EP 11 - Difference between JWT, OAuth, and SAML

JAVINPAUL AND SOMA

JUL 27, 2023 • PAID

Hello friends, one of the most common questions in Java developer interviews nowadays is **the difference between JWT, OAuth2.0, and SAML?** and **when to use them**. If you are preparing for Java developer interviews, asked this question, and looking for an answer, you have come to the right place.

While `JWT`, `OAuth`, and `SAML` These are well-known standards used for authentication and authorization purposes in web applications. There are many differences between them.

For example, JWT is **JSON Web Token,** a *standard for securely transmitting information between parties as a JSON object.* It is used to authenticate and authorize users and is commonly used in modern web applications. JWTs are digitally signed so that they can be verified and trusted.

On the other hand, **OAuth** (**Open Authorization**) is an *open standard for authorization that allows third-party applications to access user data without requiring users to share their login credentials.* It is commonly used in applications that need to access data from external services, such as social media platforms or APIs.

Similarly, **SAML** (**Security Assertion Markup Language**) is another standard for *exchanging authentication and authorization data between parties, specifically between an identity provider (IdP) and a service provider (SP).* It is commonly used in enterprise applications to provide single sign-on (SSO) functionality.

One of the most famous examples of SAML is SingPass authentication, which is used Singapore Government to access government websites like Vaccination certificates, CPF, IRAS, etc.

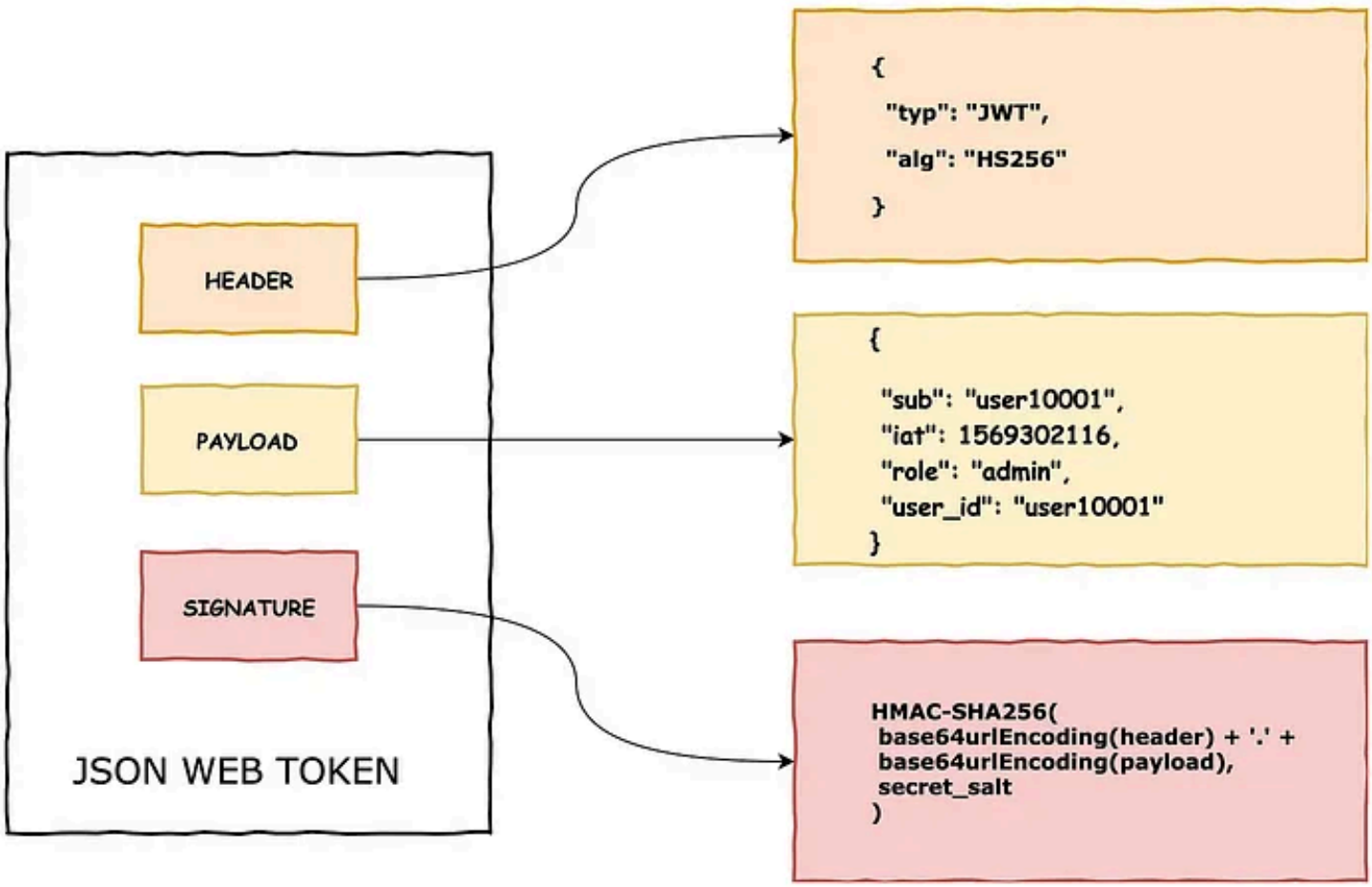
Now that we know the basics, it's time to dive deep and learn them in more detail so that you can answer any follow-up questions.

What is JWT (JSON Web Token)? When to use it?

As I said, JWT stands for JSON Web Token, a type of token used for securely transmitting information between parties. JWTs are commonly used for authentication and authorization purposes in web applications.

A JWT comprises three parts: **a header, a payload, and a signature**. The header specifies the token type and the signing algorithm, while the shipment contains the transmitted data.

> The signature is created by combining the header and payload with a secret key known only to the server.



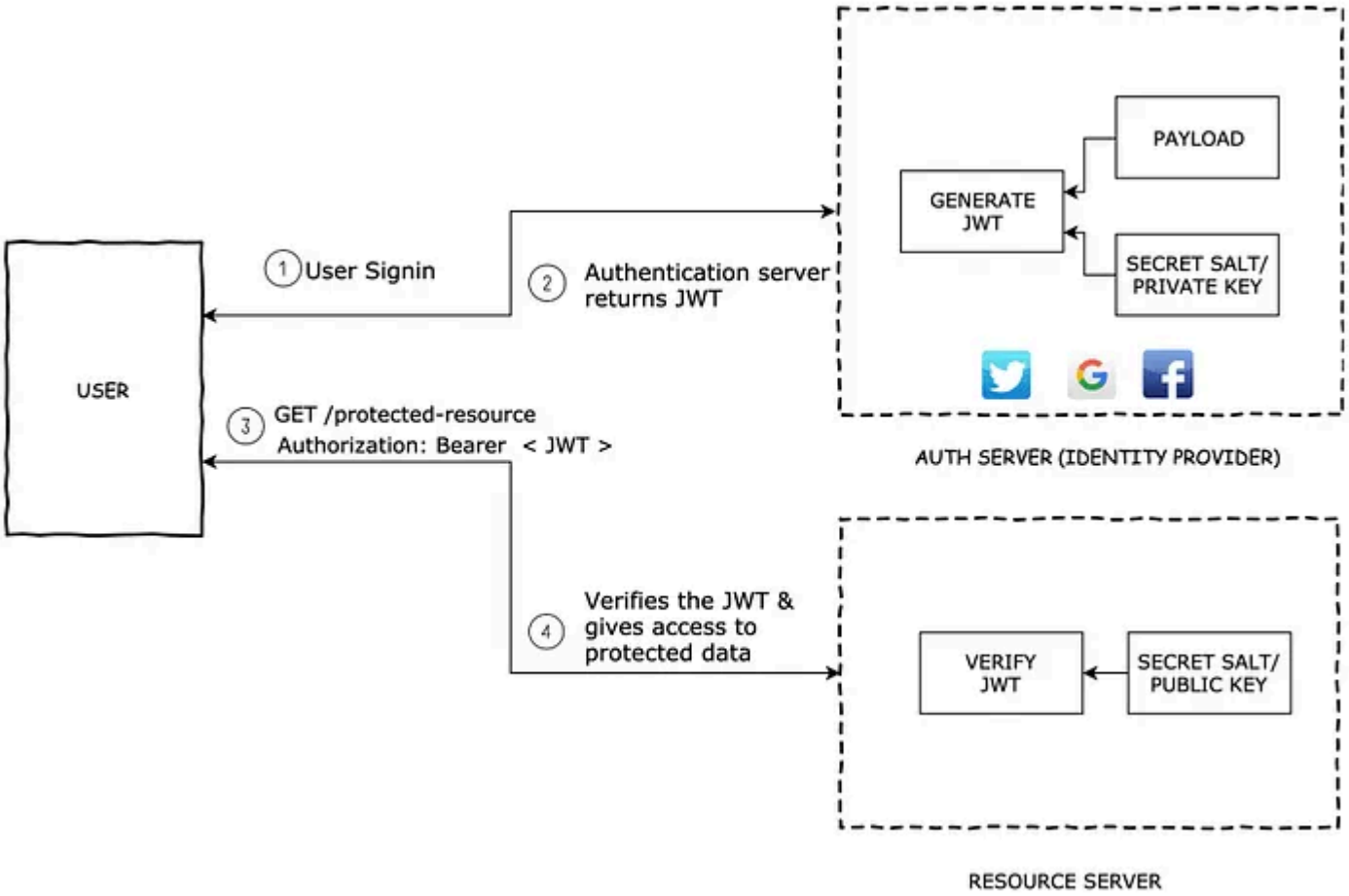
JWTs are commonly used in web applications to transmit user authentication data between the client and the server. When a user logs in, the server generates a JWT containing the user's ID and any relevant permissions or roles. This token is then sent back to the client, where it is stored and included in subsequent requests to the server.

> The server can then verify the authenticity of the JWT and use the information contained within to determine whether the user is authorized to perform the requested action.

JWTs are also helpful in distributed systems where multiple services need to share user authentication information. Instead of each service maintaining its authentication system, a single JWT can authenticate users across all services.

Overall, JWTs are helpful when transmitting sensitive information between parties securely and efficiently, especially in situations where traditional session-based authentication is not feasible.

Here is a nice diagram **that explains how JWT (JSON Web Tokens) works** in a web application:



What is OAuth2.0? When to use it?

OAuth2.0 is a protocol for authorization and authentication in web and mobile applications. It allows users to grant third-party applications access to their resources, such as their social media accounts or other online services, without giving away their credentials or passwords.

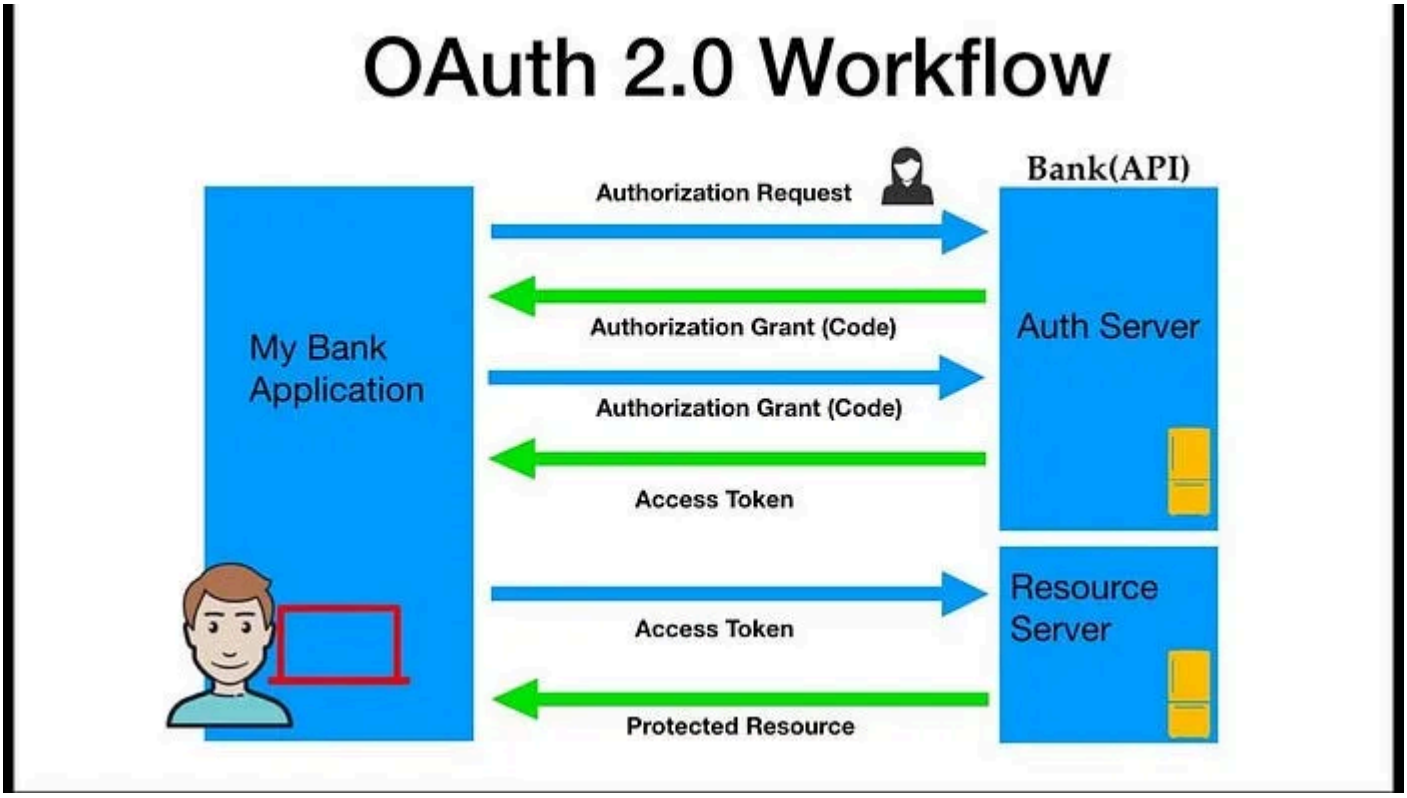
OAuth2.0 establishes trust between the user, the third-party application, and the resource server. The process involves several steps:

1. The user initiates the process by attempting to access a protected resource on the resource server, such as logging in to a social media account.
2. The resource server responds by redirecting the user to an authorization server, where they can grant permission for the third-party application to access their resources.
3. The user then logs in to the authorization server and grants permission for the third-party application to access their resources.
4. The authorization server generates an access token and sends it to the third-party application.

5. The third-party application uses the access token to request and access the user's resources on the resource server.

OAuth2.0 is useful when users want to grant third-party applications access to their resources but do not want to give away their credentials or passwords. It is commonly used in web and mobile applications, especially those that rely on external APIs or services for data and functionality.

Here is a nice diagram that explains OAuth2.0 workflow and working:



OAuth2.0 is also helpful in ensuring that users retain control over their resources and can revoke access anytime. It provides a secure and standardized way for users to share their data and help with third-party applications while maintaining privacy and security.

For example, most websites now allow you to log in using Twitter, Facebook, or Google accounts, where you don't need to create a new username or password, neither you have to share your Google, Twitter, or Facebook password to access any resource on the third-party website. However, you can still use it using OAuth.

What is SAML? When to use it?

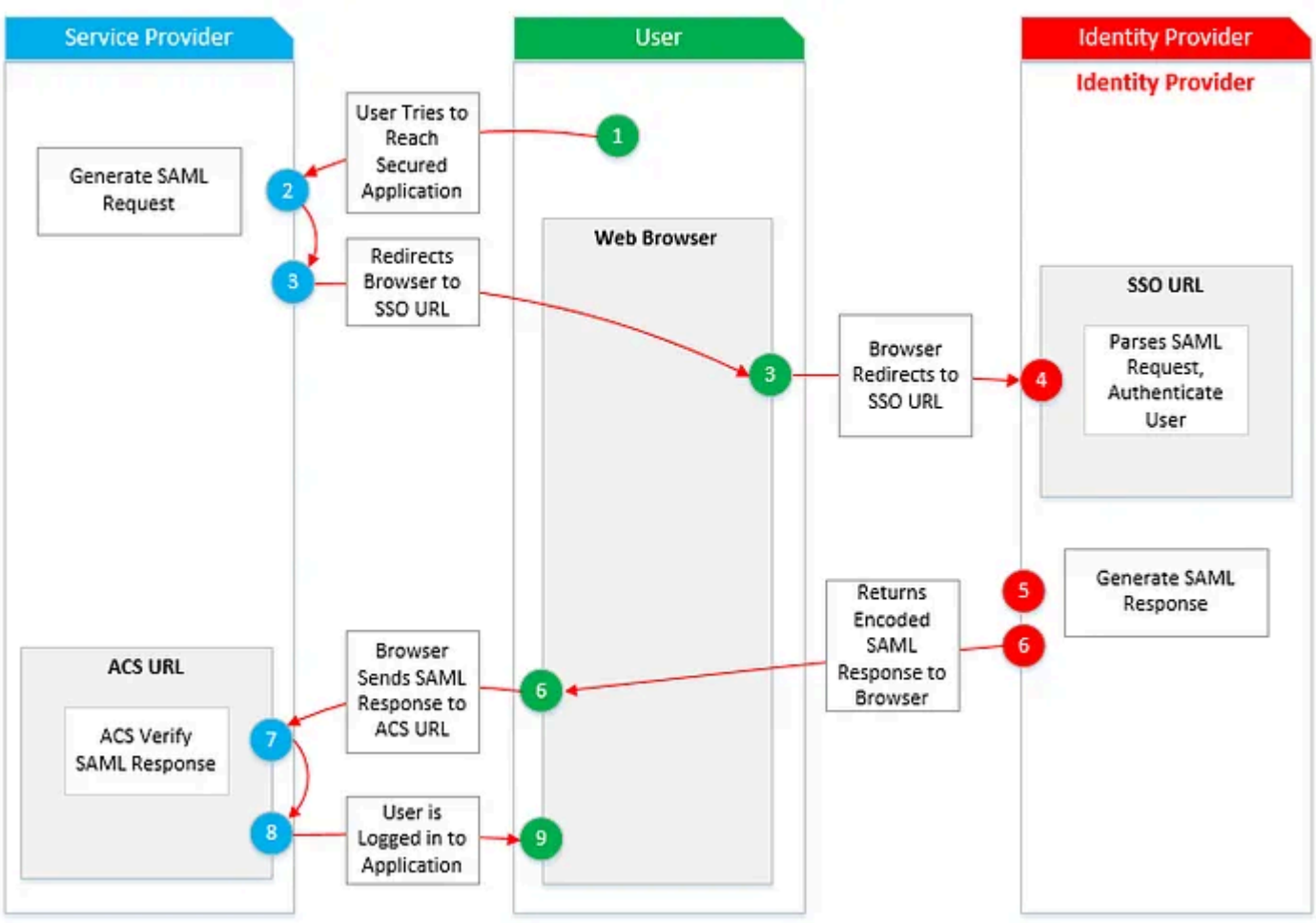
SAML stands for Security Assertion Markup Language and is an XML-based protocol for exchanging authentication and authorization data between parties, such as **identity providers** (**IdPs**) and **service providers** (**SPs**).

SAML works by establishing a trust relationship between the **IdP and SP**. The IdP is responsible for authenticating the user and generating a SAML assertion containing information about the user's identity and authentication status. The SP relies on the SAML assertion to make authorization decisions and grant access to protected resources.

SAML can be used in various scenarios, such as single sign-on (SSO) and federation.

In SSO, SAML allows users to access multiple applications or services with a single set of credentials. When a user logs in to one application or service, the IdP generates a SAML assertion, which can authenticate the user to other applications or services without requiring the user to log in again.

SAML enables trust relationships between organizations in a federation, allowing users to access resources across multiple organizations using a single set of credentials.



SAML is often used in enterprise environments, where users need to access multiple applications and services across different domains or organizations. It offers a standardized way to exchange authentication and authorization data, making it easier to manage access control and ensure the security of sensitive resources.

One of the famous examples of SAML usage is the SingPass authentication system by Singapore Government which allows you to access all government websites like CPF and IRAS

using Singapss login.

Overall, SAML is a powerful tool for enabling secure and seamless access to resources across different organizations and applications, making it a popular choice for many enterprises and organizations

JWT vs OAuth vs SAML

Now that we know what they are, how they work and where they are used, it's time to revise their critical differences. Here are some critical differences between JWT, OAuth, and SAML in point format:

JWT

- JWT is a token-based authentication mechanism.
- It transmits claims (user identity, permissions, etc.) between parties.
- It does not rely on a centralized authentication server or session state.
- It is commonly used in single-page applications (SPAs) and mobile applications.
- It can be used for authentication and authorization.

OAuth

- OAuth is a protocol for authorization and authentication in web and mobile applications.
- It is used for granting third-party applications access to resources on behalf of the user.
- It relies on a centralized authorization server.
- It is commonly used in applications that rely on external APIs or services for data and functionality.
- It is used for authorization, not authentication.

SAML

- SAML is a protocol for exchanging authentication and authorization data between parties, such as identity providers (IdPs) and service providers (SPs).
- It establishes trust relationships between organizations and enables single sign-on (SSO) and federation.
- It relies on a centralized identity provider (IdP).
- It is commonly used in enterprise environments.
- It is used for both authentication and authorization.

Here is a nice table that you can print to remember these differences between JWT, SAML, and OAuth2.0

	JWT	OAuth	SAML
Purpose	Token-based authentication mechanism for transmitting claims	Protocol for authorization and authentication in web and mobile apps	Protocol for exchanging authentication and authorization data between parties
Centralized Server	None	Authorization server	Identity provider
Used For	Authentication and authorization	Authorization, not authentication	Authentication and authorization
Applications	Single-page apps, mobile apps	Apps that rely on external APIs or services	Enterprise environments
Relationship	Directly between parties	Between third-party apps and resource owners	Between identity providers and service providers
Authentication	Yes	No	Yes
Authorization	Yes	Yes	Yes

In summary, JWT is a token-based authentication mechanism used for transmitting claims, OAuth is a protocol used for granting third-party applications access to resources on behalf of the user, and SAML is a protocol used for exchanging authentication and authorization data between parties to establish trust relationships between organizations.

That's all about **the difference between JWT, OAuth, and SAML for authentication and authorization**. In short, JWT is a standard for transmitting data securely between parties, while OAuth is a standard for approval that allows third-party applications to access user data.

SAML is a standard for exchanging authentication and authorization data between an IdP and an SP, typically used in enterprise applications. Each of these standards serves a different purpose, and they can all be used together to provide a secure and efficient authentication and authorization process for web applications.

My Books and Courses

If you are new here and want to support me, you can check out my books and courses; you can use the unique discount code "friends20" to get a 20% discount on my books.

1. **Grokking the Java Interview Questions**

   It covers:

   ➡️ OOP

   ➡️ Thraed

   ➡️ Collection

   ➡️ Stream

   ➡️ Lambda

   ➡️ JVM

   ➡️ Design Patterns

   ➡️ Generics

   Download the FREE PDF Sample here - https://bit.ly/3PywdMc

2. **Grokking the Spring Boot Interview**

   It covers:

   ➡️ Core Spring

   ➡️ Spring Boot

   ➡️ Spring MVC

   ➡️ Spring Data JPA

   ➡️ Spring Cloud

   ➡️ Security

   Download the FREE PDF Sample here - https://bit.ly/3PywdMc

Courses:

I have also created multiple courses for IT certification on Udemy, like Java, Spring, Azure, and AWS Cloud certifications; you can use them to prepare better for your IT certifications.

1. Java SE 17 - 1Z0-829 Certification
2. Java SE 11 - 1Z0-819 Certification
3. Spring Professional Certification
4. Java Fundamentals 1Z0-811 Certification
5. Azure Fundamentals Certification
6. AWS Cloud Practitioner certification
7. Java EE Application Developer Certification

Share Our Newsletter

If you've found our weekly helpful newsletter, please consider sharing it with a friend on Twitter, Facebook, LinkedIn, or any other social platform. You can even share by simply forwarding this email to them.

You can also send them to the **subscription page** by **clicking the subscribe now button below**.

Once again, thanks for reading this so far. Do let me know how you found this newsletter and what you want to see; your feedback is critical as it will drive how we produce this newsletter in

the future.

I plan to share one or two Java interview questions and essential concepts in every issue, but if you want more or want to see any particular type of content, do let us know.

All the best, and keep learning

To make this the best Java Newsletter, I also need your input, so please suggest what you like to see more:

POLL

What type of article you like to see here

- Deep Dive on Java Questions
- Spring Boot
- Microservices
- Data Structure and Algorithms
- System Design

0 VOTES ·

2 Likes

← Previous

Next →

A guest post by

Soma

Java and React Developer

Subscribe to Soma

Comments

Write a comment...