LU06b - Autorisation in einer Webapplikation mit JSON Web Token

Token im Client

Der Client erhält vom Authorisationsservice ein Token. Dieses muss er lokal speichern und bei jedem Request mitsenden.

Token speichern

Ein Webbrowser als Client hat verschiedene Möglichkeiten, um Daten zu speichern. Je nach Art des Speichers haben wir gewisse Vor- und Nachteile.

Cookies

In Cookies können Daten gespeichert und weitergeleitet werden. Cookies werden bei einem Request automatisch durch den Webbrowser mitgesendet. Diese haben einen schlechten Ruf, da sie oftmals zur Verfolgung von Benutzeraktivitäten über verschiedene Webseiten hinweg verwendet wurden bzw. noch werden.

Local Storage

Auch der Local Storage bietet eine Datenablage. Im Gegensatz zu Cookies werden Werte nicht automatisch gespeichert und gesendet. Beides muss explizit durch Javascript programmiert werden. Die Daten im Local Storage haben kein Ablaufdatum und werden auch über mehrere Browser Sessions hinweg geteilt.

Session Storage

Der Unterschied zum Local Storage besteht darin, dass Session Storage nur für eine Browser Session gilt. Die Daten werden automatisch gelöscht, wenn die Seite geschlossen wird. Da ein Authentifizierungs-Token sowieso nicht lange gültig ist, bietet sich aus meiner Sicht der Session Storage als Speicherort an.

Dateien

Ein Webbrowser kann auch Daten in einer Datei auf dem Clientrechner speichern.

Speichern des Tokens im Session Storage

Über den Bezeichner „sessionStorage“ können wir diesen Speicher verwalten. Die Methode `setItem` speichert meine Daten als Key/Value-Pair:

```
data = ...
sessionStorage.setItem("token", data);
```

Token senden

Bei jedem Request muss der Client sein Token mitsenden. Dazu wird beim Request ein zusätzlicher `Authorization`-Header gesendet. In diesem Header ist es üblich, das Schlüsselwort `Bearer ` vor das eigentliche Token zu stellen.

```
token = sessionStorage.getItem("token");
fetch("./????",
    {
        method: "POST",
        headers: {
            "Content-Type": "application/x-www-form-urlencoded",
            "Authorization": "Bearer " + token
        },
        body: data
    })
...
```

Token überprüfen

In unserer Webapplikation können wir über den Request-Header `Authorization` auf das mitgesendete Token zugreifen. Zunächst prüfen wir, ob ein `Authorization`-Header vorhanden ist. Danach können wir das Token ab Position 8 aus diesem Header lesen.

Wir stellen ja beim Senden das Schlüsselwort `Bearer ` vor das eigentliche Token. Daher müssen wir diese 7 Zeichen beim Verarbeiten des Tokens wieder entfernen.

Anschliessend müssen wir prüfen ob das Token gültig ist. Die einzelnen Schritte sind:

1. Token mit dem geheimen Schlüssel entschlüsseln.
2. Prüfen ob die Signatur korrekt ist.
3. Prüfen ob das Ablaufdatum noch nicht erreicht ist.

Zum Glück sind viele der komplizierteren Aufgaben in entsprechenden Bibliotheken für uns gelöst. Schaue dir dazu das Umsetzungsbeispiel an.

M321-LU06