JWT (JSON Web Token) – Complete Notes for Spring Boot

Purpose of these notes:

• Quick revision before interviews
• Clear conceptual understanding
• Easy recall while coding JWT from scratch
• Based on your implemented code (Spring Boot + Spring Security + JWT)

---

1. What is JWT?

JWT (JSON Web Token) is a **stateless authentication mechanism** used to securely transmit information between client and server as a **JSON object**.

• Used mainly for **authentication & authorization**
• Token is generated by server and sent to client
• Client sends token in every request
• Server validates token → no session stored

JWT Full Form

JSON Web Token

---

2. Why JWT?

Problems with **Session-based Authentication**:

• Server stores session data
• Not scalable
• Difficult in microservices
• Requires sticky sessions

Advantages of **JWT**:

• Stateless
• Scalable
• Suitable for REST APIs
• Works well with microservices
• Faster (no DB lookup per request)

---

3. JWT Structure

JWT consists of **3 parts**, separated by `.`

```
HEADER.PAYLOAD.SIGNATURE
```

3.1 Header

Contains metadata about token

```
{
  "alg": "HS256",
  "typ": "JWT"
}
```

3.2 Payload

Contains **claims (data)**

```
{
  "sub": "ganesh",
  "iat": 1700000000,
  "exp": 1700003600
}
```

Types of Claims:

- Registered (sub, exp, iat)
- Public
- Private

3.3 Signature

Used to verify token integrity

```
HMACSHA256(
  base64UrlEncode(header) + "." + base64UrlEncode(payload),
  secretKey
)
```

> If payload is modified → signature becomes invalid

---

4. JWT Authentication Flow

1. User sends **username + password**
2. Server authenticates credentials
3. Server generates **JWT**
4. Token sent to client
5. Client stores token (Header / LocalStorage)
6. Client sends token in every request
7. Server validates token
8. Request allowed or rejected

---

5. Where JWT is Stored?

- HTTP Header (Recommended)

```
Authorization: Bearer <JWT_TOKEN>
```

- Not recommended: Cookies / LocalStorage (security risks)

---

6. Stateless Authentication Concept

- Server does NOT store any user session
- All information comes from token
- Token itself proves identity

Important Interview Point: JWT is stateless → server does not remember user

---

7. JWT in Spring Boot (Your Implementation)

Key Components Used:

1. Authentication Controller
2. JWT Utility Class
3. JWT Filter
4. Security Configuration
5. UserDetailsService

---

8. JWT Utility Class (JwtUtil)

Responsibilities:

- Generate token
- Extract username
- Extract expiration
- Validate token

Token Generation Logic

- Uses secret key
- Uses HS256 algorithm
- Sets subject (username)
- Sets issued & expiration time

Token validity example: **5 hours**

---

9. JWT Filter (OncePerRequestFilter)

Why Filter?

- To intercept every request
- To validate JWT before controller execution

Responsibilities:

- Extract Authorization header
- Check Bearer token
- Extract username
- Load UserDetails
- Validate token
- Set authentication in SecurityContext

Runs **once per request**

---

10. SecurityContext & Authentication

- `SecurityContextHolder` stores authentication info
- After JWT validation → authentication is set
- Spring Security uses this context

Without setting authentication → request is unauthorized

---

11. Spring Security Configuration

Key configurations:

- CSRF disabled
- Stateless session policy
- JWT filter added before UsernamePasswordAuthenticationFilter
- Public & protected endpoints defined

Stateless Session

```
sessionManagement().sessionCreationPolicy(SessionCreationPolicy.STATELESS)
```

---

12. CSRF & JWT

Why CSRF Disabled?

- JWT is stateless
- No session
- Token sent in header

  CSRF mainly applies to cookies + sessions

---

13. Authentication Endpoint

- `/authenticate`
- Accepts username & password
- Uses AuthenticationManager
- Generates JWT on success

Failure → Exception

---

14. Authorization Flow

- JWT validated
- Roles extracted (if implemented)
- Access granted based on configuration

---

15. Token Expiration Handling

• Token has expiry time
• Expired token → rejected
• User must re-login

---

16. Common JWT Errors

• Token expired
• Invalid signature
• Missing Bearer keyword
• Wrong secret key

---

17. JWT vs Session Authentication (Interview)

Feature	Session	JWT
Storage	Server	Client
State	Stateful	Stateless
Scalability	Low	High
Microservices	Poor	Excellent

---

18. Best Practices

• Keep secret key secure
• Use HTTPS
• Short token expiry
• Refresh token mechanism
• Do not store JWT in LocalStorage (prefer HttpOnly cookies if needed)

---

19. JWT Interview One-Liners

• JWT is stateless authentication
• JWT has Header, Payload & Signature
• Signature ensures data integrity
• JWT is commonly used in REST APIs
• JWT removes server-side session

---

20. Quick Revision Summary

• JWT = Token-based authentication
• No session stored on server
• Token sent in Authorization header
• Filter validates token
• SecurityContext updated

---

21. Next Advanced Topics (Optional)

• Refresh Token
• Role-based JWT
• OAuth2 + JWT
• JWT with Microservices
• Token Blacklisting

✅**These notes are structured for fast revision & interviews** If you want: I can also convert this into **PDF**, **handwritten-style notes**, or **interview Q&A format**