☑ FULL LOGIN FLOW — CLASS → METHOD → ACTION SUMMARY

1️⃣ Security Layer (Spring Security Filter Chain)

---

1. FilterChainProxy

Method: *doFilterInternal()*
क्या करता है:

Request को पूरी security filter chain से गुजरवाता है

Logs: *"Securing POST /account/login"*

---

2. JWTAuthFilter *(Your Custom Filter)*

Method: *doFilterInternal(HttpServletRequest, HttpServletResponse, FilterChain)*
क्या होता है:

Authorization header चेक करता है

अगर header null है → token नहीं है → आगे chain को allow करता है

अगर valid token होता → username निकालकर authentication set कर देता

Logs:

"Inside doFilterInternal"

Token found/not found

---

3. AnonymousAuthenticationFilter

Method: *doFilter()*
क्या करता है:

अगर previous filters ने authentication set नहीं किया

तब anonymous user को SecurityContext में डालता है

---

4. SessionManagementFilter

Method: *doFilterInternal()*
क्या करता है:

JSESSIONID चेक करता है

Invalid session ID log करता है

2️⃣ DispatcherServlet Layer (Spring MVC Request Processing)

---

5. DispatcherServlet

Method: *doDispatch()*
क्या करता है:

Request को matching controller तक पहुंचाता है

---

6. RequestMappingHandlerMapping

Method: *getHandler()*
क्या करता है:

URL /account/login को map करता है

Controller method: AccountController.login() ढूँढता है

Log: *Mapped to AccountController#login*

---

7. RequestResponseBodyMethodProcessor

Method: *readWithMessageConverters()*
क्या होता है:

JSON request body को convert करता है into LoginRequestDTO

Log: *Read "application/json" to LoginRequestDTO*

3️⃣ Controller Layer

---

8. AccountController

Method: *login(LoginRequestDTO loginRequestDTO)*
क्या करता है:

Logs: "Inside login Controller"

Calls service method → accountService.login(loginRequestDTO)

Service से वापस मिले response को GlobalResponse में wrap करके return करता है

4️⃣ Service Layer

---

9. AccountService

Method: *login(LoginRequestDTO)*
Main actions:

Logs request

Authentication Manager को call करता है

username = email

password raw (123)

Spring Security authentication होता है

अगर successful:

User entity को fetch करता है

Token generate करता है via JWTUtil

LoginResponseDTO बनाकर return करता है

Log sequence:

"Inside login method"

"Calling the authenticate"

"Authenticated!!"

"User is ready so generatingToken"

"Token is generated : eyJ..."

5️⃣ Security Authentication Layer

---

10. CustomUserDetailsService

Method: *loadUserByUsername(String email)*
क्या करता है:

DB से user ढूँढता है (via Repository)

User entity → Spring UserDetails में convert करता है

Log:

"Inside loadUserByUsername with email ..."

"Got the user User(...)"

---

11. UserRepo

Method: *findByUserEmail(String email)*
क्या करता है:

Hibernate query चलाता है:
SELECT * FROM users WHERE user_email = ?

DB से encrypted password सहित user entity return करता है

---

12. DaoAuthenticationProvider

Method: *authenticate(Authentication)*
क्या करता है:

raw password और stored BCrypt hash compare करता है

Success होने पर authenticated token return करता है

Log: *"Authenticated user"*

6️⃣ JWT Layer

---

13. JWTUtil

Method: *generateToken(UserDTO or User)*
क्या करता है:

subject = userEmail

iat = issued at

exp = expiration time

secret key से token sign करता है

Log: *"Inside generateToken with userDetails..."*

Also uses methods:

extractUsername(token)

Token decode करके subject निकालता है

isTokenValid(token, userDetails)

Username match

Expiration check

7️⃣ Response Generation Layer

---

14. HttpEntityMethodProcessor

Method: *writeWithMessageConverters()*
क्या करता है:

GlobalResponse<LoginResponseDTO> को JSON में convert करता है

Response में लिखता है

---

15. DispatcherServlet

Method: *processDispatchResult()*
क्या करता है:

Logs: *"Completed 200 OK"*

Final response client को भेज देता है

---

16. OpenEntityManagerInViewInterceptor

Method: *afterCompletion()*
क्या करता है:

JPA EntityManager close करता है

DB resources free करता है

✅ **FINAL SUPER-SHORT SUMMARY TABLE (Interview वाला)**

Layer	Class	Method	क्या होता है
Security	FilterChainProxy	doFilterInternal	Filter chain शुरू
Security	JWTAuthFilter	doFilterInternal	Token चेक
Security	AnonymousAuthenticationFilter	doFilter	Anonymous auth set
MVC	DispatcherServlet	doDispatch	Controller route
MVC	HandlerMapping	getHandler	Login method mapped
MVC	BodyProcessor	readWithMessageConverters	JSON → DTO
Controller	AccountController	login()	Service call
Service	AccountService	login()	Authenticate, generate token
Security	CustomUserDetailsService	loadUserByUsername	User load from DB
DB	UserRepo	findByUserEmail	Query execution
Security	DaoAuthenticationProvider	authenticate	Password verify
JWT	JWTUtil	generateToken	JWT sign
Response	HttpEntityMethodProcessor	writeWithMessageConverters	DTO → JSON
MVC	DispatcherServlet	return 200 OK	Response sent

🚀 **FINAL REQUEST FLOW (LOGIN Example) — CLASS → METHOD ORDER**

(यह sequence 100% उसी order में चलता है जैसा runtime पर होता है)

1️⃣ Incoming HTTP Request

👉 POST /account/login

2️⃣ SPRING SECURITY FILTER CHAIN

STEP 1 → FilterChainProxy

Class: org.springframework.security.web.FilterChainProxy

Method: doFilterInternal()

✓ पूरे filter chain को activate करता है

STEP 2 → JWTAuthFilter (Your filter)

Class: com.jwt.JWTApp.Configuration.JWTAuthFilter

Method: doFilterInternal(request, response, chain)

✓ Authorization header check
✓ Token validate (if present)
✓ SecurityContext में authenticated user set करता है (only if token found)
✓ Login request में token नहीं होता → pass-through

STEP 3 → AnonymousAuthenticationFilter

Class: org.springframework.security.web.authentication.AnonymousAuthenticationFilter

Method: doFilter()

✓ अगर JWT ने authentication नहीं set किया → anonymous user सेट करता है

STEP 4 → SessionManagementFilter

Class: org.springframework.security.web.session.SessionManagementFilter

Method: doFilterInternal()

✓ Invalid session ID log करता है

3️⃣ SPRING MVC REQUEST HANDLING

STEP 5 → DispatcherServlet

Class: org.springframework.web.servlet.DispatcherServlet

Method: doDispatch()

✓ Request को controller तक पहुंचाने की तैयारी करता है

STEP 6 → RequestMappingHandlerMapping

Class: org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerMapping

Method: getHandler()

✓ /account/login को map करता है to controller method:

AccountController.login(LoginRequestDTO)

STEP 7 → RequestResponseBodyMethodProcessor

Class:
org.springframework.web.servlet.mvc.method.annotation.RequestResponseBodyMethodProcessor

Method: readWithMessageConverters()

✓ JSON request body → LoginRequestDTO में convert करता है

4️⃣ CONTROLLER LAYER

STEP 8 → AccountController

Class: com.jwt.JWTApp.Controller.AccountController

Method: login(LoginRequestDTO loginRequestDTO)

✓ Logs: “Inside login controller”
✓ Calls service:

accountService.login(loginRequestDTO)

5️⃣ SERVICE LAYER

STEP 9 → AccountService

Class: com.jwt.JWTApp.Service.AccountService

Method: login(LoginRequestDTO dto)

✓ Logs: “Inside login method”
✓ STEP A → authentication start:

```
authenticationManager.authenticate(
    new UsernamePasswordAuthenticationToken(dto.email, dto.password)
)
```

STEP 10 → AuthenticationManager → ProviderManager

Class:
org.springframework.security.authentication.ProviderManager

Method: authenticate(Authentication)

✓ Authentication provider को delegate करता है

6️⃣ **USER LOADING (Security)**

STEP 11 → DaoAuthenticationProvider

Class:
org.springframework.security.authentication.dao.DaoAuthenticationProvider

Method: authenticate()

✓ Username से user को load करने के लिए calls:

customUserDetailsService.loadUserByUsername(email)

STEP 12 → CustomUserDetailsService

Class: com.jwt.JWTApp.Security.CustomUserDetailsService

Method: loadUserByUsername(String email)

✓ Calls repository:

userRepo.findByUserEmail(email)

✓ User → Spring Security UserDetails में convert करता है
✓ Return करता है user

7️⃣ DATABASE LAYER (Spring Data JPA + Hibernate)

STEP 13 → UserRepo

Class: com.jwt.JWTApp.Repository.UserRepo

Method: findByUserEmail(String email)

✓ Generates query:
SELECT u FROM User u WHERE u.userEmail = :email

✓ Hibernate execute करता है
✓ User entity return होती है

8️⃣ PASSWORD VERIFICATION

STEP 14 → DaoAuthenticationProvider

Method: additionalAuthenticationChecks()

✓ Compares raw password with BCrypt hashed password
✓ Authenticated होने पर authenticated token return

STEP 15 → Back to AccountService

✓ Log: “Authenticated!!”

✓ फिर user DB से दोबारा load होता है (optional but you did)
✓ Calls JWTUtil.generateToken()

9️⃣ JWT TOKEN GENERATION

STEP 16 → JWTUtil

Class: com.jwt.JWTApp.Globals.JWTUtil

Method: generateToken(User user)

✓ Sets subject = email
✓ Sets issuedAt & expiration
✓ Signs token using secret key
✓ Returns JWT string

🔟 SERVICE RETURNS RESPONSE DTO

STEP 17 → AccountService

Method: login()

✓ Creates LoginResponseDTO(token, userDTO)
✓ Returns to controller

1️⃣1️⃣ CONTROLLER SENDS RESPONSE

STEP 18 → AccountController

Method: login()

✓ Wraps into GlobalResponse
✓ Returns to MVC layer

1️⃣2️⃣ RESPONSE BODY CONVERSION (DTO → JSON)

STEP 19 → HttpEntityMethodProcessor

Class: HttpEntityMethodProcessor

Method: writeWithMessageConverters()

✓ Converts GlobalResponse → JSON
✓ Writes to HTTP response

1️⃣3️⃣ REQUEST COMPLETED

STEP 20 → DispatcherServlet

Method: processDispatchResult()

✓ Logs: “Completed 200 OK”
✓ Response भेज देता है client को

🎉 FINAL CLEAN FLOW FOR YOUR NOTES

(Use this in your notebook)

1. FilterChainProxy → doFilterInternal
2. JWTAuthFilter → doFilterInternal
3. AnonymousAuthenticationFilter → doFilter
4. SessionManagementFilter → doFilterInternal
5. DispatcherServlet → doDispatch
6. HandlerMapping → getHandler
7. BodyProcessor → readWithMessageConverters
8. AccountController → login
9. AccountService → login
10. AuthenticationManager → authenticate
11. DaoAuthenticationProvider → authenticate
12. CustomUserDetailsService → loadUserByUsername
13. UserRepo → findByUserEmail

14. DaoAuthenticationProvider → password check
15. AccountService → generate token request
16. JWTUtil → generateToken
17. AccountService → return LoginResponseDTO
18. AccountController → return GlobalResponse
19. HttpEntityMethodProcessor → writeWithMessageConverters
20. DispatcherServlet → Completed 200 OK