 JWT stands for ‘JSON Web Token’, it is clear that the token holds the data in the form of JSON only.

JWT comes under stateless authentication.

**Stateful Authentication**

In this type of Authentication, there is a session management involved between client & server. When a client requests for a service from a server, it first logins to the server. Then server creates a session and stores this information in the form of key – value pairs. This session is a kind of memory at server side. We also call it HttpSession as Http protocol manages it. Further, in response to client requests, the server provides a session id with the response in the form of Cookie to the client. That cookie gets stored in the client browser. When same client makes request for the second time, cookie also comes with the request header. Consequently, the server checks the request header and if it finds the same SID (Session id) in the cookie it assumes that the request came from the same client. In this way session management happens.

When a client logs out from the server, the session gets destroyed accordingly. As a result, the server removes the session information (key-value) from the memory accordingly. Equally important, For every new client, the server creates a new session(memory).

**Stateless Authentication**

When a client sends a request to the server for a service, it first logins to the server. Consequently, the server generates a token (data in encoded format) and sends to the client with the response. While making second request, Client sends the same token along with the request to the server. Now, the server reads token from the request and validates the token. In fact, from the first request server checks the valid login(credentials) of the client. If it is a valid login, then only, server creates a token.

Furthermore, on the second request it validates the token. If the token is valid it sends the requested response, otherwise asks the client to login again. However, every token will have a valid time period, such as 30 minutes, 1 hour etc. Based on business requirements, token validity period can be configured.

In case of Token, there is no concept of logout. Instead, the client can make a request & get the response until the token expires.

**What is a Token, What is JWT authentication all about and What is the benefit of using it?**

Token is a data in an encoded format. It can be generated using a secret key(a kind of password). JWT is an abbreviation to ‘JSON Web Token’, which is a standard mechanism to generate tokens. It defines a compact and self-contained way of transmitting information securely between parties(multiple servers) as a JSON object. JWT has three parts : Header, Payload & Signature. Each part is separated by comma. Its an open source API. JWT concept exists not only in Java, but also in other languages.

Header : contains JWT Specific Information

Payload : contains Claims (Client ID, Client Name, Issuer Name, Audience Name, Date of issue, Expiry date etc…)

Signature: Base64 encoded form of Header & Payload. Additionally, signed with a secret key

**Example of an encoded** **JWT :**

**eyJhbGciOiJIUzUxMiJ9**

**.eyJqdGkiOiIzNDMyIiwic3ViIjoiZHMyNTI1IiwiaXNzIjoiQUJDX0x0ZCIsImF1ZCI6IlhZWl9MdGQiLCJpYXQiOjE2MDc0NDI1NzQsImV4cCI6MTYwNzQ0NjE3NH0**

**.3fIcXIvL9Uz0WtZgaXC95Wj8Hn7ONWKkaaspRwCT6v5Q8QSxZx7hiDQY3klYUMkfe5t2ioasYzEulM\_OGc\_GEw**

**How does JWT authentication work?**

When a client requests for a service from a server, it first logins to the server. Consequently, the server generates a token(data in encoded format) and send to the client with the response. While making a second request Client sends token along with the request to the server. Now server reads token from the request and validate the token. While validating the token which client sends with the request, server requires that secret key again to decode it. Further to validate the token server always requires the secret key. Even after the successful login, server generates token using secret key only for the first time. In summary, server requires the secret key while generating the token & even at the time of validating it too.

https://www.geeksforgeeks.org/spring-boot-3-0-jwt-authentication-with-spring-security-using-mysql-database/