Table of Contents

[Terms 2](#_Toc100412288)

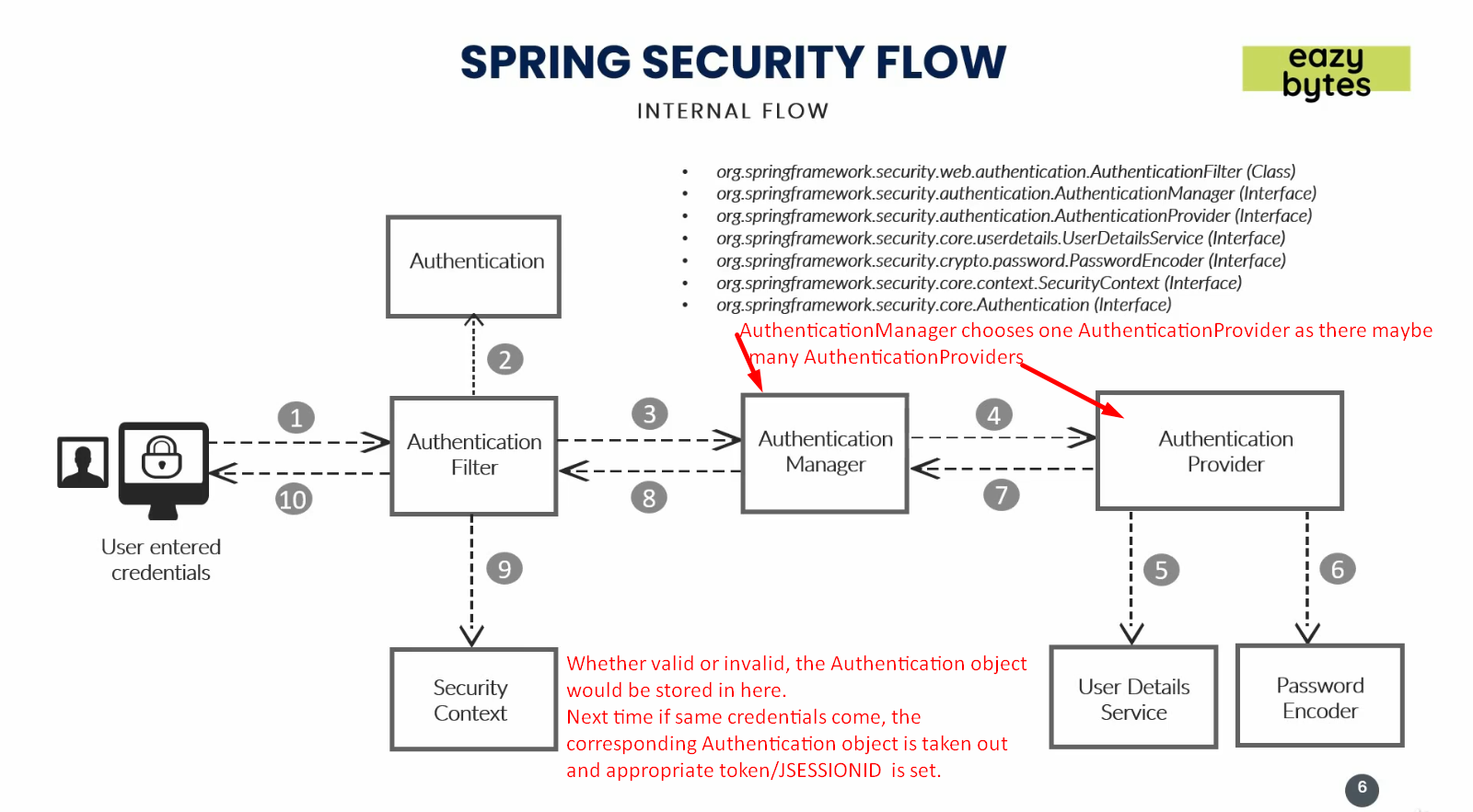
[Java classes, Interfaces 2](#_Toc100412289)

[Annotations 6](#_Toc100412290)

[application.properties 6](#_Toc100412291)

[Questions 6](#_Toc100412292)

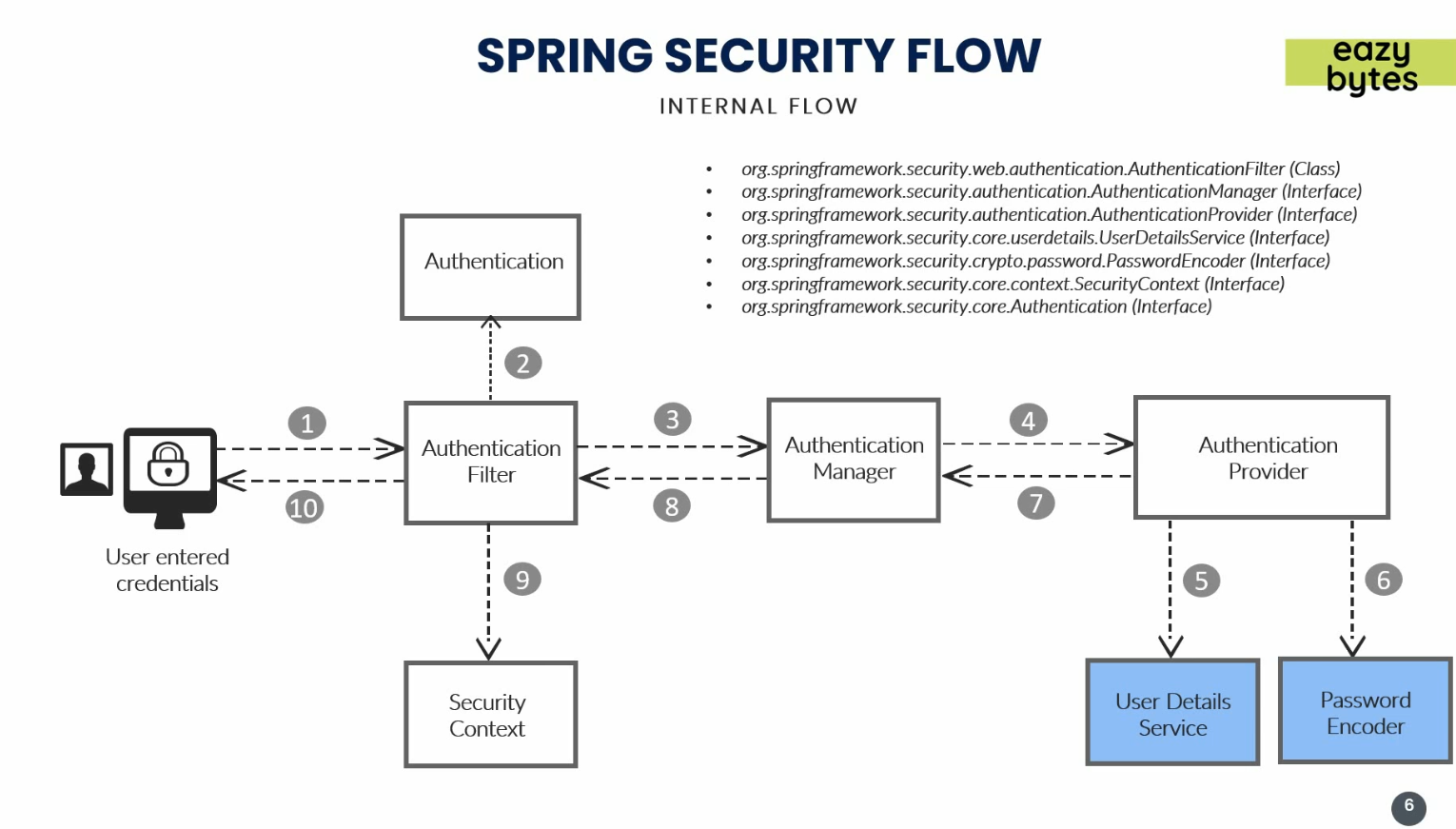
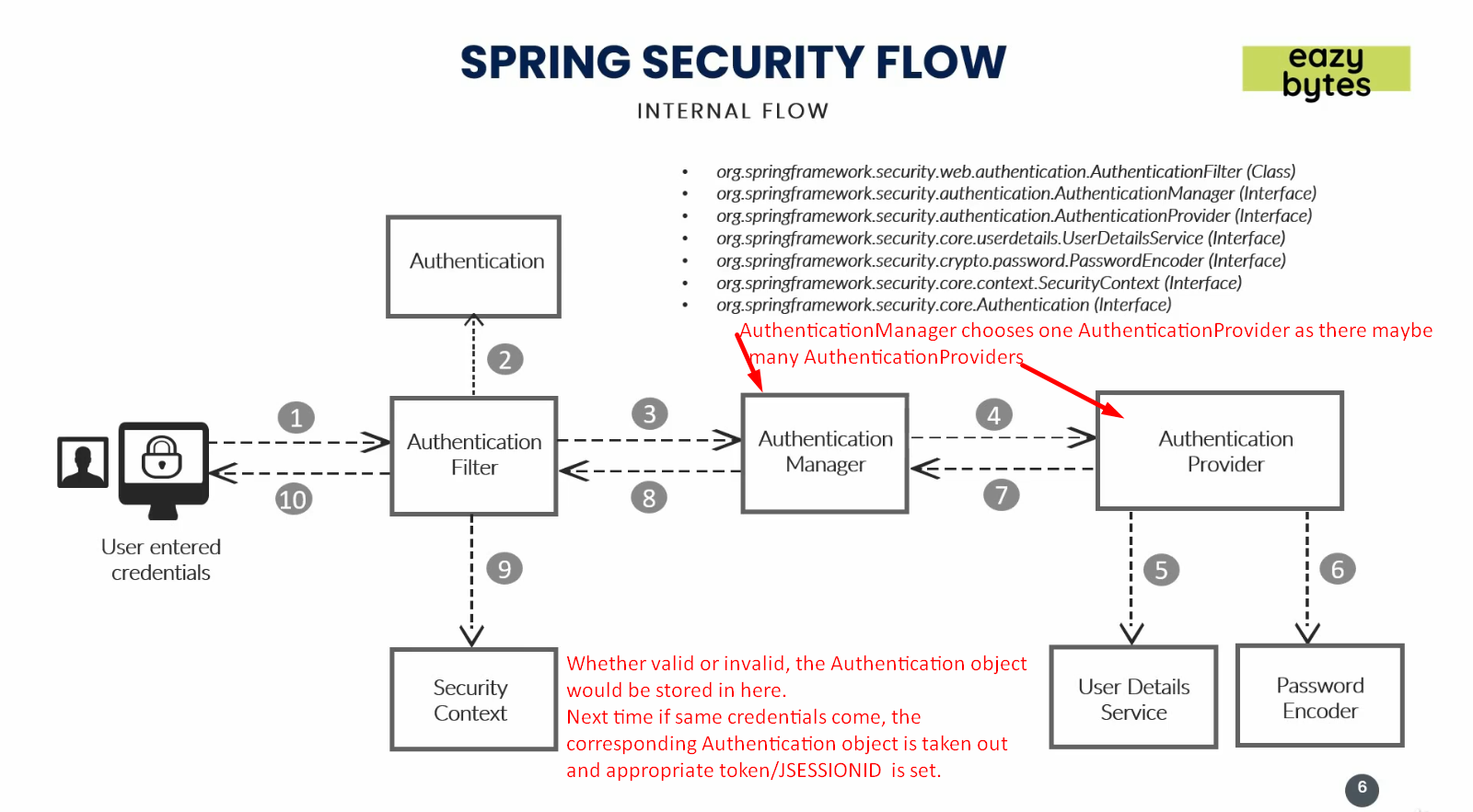
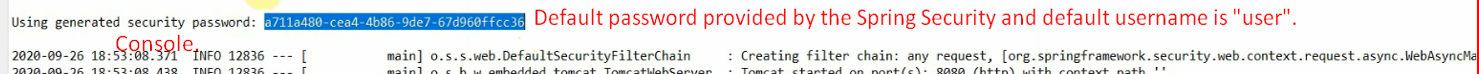
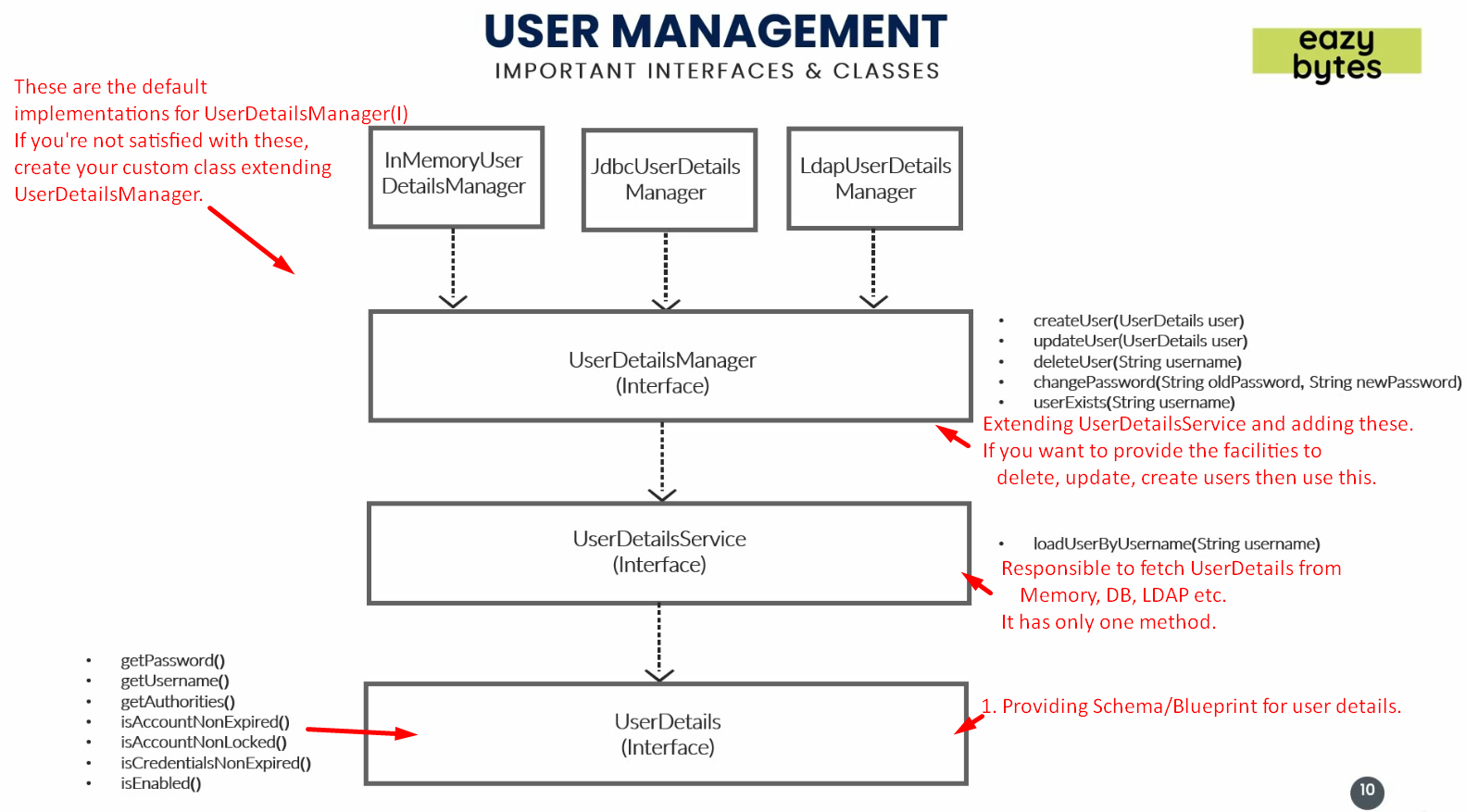
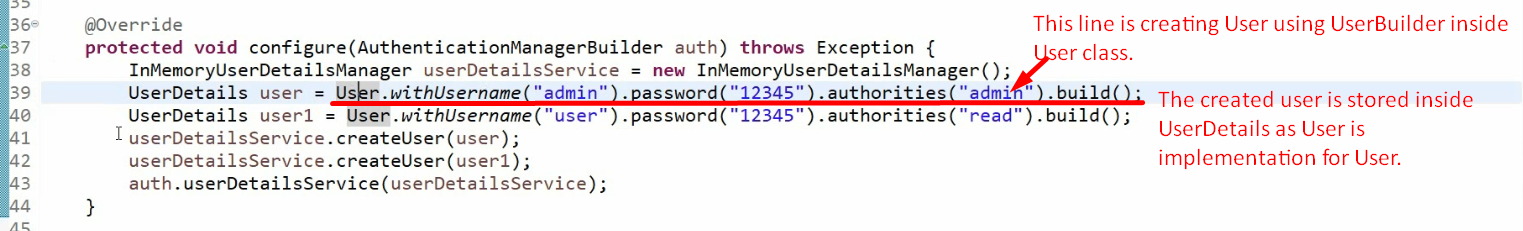
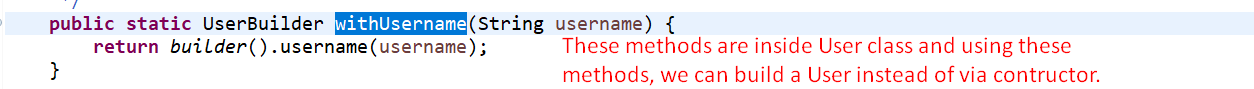
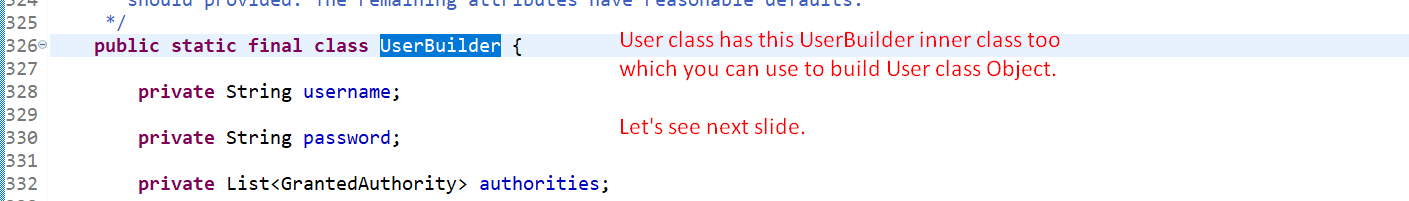
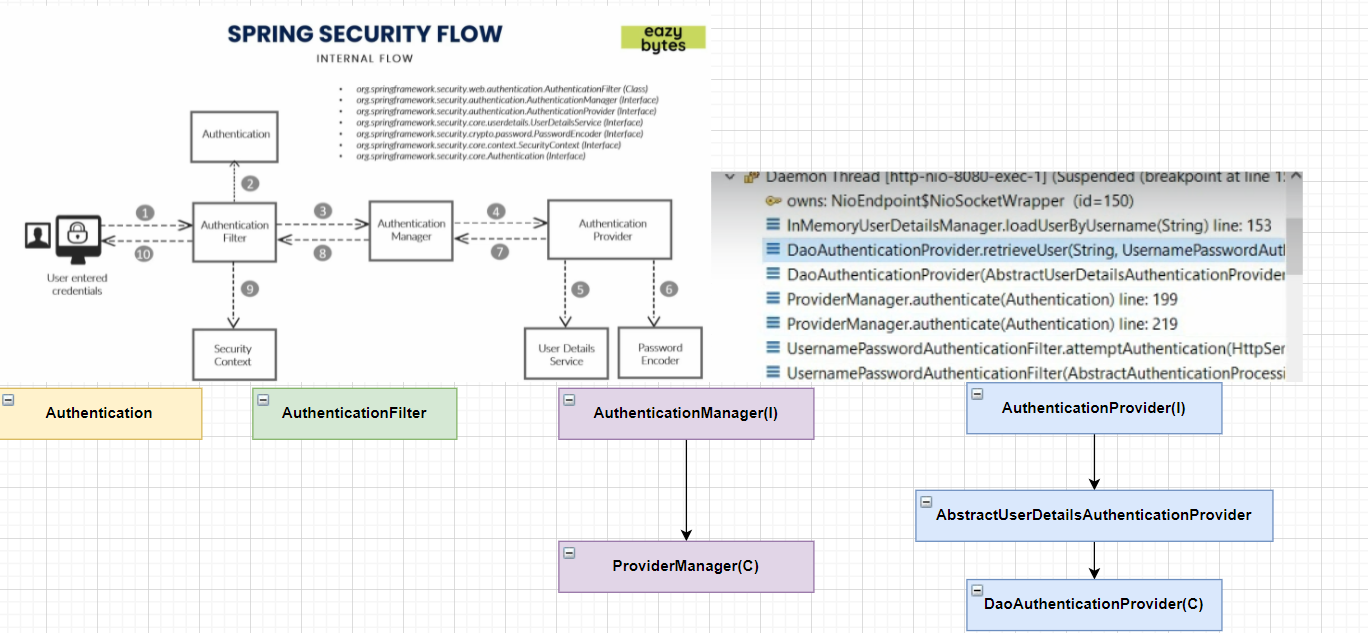
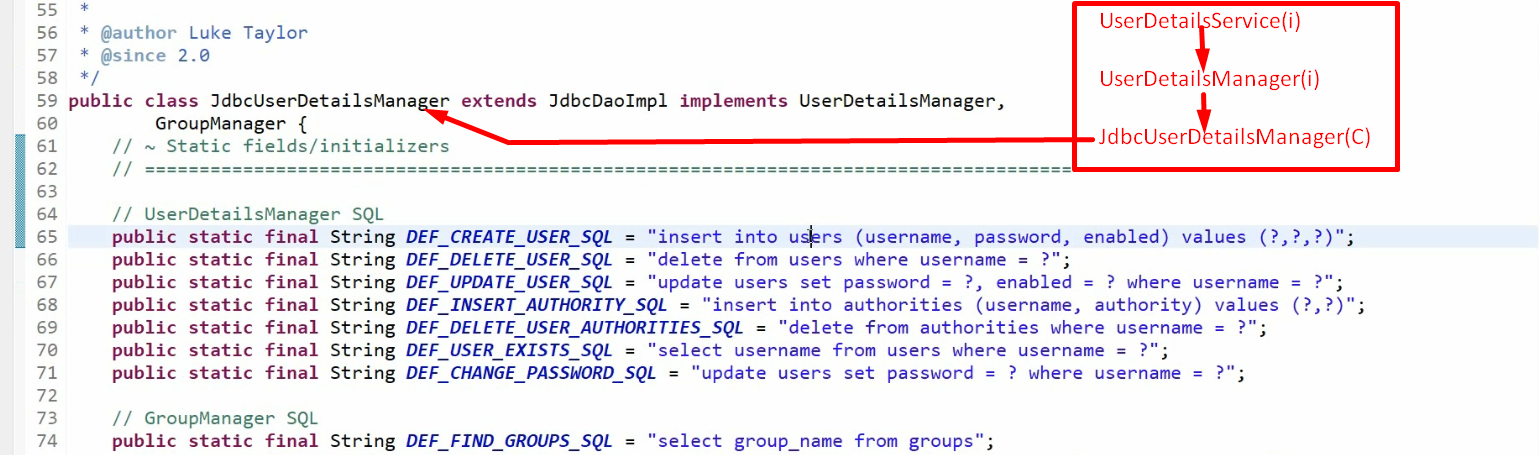
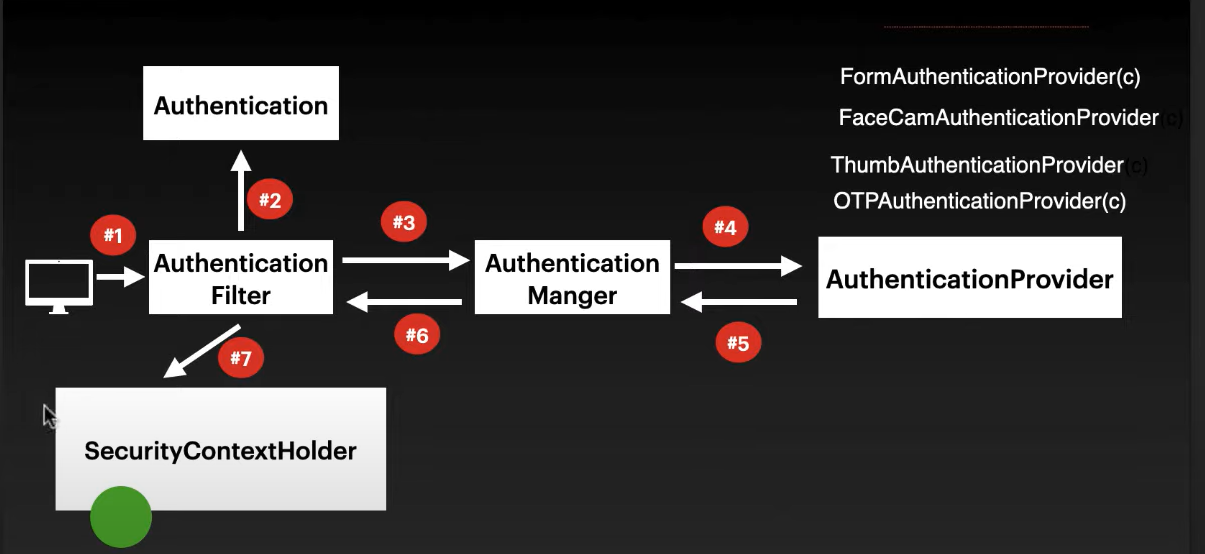
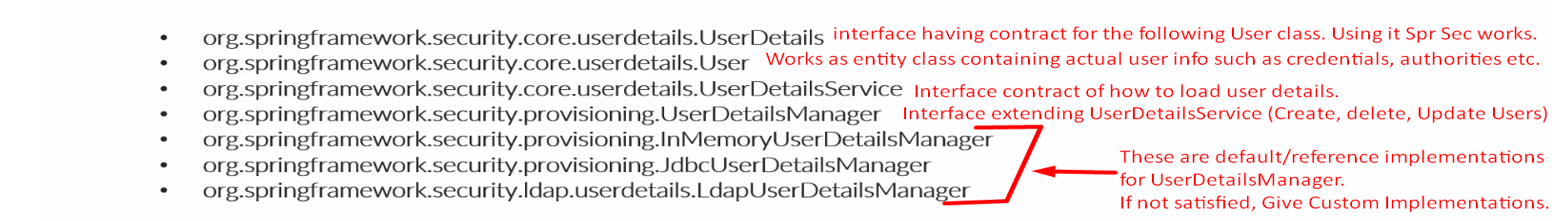
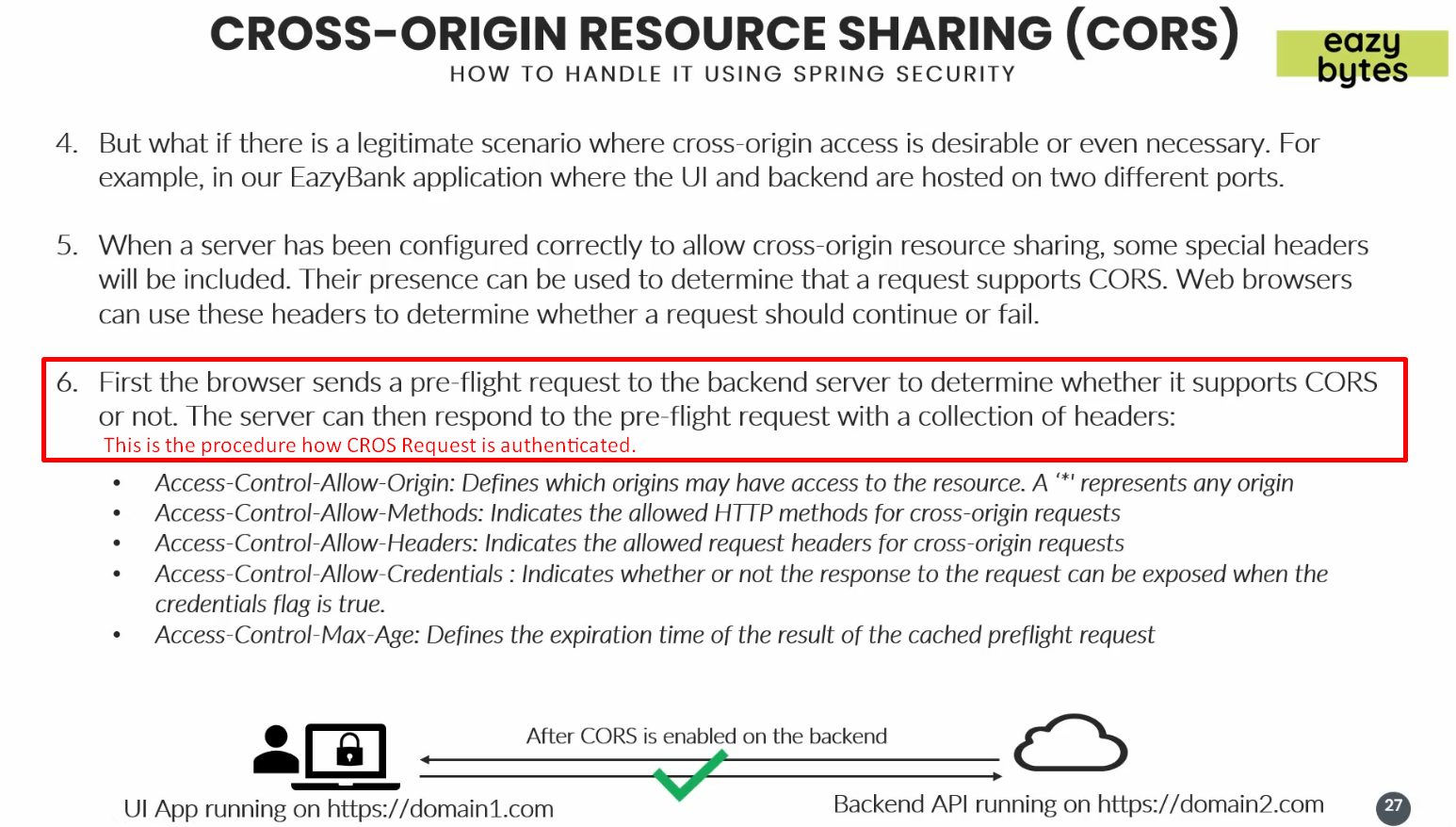
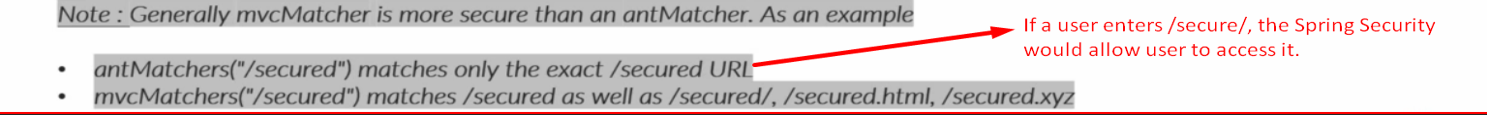
[Skipped 6](#_Toc100412293)



# Terms

1. RESTful End-Point
2. Spring Security Starter Dependency
3. JSESSIONID, Cookies, Header, Basic Auth (Username, Password).
4. RESTful API Consumers, Browsers, Postman.
5. LDAP Server

# Java classes, Interfaces

1. WebSecurityConfigurerAdapter.java
2. 
3.   
   Authentication.java, AuthenticationFilter.java, AuthenticationManager.java, AuthenticationProvider.java  
   UserDetailsService.java, PasswordEncoder.java
4. 
5. Authentication, Authorization, Roles, Authorities.
6. User implements UserDetails  
   UserDetailsService(I).loadUserByUsername(String username):UserDetails  
   UserDetailsManager(I) extends UserDetailsService  
     
   NOTE: UserDetailsService doesn’t extend UserDetails but User implements UserDetails.  
   NOTE: If LDAP Server, then LdapUserDetailsManager  
   
7. AuthenticationProvider(I)  
   DaoAuthenticationProvider implements AuthenticationProvider.
8. ProviderManager implements AuthenticationManage  
   
9. 
10.  
11. 
12. AuthenticationFilter 🡺 AuthenticationManager(ProviderManager)🡺AuthenticationProvider
13. For Face, Filter etc, create a specific class implementing Authentication as we have UsernamepasswordAuthentication for username and password already.
14. DaoAuthenticationProvider for username and password is configured automatically.
15. Classes, interfaces 🡺 Authentication, Principle(Core Java),
16. CORS 🡺 Cross-Origin Resource Sharing  
    
    1. Cross Origin access
    2. pre-flight request by browser
    3. Access-Control-Allow-<name>
17. Authorization (AUTHZ=Single Individual Permission/Privilege) = Privileges = Roles (Group of Authorities)
    1. UserDetails (Contract for User in Spring Security).
    2. UserDetails.getAuthorities(): Collection<? extends GrantedAuthorities>
    3. GrantedAuthority.getAuthority(): String
    4. Authorization Configuration.
    5. SimpleGrantedAuthority implements GrantedAuthority
    6. UsernamePasswordAuthenticationToken
    7. BadCredentialsException(msg);
    8. Authorities🡺 READ, WRITE etc.  
       ROLE🡺 Admin, Supervisor etc.
18. Matchers  
    
    1. MVC Matcher🡺 HandlerMappingIntrospector.
       1. mvcMatchers (enum HttpMethod, patterns)
       2. mvcMatchers(patterns)
    2. Ant Matcher 🡺 Ant-Style Path Patterns from Apache Ant.
       1. Exact Match.
       2. Same methods as for MVC.
    3. RegEx Matcher 🡺 regexMatcher (…) Same methods as for MVC.
19. Filters
    1. Filter Interface. doFilter ()
    2. ServletRequest, ServletResponse, FilterChain
    3. Methods to add filters:  
       addFilterBefore (filter, class)  
       addFilterAfter (filter, class)

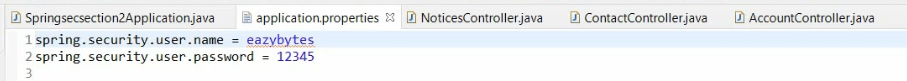
addFilterAt (filter, class)

* 1. GenericFilterBean implements Filter
  2. OncePerRequest extends GenericFilterBean

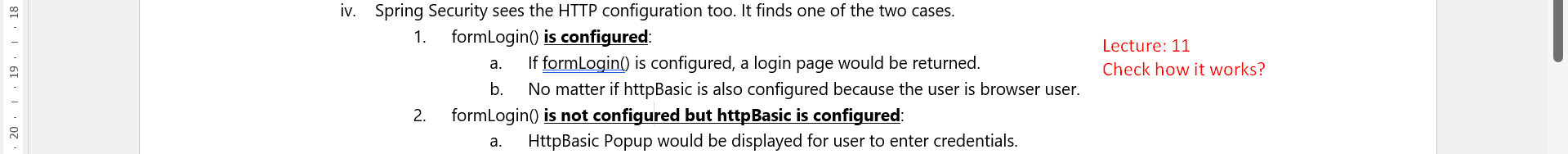
# Annotations

1. @RestController
2. @GetMapping
3. @Configuration
4. @SpringBootApplication  
   @ComponentScan(“packagename”)

# application.properties

1. 

# Questions

1. formLogin() and httpBasic()
2. 

# Skipped

1. Spring Security From Zero to Master
   1. 16, 17