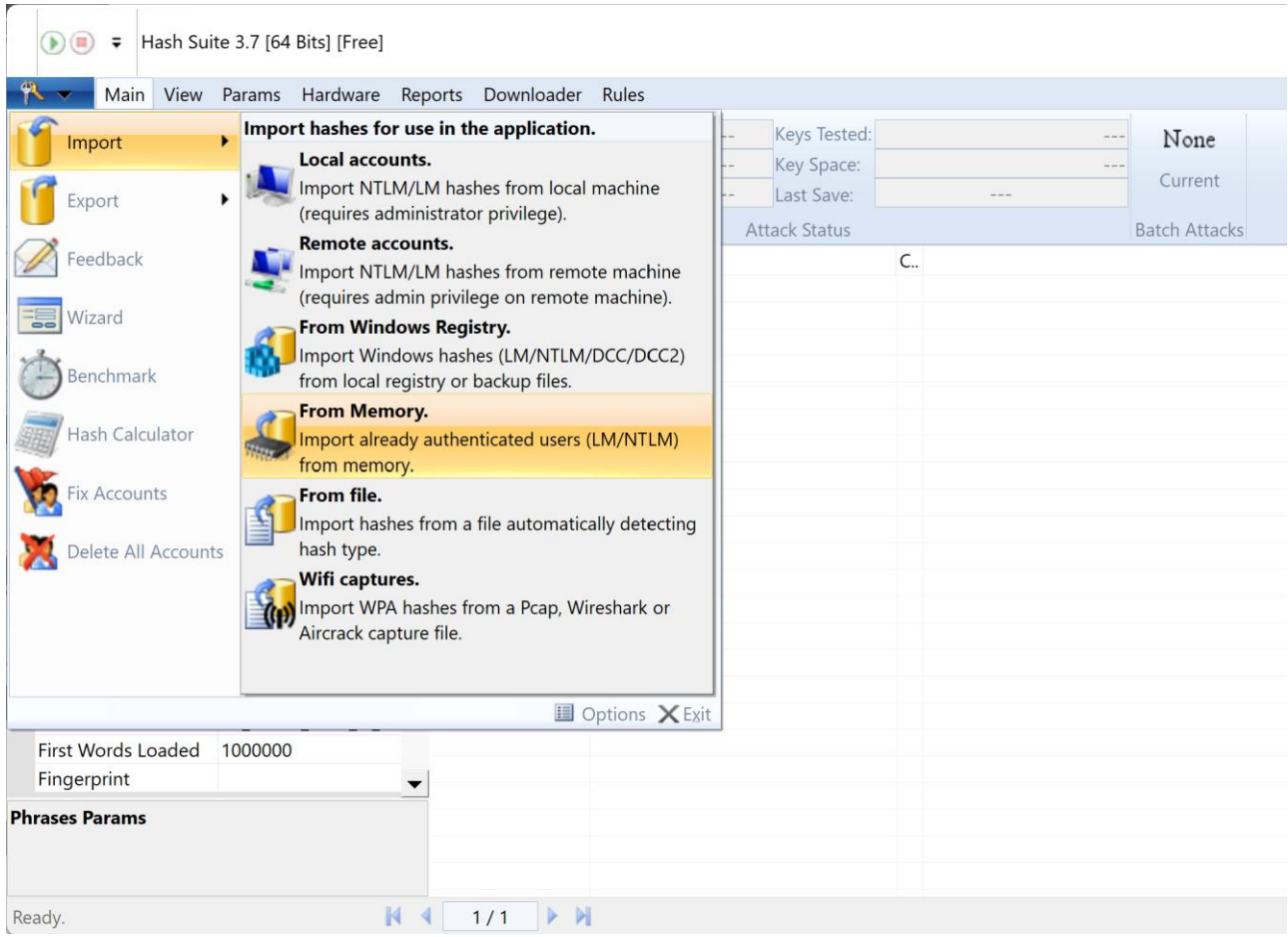
Download John The Ripper / HashSuite

- John the Ripper *Pro* for Linux
- John the Ripper *Pro* for macOS
- **On Windows, consider Hash Suite** (develc
- On Android, consider Hash Suite Droid

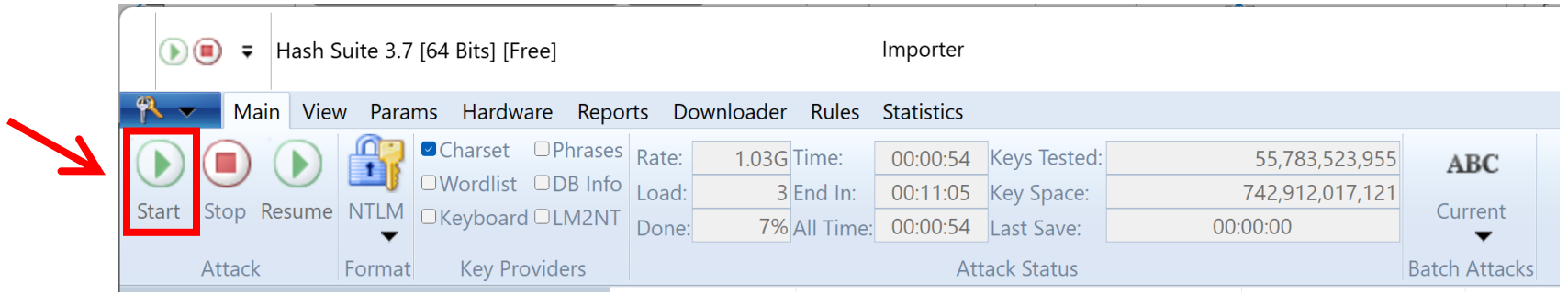
https://hashsuite.openwall.net



Step 1/ Import Your Windows Hashed Password From Memory



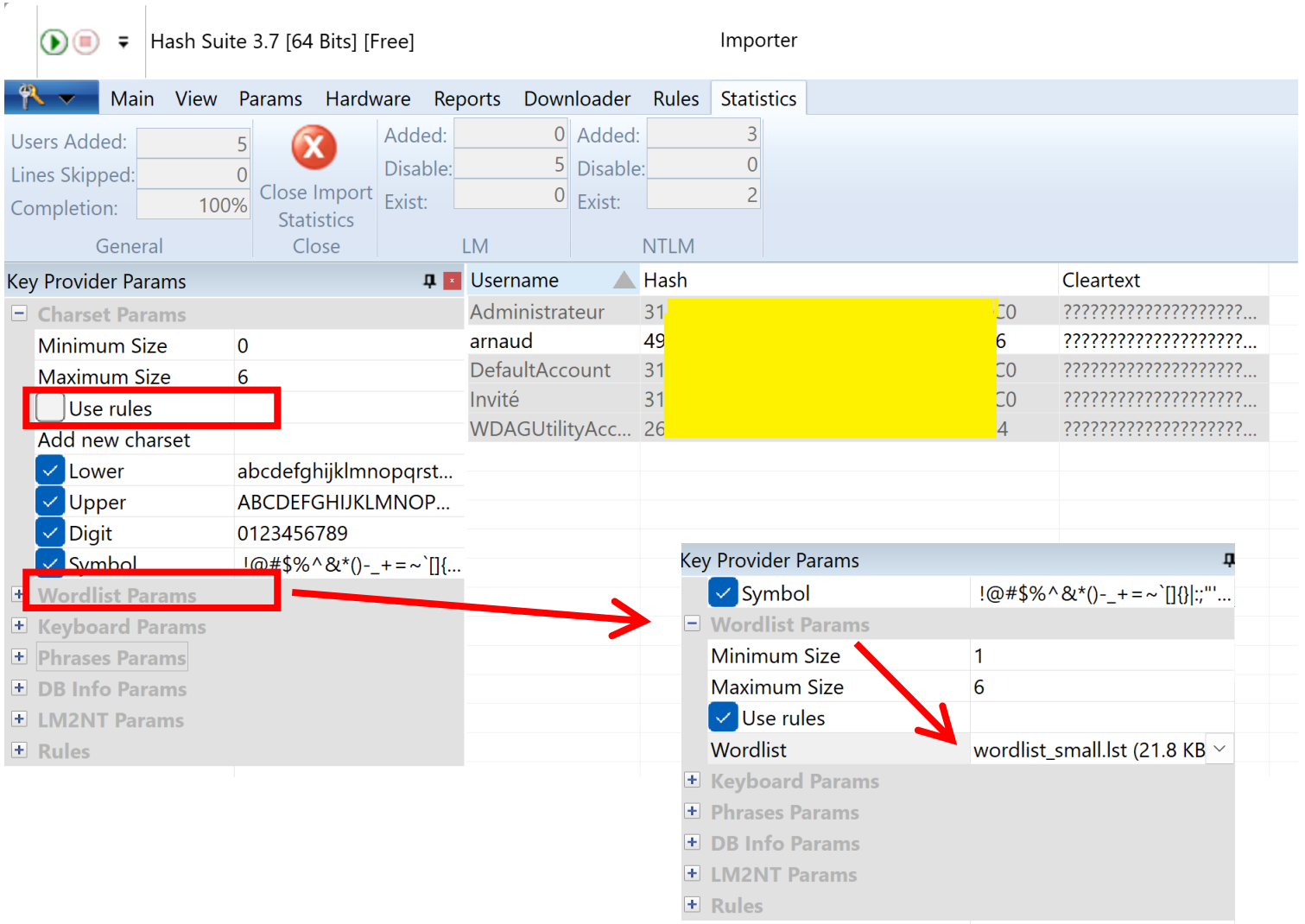
Step 2/ Test your password strength
Run 5-10mn (default settings)



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



Summary

Authentication

Authorization - Permission

Confidentiality/Encryption

Backend-end: storing password in Database

Cryptographic "Hash" function, Salt

Who is "John The Ripper" ?

Transmit Password to Http

password=>authentication=>role

password not in clear over internet

storing password ...

... storing hashed+salted password

finding password from hashed

Questions ?

arnaud.nauwynck@gmail.com