Java JWT Bibliothek

Um die Bibliothek `com.nimbusds.jwt` in Java zu verwenden, um Informationen wie das Ablaufdatum, das Ausstellungsdatum und andere Ansprüche aus einem JSON Web Token (JWT) zu extrahieren, befolgen Sie die unten aufgeführten Schritte. Diese Lösung geht davon aus, dass Sie mit einem signierten JWT arbeiten, was der häufigste Typ ist, und konzentriert sich darauf, einen klaren, praktischen Ansatz zu bieten, um Ihr Ziel zu erreichen.

Voraussetzungen

Stellen Sie zunächst sicher, dass die Nimbus JOSE + JWT-Bibliothek zu Ihrem Projekt hinzugefügt ist. Wenn Sie Maven verwenden, fügen Sie die folgende Abhängigkeit in Ihre `pom.xml` ein:

```
<dependency>
    <groupId>com.nimbusds</groupId>
    <artifactId>nimbus-jose-jwt</artifactId>
    <version>9.37</version> <!-- Ersetzen Sie durch die neueste Version -->
</dependency>
```

Sie können die neueste Version auf Maven Central überprüfen.

Schritte zum Extrahieren des Ablaufdatums, des Ausstellungsdatums und anderer Ansprüche

Hier erfahren Sie, wie Sie ein JWT analysieren und das Ablaufdatum, das Ausstellungsdatum und zusätzliche Ansprüche mit der `com.nimbusds.jwt`-Bibliothek abrufen können:

1. **Analysieren der JWT-Zeichenfolge**: Verwenden Sie die Methode `SignedJWT.parse()`, um die JWT-Zeichenfolge in ein `SignedJWT`-Objekt umzuwandeln.
2. **Abrufen des Claims-Sets**: Greifen Sie auf die Ansprüche (Schlüssel-Wert-Paare) aus dem JWT mit `getJWTClaimsSet()` zu.
3. **Extrahieren spezifischer Ansprüche**:
    - Verwenden Sie `getExpirationTime()` für das Ablaufdatum (`exp`-Anspruch).
    - Verwenden Sie `getIssueTime()` für das Ausstellungsdatum (`iat`-Anspruch).
    - Verwenden Sie `getSubject()`, `getClaim()` oder andere Methoden für zusätzliche Ansprüche.

4. **Fehlerbehandlung**: Umhüllen Sie die Analyselogik in einen try-catch-Block, um potenzielle Analysierungsprobleme zu verwalten.

Beispielcode

Hier ist ein vollständiges Java-Beispiel, das zeigt, wie Sie das Ablaufdatum, das Ausstellungsdatum und einen zusätzlichen Anspruch (z. B. den Betreff) aus einem JWT extrahieren:

```java
import com.nimbusds.jwt.SignedJWT;
import com.nimbusds.jwt.JWTClaimsSet;
import java.text.ParseException;
import java.util.Date;

public class JwtExtractor {

    public static void main(String[] args) {
        // Ersetzen Sie dies durch Ihre tatsächliche JWT-Zeichenfolge
        String jwtString = "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwiaWF0IjoxNTE2MjM5N

        try {
            // Schritt 1: Analysieren der JWT-Zeichenfolge
            SignedJWT signedJWT = SignedJWT.parse(jwtString);

            // Schritt 2: Abrufen des Claims-Sets
            JWTClaimsSet claimsSet = signedJWT.getJWTClaimsSet();

            // Schritt 3: Extrahieren von Ablauf- und Ausstellungsdaten
            Date expirationDate = claimsSet.getExpirationTime();
            Date issuedDate = claimsSet.getIssueTime();
            String subject = claimsSet.getSubject(); // Beispiel für einen weiteren Anspruch

            // Schritt 4: Anzeigen der Ergebnisse
            if (expirationDate != null) {
                System.out.println("Ablaufdatum: " + expirationDate);
            } else {
                System.out.println("Kein Ablaufdatum festgelegt.");
            }
```

```
            if (issuedDate != null) {
                System.out.println("Ausstellungsdatum: " + issuedDate);
            } else {
                System.out.println("Kein Ausstellungsdatum festgelegt.");
            }

            if (subject != null) {
                System.out.println("Betreff: " + subject);
            } else {
                System.out.println("Kein Betreff festgelegt.");
            }

        } catch (ParseException e) {
            System.out.println("Ungültiges JWT: " + e.getMessage());
        }
    }
}
```

Erklärung des Codes

1. Imports

- `SignedJWT`: Stellt ein signiertes JWT dar und bietet Methoden zum Analysieren und Verarbeiten.
- `JWTClaimsSet`: Enthält die Ansprüche aus dem JWT-Payload.
- `ParseException`: Wird ausgelöst, wenn die JWT-Zeichenfolge fehlerhaft ist oder nicht analysiert werden kann.
- `Date`: Wird verwendet, um die Ablauf- und Ausstellungszeiten darzustellen.

2. Analysieren des JWT

- Die Methode `SignedJWT.parse(jwtString)` nimmt eine JWT-Zeichenfolge (z. B. `header.payload.signature`) und gibt ein `SignedJWT`-Objekt zurück. Wenn das JWT ungültig ist, wird eine `ParseException` ausgelöst.

3. Zugreifen auf die Ansprüche

- `signedJWT.getJWTClaimsSet()` ruft das Claims-Set ab, das alle Ansprüche aus dem Payload des JWT enthält.

4. Extrahieren spezifischer Ansprüche

- `getExpirationTime()`: Gibt den `exp`-Anspruch als `Date`-Objekt (oder `null`, wenn nicht vorhanden) zurück. Dies stellt dar, wann das Token abläuft.
- `getIssueTime()`: Gibt den `iat`-Anspruch als `Date`-Objekt (oder `null`, wenn nicht vorhanden) zurück. Dies gibt an, wann das Token ausgestellt wurde.
- `getSubject()`: Gibt den `sub`-Anspruch als `String` (oder `null`, wenn nicht vorhanden) zurück, ein Beispiel für einen weiteren Standardanspruch. Sie können auch `getClaim("key")` verwenden, um benutzerdefinierte Ansprüche als `Object` abzurufen.

5. Fehlerbehandlung

- Der try-catch-Block stellt sicher, dass das Programm bei einem fehlerhaften oder ungültigen JWT den Fehler anmutig behandelt, indem es eine Fehlermeldung ausgibt.

Hinweise

- **Signierte vs. unsignierte JWTs**: Dieses Beispiel verwendet `SignedJWT` für signierte Token. Wenn Sie ein unsigniertes JWT haben, verwenden Sie `PlainJWT.parse(jwtString)` stattdessen. Für einen generischeren Ansatz können Sie `JWTParser.parse(jwtString)` verwenden und dann den Typ (`SignedJWT`, `PlainJWT` usw.) mit `instanceof` überprüfen.
- **Signaturverifizierung**: Dieser Code verifiziert die Signatur des JWT nicht. In einer Produktionsumgebung sollten Sie die Signatur mit `signedJWT.verify(verifier)` und einem geeigneten Schlüssel überprüfen, um die Authentizität des Tokens sicherzustellen.
- **Null-Überprüfungen**: Überprüfen Sie immer auf `null`, wenn Sie Ansprüche abrufen, da sie optional sind und in jedem JWT nicht vorhanden sein können.
- **Benutzerdefinierte Ansprüche**: Um benutzerdefinierte Ansprüche zuzugreifen, verwenden Sie `claimsSet.getClaim("claimName")`, das ein `Object` zurückgibt, das Sie in den entsprechenden Typ (z. B. `String`, `Integer`) umwandeln können.

Beispielausgabe

Für die obige Beispiel-JWT-Zeichenfolge:

```
Ablaufdatum: Mi Jan 17 19:52:02 UTC 2018
Ausstellungsdatum: Mi Jan 17 19:50:22 UTC 2018
Betreff: 1234567890
```

Dieser Ansatz bietet eine einfache und effektive Möglichkeit, das Ablaufdatum, das Ausstellungsdatum und andere Ansprüche aus einem JWT mit der `com.nimbusds.jwt`-Bibliothek zu extrahieren. Ersetzen Sie die `jwtString` durch Ihr eigenes Token und passen Sie den Code an Ihre spezifischen Anforderungen an!