Spring Security — Step by Step (Simple Hindi)

Ye PDF teri provided code base ke basis par Spring Security ka poora flow step-by-step samjhata hai. Simple aur practical language me—seedha point par.

1) Login se Process Start hota hai
- User /login page par username + password enter karta hai aur form submit karta hai.
- Tere SecurityConfig me login page aur username/password field names defined hain:
  formLogin.loginPage("/login");
  formLogin.usernameParameter("username");
  formLogin.passwordParameter("password");

2) Spring Security Filter Chain
- Spring request ko filter chain se pass karta hai; authorize rules check hote hain:
  .requestMatchers("/login", "/css/**", "/js/**").permitAll()
  .requestMatchers("/student/**").hasAnyRole("STUDENT", "ADMIN")
  .requestMatchers("/faculty/**").hasAnyRole("FACULTY", "ADMIN")
  .requestMatchers("/admin/**").hasRole("ADMIN")

3) AuthenticationManager → AuthenticationProvider
- Form submit hone par AuthenticationManager, DaoAuthenticationProvider ko call karta hai.
- Tere config me DaoAuthenticationProvider set hua hai aur usme CustomUserDetailsService diya hai.

4) CustomUserDetailsService.loadUserByUsername
- Ye method DB se user fetch karta hai: userRepo.findByUsername(username)
- Agar user milta hai to UserMaster (jo UserDetails implement karta hai) return hota hai, warna exception.

5) UserMaster implements UserDetails
- getAuthorities() user ke roles ko SimpleGrantedAuthority list me convert karta hai.
- isEnabled(), isAccountNonLocked() etc. authentication checks ke liye use hote hain.
- password field DB me hashed form me hona chahiye (BCrypt recommended).

6) Password Verification
- DaoAuthenticationProvider provided PasswordEncoder (BCryptPasswordEncoder) se raw aur encoded password match karta hai.
- Agar matches → authentication success; agar nahi → login fail (redirect /login?error=true).

7) CustomSuccessHandler (Login ke baad)
- Successful authentication par CustomSuccessHandler.onAuthenticationSuccess run hota hai.
- Ye user ka record DB se lekar session me userId store karta hai:
  session.setAttribute("userId", userMaster.getUserId());
- Fir role ke basis par redirect karta hai:
  ROLE_ADMIN -> /admin/dashboard
  ROLE_FACULTY -> /faculty/dashboard
  ROLE_STUDENT -> /student/dashboard

8) Authorization during requests
- Jab bhi user koi protected URL hit karta hai, Spring Security user ke authorities check karta hai.
- Agar authority match kare to access milta hai, nahi to 403 Forbidden.

9) Important Implementation Notes / Best Practices
- DB me passwords hamesha encode (BCrypt) kar ke store karo.
- roleList me roles standard format me do: "ROLE_ADMIN", "ROLE_FACULTY", "ROLE_STUDENT".
- @ElementCollection fetch type EAGER hai; large role lists me performance consider karo.
- Sensitive endpoints ke liye CSRF ki zarurat aur exceptions ko check karo.

10) Quick Flow Diagram (textual)
User -> /login -> Security Filter Chain -> AuthenticationManager -> DaoAuthenticationProvider -> CustomUserDetailsService -> DB (UserMaster) -> Password check -> on success -> CustomSuccessHandler -> Redirect

11) Useful Code Snippets (tera code se):
AuthenticationProvider bean:
DaoAuthenticationProvider dao = new DaoAuthenticationProvider();
dao.setUserDetailsService(userDetailsService);

dao.setPasswordEncoder(passwordEncoder());

12) Troubleshooting Tips

- Agar login kaam nahi kar raha: check DB stored password (encoded), check username parameter name in form.
- Agar role based redirect galat ja raha: ensure roles in DB include "ROLE_" prefix or map accordingly.
- Static resources load nahi ho rahe: ensure paths like /css/** are permitted in config.

End of document. Agar tu chahe to main is PDF me aur diagrams add kar ke updated version de sakta hu.