Sesje i ciasteczka w Node.js + Express

INF.04.7.1(19), INF.04.7.3(2)

---

1. Dlaczego potrzebne są sesje i ciasteczka?

- Protokół **HTTP jest bezstanowy** – każde żądanie od klienta do serwera jest niezależne i serwer „nie pamięta" poprzednich działań.
- Aby rozpoznać użytkownika i utrzymać informacje między żądaniami, stosuje się:
  - **Ciasteczka (cookies)** – dane zapisane w przeglądarce.
  - **Sesje (sessions)** – dane zapisane na serwerze, identyfikowane przez ID w ciasteczku.

---

2. Ciasteczka (Cookies)

Definicja

- To małe pliki tekstowe przechowywane w przeglądarce użytkownika.
- Serwer wysyła nagłówek `Set-Cookie`, a przeglądarka odsyła ciasteczko w nagłówku `Cookie` przy kolejnych żądaniach.

Właściwości ciasteczek

- **name=value** – para klucz-wartość.
- **Expires / Max-Age** – czas ważności.
- **Path** – zakres działania (np. `/admin`).
- **Domain** – domena, dla której ciasteczko jest ważne.
- **Secure** – ciasteczko wysyłane tylko przez HTTPS.
- **HttpOnly** – ciasteczko niedostępne z poziomu JavaScript (chroni przed XSS).
- **SameSite** – ogranicza wysyłanie ciasteczek w żądaniach cross-site (ochrona przed CSRF).

Instalacja i użycie w Express

```
npm install express cookie-parser
```

Przykład kodu

```
const express = require('express');
const cookieParser = require('cookie-parser');

const app = express();
app.use(cookieParser());

// ustawienie ciasteczka
app.get('/set-cookie', (req, res) => {
```

```
  res.cookie('language', 'pl', {
    maxAge: 3600000,
    httpOnly: true,
    sameSite: 'lax'
  });
  res.send('Ciasteczko ustawione');
});

// odczyt ciasteczka
app.get('/get-cookie', (req, res) => {
  const lang = req.cookies.language || 'brak';
  res.send(`Wybrany język: ${lang}`);
});

app.listen(3000, () => console.log('Serwer działa na http://localhost:3000'));
```

---

3. Sesje (Sessions)

Definicja

- Sesja to sposób przechowywania danych **po stronie serwera**.
- Użytkownik otrzymuje ciasteczko zawierające **ID sesji** (np. `connect.sid`).
- Na serwerze dane są powiązane z tym ID (np. dane logowania, koszyk).

Zalety sesji

- Dane nie są przechowywane w przeglądarce (większe bezpieczeństwo).
- Możliwość przechowywania dużych obiektów.
- Łatwa implementacja logowania i kontroli dostępu.

Instalacja i użycie w Express

```
npm install express express-session
```

Przykład kodu

```
const express = require('express');
const session = require('express-session');

const app = express();

app.use(session({
  secret: 'tajny_klucz',       // klucz do podpisywania ciasteczek
  resave: false,               // nie zapisuje sesji bez zmian
  saveUninitialized: true,     // zapisuje nową pustą sesję
  cookie: {
    secure: false,             // przy HTTPS ustawić true
```

```
    httpOnly: true,
    maxAge: 3600000            // 1 godzina
  }
}));

// ustawienie sesji
app.get('/set-session', (req, res) => {
  req.session.user = 'Jan Kowalski';
  res.send('Sesja ustawiona dla użytkownika Jan Kowalski');
});

// odczyt sesji
app.get('/get-session', (req, res) => {
  if (req.session.user) {
    res.send(`Użytkownik w sesji: ${req.session.user}`);
  } else {
    res.send('Brak sesji');
  }
});

app.listen(3000, () => console.log('Serwer działa na http://localhost:3000'));
```

---

4. Jak to działa w Express?

1. Użytkownik odwiedza stronę i loguje się.
2. Serwer zapisuje dane użytkownika w sesji (`req.session.user`).
3. Serwer wysyła ciasteczko z identyfikatorem sesji (`connect.sid`).
4. Przy każdym kolejnym żądaniu przeglądarka wysyła ciasteczko z ID sesji.
5. Express odczytuje dane powiązane z tym ID i „pamięta”, że użytkownik jest zalogowany.

---

5. Porównanie sesji i ciasteczek

Cecha	Cookies (ciasteczka)	Sesje (sessions)
Gdzie są dane?	W przeglądarce użytkownika	Na serwerze
Bezpieczeństwo	Mniejsze (dane można podejrzeć)	Większe (tylko ID w ciasteczku)
Rozmiar danych	do ok. 4 KB	brak sztywnego limitu
Czas życia	do daty wygaśnięcia / zamknięcia przeglądarki	do zamknięcia sesji / wylogowania
Typowe zastosowania	preferencje (język, motyw), token JWT	logowanie, koszyk, profil użytkownika

---

6. Zastosowania praktyczne

- **Ciasteczka**:

  - zapamiętywanie preferencji użytkownika,
  - przechowywanie tokenów uwierzytelniających (np. JWT),
  - śledzenie użytkowników (np. Google Analytics).

- **Sesje**:
  - logowanie i autoryzacja użytkowników,
  - przechowywanie koszyka w sklepie internetowym,
  - implementacja paneli administratora,
  - tymczasowe dane użytkownika (np. formularze wieloetapowe).

---

7. Rozszerzenie – przechowywanie sesji

Domyślnie dane sesji w Express przechowywane są w pamięci RAM, co nie nadaje się do aplikacji produkcyjnych.
Popularne opcje przechowywania sesji:

- **Redis** – bardzo szybka baza NoSQL w pamięci.
- **MongoDB** – sesje przechowywane w kolekcji dokumentów.
- **MySQL/PostgreSQL** – sesje w bazie relacyjnej.
- **FileStore** – sesje zapisywane do plików.

Przykład użycia Redis (z pakietem `connect-redis`):

```
npm install connect-redis redis
```

```
const RedisStore = require('connect-redis')(session);
const redis = require('redis').createClient();

app.use(session({
  store: new RedisStore({ client: redis }),
  secret: 'tajny_klucz',
  resave: false,
  saveUninitialized: false
}));
```

---

8. Bezpieczeństwo sesji i ciasteczek

- Ustawiaj ciasteczka z flagami `HttpOnly`, `Secure`, `SameSite`.
- Wygaszaj sesje po określonym czasie (`maxAge`).
- W przypadku aplikacji wrażliwych stosuj **rotację identyfikatora sesji** po logowaniu.
- Stosuj ochronę **CSRF** (Cross-Site Request Forgery).
- Nie zapisuj haseł ani poufnych danych bezpośrednio w sesji.

---

9. Podsumowanie

- **Cookies** → małe dane w przeglądarce (np. język, motyw).
- **Sessions** → dane użytkownika na serwerze, identyfikowane przez ciasteczko.
- **Express** ułatwia pracę z oboma mechanizmami poprzez pakiety `cookie-parser` i `express-session`.
- Sesje i ciasteczka to podstawowe narzędzia w **logowaniu, koszykach zakupowych i personalizacji aplikacji webowych**.