Spring boot Security - Part2

Friday, 7 March 2025 3:48 PM

User Creation:

Before, we proceed with User Authentication and Authorization methods, we first need to see because that's the first step.

Authentication and Authorization of User will happen only after User is created.

Lets see, what will happen when we add below security dependency, as seen in previous ***Arch*** the server:

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifactId>
</dependency>
<dependency>
```

Provides cor

Logs when server is started:

```
2025-03-08T15:09:16.356+05:30  INFO 44103 --- [          main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA
2025-03-08T15:09:16.466+05:30  WARN 44103 --- [          main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-
2025-03-08T15:09:16.500+05:30  WARN 44103 --- [          main] .s.s.UserDetailsServiceAutoConfiguration :

Using generated security password: b5341001-1e0d-4bad-9440-0f8b1c51cfa9

This generated password is for development use only. Your security configuration must be updated before running your applic
```

```
  main] r$InitializeUserDetailsManagerConfigurer : Global AuthenticationManager configured with UserDetailsSer
  main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 8080 (http) with context path '/'
  main] c.c.l.SpringbootApplication              : Started SpringbootApplication in 1.751 seconds (process run
exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring DispatcherServlet 'dispatcherServlet'
exec-1] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherServlet'
exec-1] o.s.web.servlet.DispatcherServlet        : Completed initialization in 1 ms
```

So, what exactly happened here?

- During server startup, user is created automatically with default username: "user"
- Random password is generated for testing.
- Each time, server is restarted, new random password will get generated.

SecurityProperties.java

```
public static class User {

/** Default user name */
private String name = "user";

/** Password for the default username */
Private String password = UUID.randomUUID().toString();

/** Granted roles for the default username */
private List<String> roles = new ArrayList<>();

private boolean passwordGenerated = true;
                .
                .
                //getters and setters
                .
}
```

@AutoConfiguration

UserDetailsServiceAu

```
@Bean
public InMemoryUserDetailsManager i
        ObjectProvider<PasswordEnco
    SecurityProperties.User user =
    List<String> roles = user.getRo
    return new InMemoryUserDetailsM
        .password(getOrDeducePasswo
        .roles(StringUtils.toString
        .build());
}
```

InMemory

```
private final Map <String, Mut
();
 public InMemoryUserDetailsMa
     for (UserDetails user :
         createUser(user);
     }
 }
```

```
@Override
public void createUser(UserDeta
    Assert.isTrue(!userExists(u
    this.users.put(user.getUser
}
```

How we can control the user creation logic?

1st: Using application.properties

(not recommended, only for development and testing)

application.properties

```
spring.security.user.name=my_username
spring.security.user.password=my_password
spring.security.user.roles=ADMIN
```

Internally
setUserN
method c
and_over

Now, during application startup, no default username and default p created.

```
45512 --- [           main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence u
45512 --- [           main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefo
45512 --- [           main] r$InitializeUserDetailsManagerConfigurer : Global AuthenticationManager configured with UserDetai
45512 --- [           main] o.s.b.w.embedded.tomcat.TomcatWebServer  : Tomcat started on port 8080 (http) with context path '
45512 --- [           main] c.c.l.SpringbootApplication              : Started SpringbootApplication in 1.701 seconds (proces
45512 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].[localhost].[/]       : Initializing Spring DispatcherServlet 'dispatcherServl
45512 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Initializing Servlet 'dispatcherServlet'
45512 --- [nio-8080-exec-1] o.s.web.servlet.DispatcherServlet        : Completed initialization in 0 ms
```

2nd: By creating custom InMemoryUserDetailsManager Bean
(not recommended, only for development and testing)

UserDetailsService

↓

UserDetailsManager

↓

InMemoryUserDetailsManager

```
@Configuration
@EnableWebSecurity
public class SecurityConfig {

    @Bean
    public UserDetailsService userDetailsService() {
        UserDetails user1 = User.withUsername("my_username_1")
                .password("{noop}my_password_1") // {noop} means no encoding or hashing
                .roles("ADMIN")
                .build();

        UserDetails user2 = User.withUsername("my_username_2")
                .password("{noop}1234")  // {noop} means no encoding or hashing
                .roles("USER")
                .build();
```

```
        return new InMemoryUserDetailsManager(user1, user2);
    }
}
```

why we are appending {noop} here?

The default, format for storing the password is : *{id}encodedpassword*

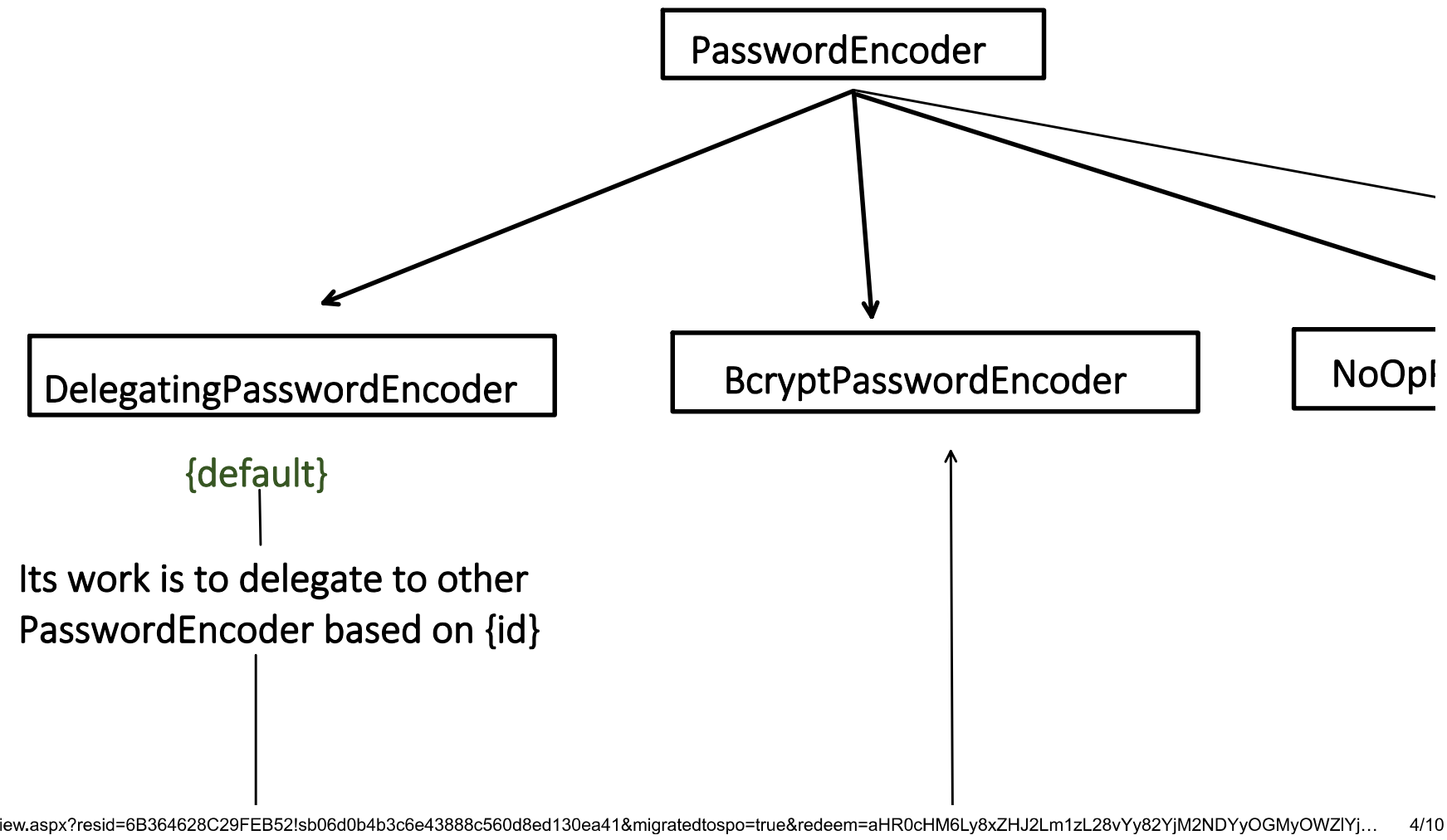
{id} can be either:

- *{noop}*
- *{bcrypt}*
- *{sha256}*
- *Etc..*

-- During User password storing step, if we want to store user password without any encoc
{noop}plain_password"

-- Now, during authentication process:

- 1st, it will fetch the user password from inMemory.
- 2nd, it goes for comparing logic, inMemory password and password provided for auth
- 3rd, it will take out the {noop} or {bcrypt} etc. from inMemory password.
- 4th, Then if its {noop}, it will directly compare the remaining inMemory password and
- 5th, if say its {bcrypt}, it first do hashing of provided password using BCryptPasswordE inMemory Password.

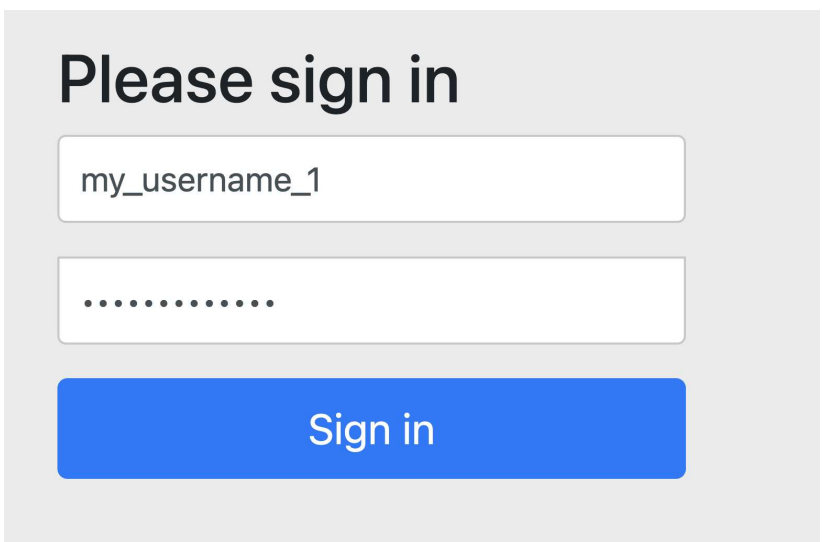


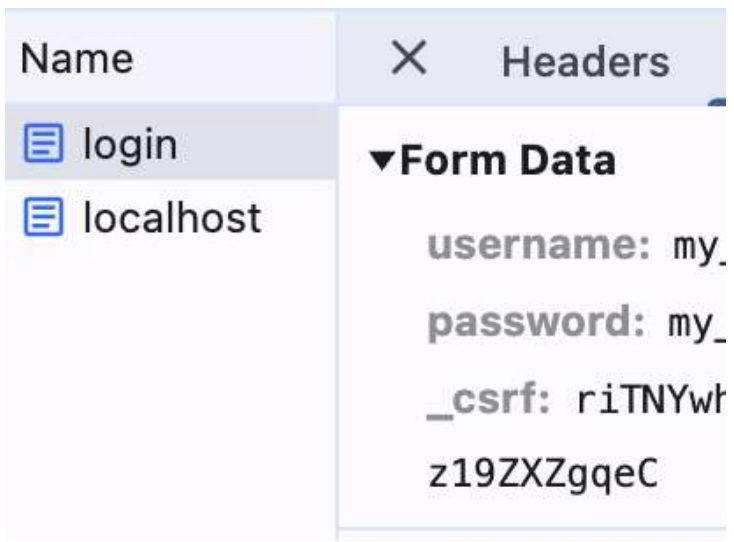
Lets say, if we want to store the hashed password (hashed using ***bcry***

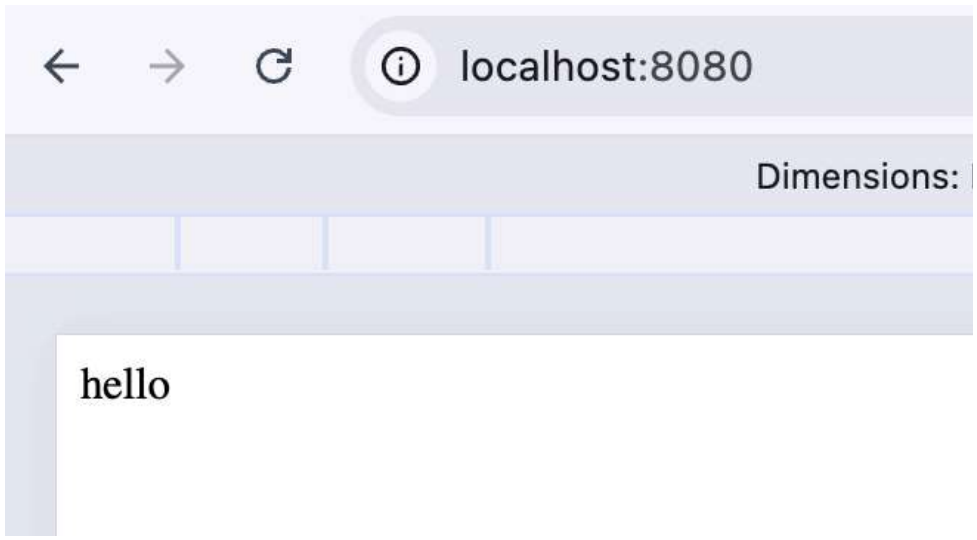
```java
@Configuration
@EnableWebSecurity
public class SecurityConfig {

    @Bean
    public UserDetailsService userDetailsService() {
        UserDetails user1 = User.withUsername("my_username_1")
                .password("{bcrypt}" + new BCryptPasswordEncoder().encode( ra
                .roles("ADMIN")
                .build();

        return new InMemoryUserDetailsManager(user1);
    }
}
```

InMemory, password is stored as : {bcrypt}hased_password
and during authentication, I am providing "my_password_1"

But still I am able to successfully authenticate because of **DelegationPasswordEncoder,** it firs
password {id} i.e. {bcrypt}, so it passes the incoming password to BcryptPasswordEncoder, ar
matching.







If, we don't want to store {bcrypt} or any other hashing algo {id} in front of password, t
PasswordEncoder to use.

Now, since we are always using 1 encoding/hashing algorithm, and control will not goe
"DelegationPasswordEncoder", and it will directly goes to specific Password Encoder, s
of password.

```java
@Configuration
@EnableWebSecurity
public class SecurityConfig {

    @Bean
    public PasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder();
    }

    @Bean
    public UserDetailsService userDetailsService() {
        UserDetails user1 = User.withUsername("my_username_1")
                .password(new BCryptPasswordEncoder().encode( rawPassword: "my
                .roles("ADMIN")
                .build();

        return new InMemoryUserDetailsManager(user1);
    }
}
```

3rd: Storing UserName and Password (after hashed) in I

(recommended for production)

Implements UserDetails because,
During Authentication (form, basic, jwt etc..), secu
framework tries to fetch the user and return the ob

UserDetails only, if we don't implement it, then we do the mapping (from UserAuthEntity to UserDetai

UserAuthEntity.java

```java
@Entity
@Table(name = "user_auth")
public class UserAuthEntity implements UserDetails {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(unique = true, nullable = false)
    private String username;

    @Column(nullable = false)
    private String password;

    private String role;

    @Override
    public Collection<? extends GrantedAuthority> getAuthorities() {
        return List.of(new SimpleGrantedAuthority(role));
    }

    @Override
    public boolean isAccountNonExpired() { return true; }

    @Override
    public boolean isAccountNonLocked() { return true; }

    @Override
    public boolean isCredentialsNonExpired() { return true; }

    @Override
    public boolean isEnabled() { return true; }

    //getters and setters
    @Override
    public String getPassword() {
        return password;
    }

    @Override
    public String getUsername() {
        return username;
    }

    public void setPassword(String password) {
        this.password = password;
    }

    public String getRole() {
        return role;
    }

    public void setRole(String role) {
        this.role = role;
    }
}
```

```
}
```

Now, by-default in spring boot security, all the endpoints are AUTHENTICATED, username/password or JWT etc.. To access any API, so how we will access "***/au*** create user.

Yes, we have to relax the authentication for this API and its industry standard.

SecurityConfig.java

```java
@Configuration
@EnableWebSecurity
public class SecurityConfig {

    @Bean
    public PasswordEncoder passwordEncoder() {
        return new BCryptPasswordEncoder();
    }

    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity http
        http
                .authorizeHttpRequests(auth -> auth
                        .requestMatchers( ...patterns: "/auth/register")
                        .anyRequest().authenticated()
                )
                .csrf(csrf -> csrf.disable())
                .httpBasic(Customizer.withDefaults());
        return http.build();
    }
}
```

POST localhost:8080/auth/register

Params Authorization Headers (9) Body Scripts Settings

none form-data x-www-form-urlencoded raw binary GraphQ

```
1  {
2      "username" : "my_username",
3      "password" : "111abc",
4      "role" : "ROLE_USER"
```

```
5   }
```

Body Cookies (2) Headers (11) Test Results

Pretty Raw Preview Visualize Text

```
1   User registered successfully!
```

Run | Run Selected | Auto complete | Clear | SQL statement:

```
SELECT * FROM USER_AUTH
```

SELECT * FROM USER_AUTH;

ID	PASSWORD	ROLE
1	$2a$10$euzihUhyp4exMejDkyDb0eK2q49sqTcG8EShOnT2GmaL7lxvIeOfO	ROLE_

(1 row, 3 ms)

Edit