DR INŻ. PIOTR SIEWNIAK

Routing

WPROWADZENIE

W ogólności, „ruting” (*routing*) definiuje sposób, w jaki aplikacja odpowiada na żądanie HTTP (HTTP request) klienta skojarzone z określonym punktem końcowym (*endpoint*). Punkt końcowy jest definiowany za pomocą adresu URL (lub ścieżki/podścieżki) wraz z określoną metodą żądania HTTP (np. GET, POST itd.). Dany punkt końcowy (*route*) może mieć przypisaną jedną lub więcej funkcji obsługi (*route handler*).

W języku polskim punkty końcowe nazywane są „rutami”, trasami lub ścieżkami. *Routing* jest nazywany po prostu „rutingiem”, „routingiem” lub „trasowaniem”. Z kolei angielskie określenie *router* jest tłumaczone na język polski jako „ruter”, „router” lub „trasownik”.

Implementację routingu w aplikacji Node Express można realizować przy wykorzystaniu zasobów:
a) obiektu aplikacji `app` (`const app = express()`),
b) obiektu `router` (`const router = express.Router()`).

W praktyce, obsługa routingu, czyli żądań HTTP na zadanych ścieżkach/podścieżkach (punktach końcowych), jest realizowana za pomocą odpowiednich metod obiektów `app` i `router`. Np. metoda

```
app.get(ścieżka, callback)
```

umożliwia obsługę żądania GET na ścieżce `ścieżka` za pomocą funkcji pośredniej definiowanej jako jej argument w postaci funkcji zwrotnej `callback`. Funkcja `callback` stanowi funkcję obsługi routa (*route handler*). W ogólności, żądanie na określonej ścieżce może mieć wiele funkcji obsługi:

```
app.get(ścieżka, callback1, callback2, ...)
```

Pominięcie ścieżki w wywołaniu metody app.get()

```
app.get(callback1, callback2, ...)
```

oznacza, że funkcje obsługi routa (`callback1`, `callback2` itd) zostaną wykonane dla każdego żądania GET, niezależnie od adresu URL żądania.

Podobne zasady dotyczą metody `router.get()` z tym, że `ścieżka` jest w tym przypadku rozumiana jako podścieżka aplikacji ustawiona (lub nie) przy wykorzystaniu metody `app.use()`. W przypadku wywołania metody `app.use()` bez podawania ścieżki, serwer Express zakłada, że ścieżkę aplikacji stanowi ścieżka domyślna, czyli `/`. Tym samym, funkcja pośrednia stanowiąca jej argument zostanie zamontowana na wszystkich ścieżkach aplikacji, niezależnie od typu żądania HTTP.

W aplikacji można zdefiniować pojedynczy router lub więcej. W pierwszym przypadku, różne punkty końcowe (np. adresy URL podstron aplikacji) są obsługiwane przez ten pojedynczy router. W drugim przypadku, poszczególnym punktom końcowym aplikacji mogą odpowiadać różne routery.

W ogólności, metoda `app.use()` służy do zamontowania określonej funkcji pośredniej lub funkcji pośrednich na zadanej ścieżce/ścieżkach, niezależnie od metody HTTP żądania (np. GET, POST). W przypadku wywołania metody `app.use()` bez podawania ścieżki, serwer Express montuje funkcje pośrednie dla wszystkich ścieżkach.

Z drugiej strony wywołanie:

```
app.use(ścieżka, callback)
```

powoduje zamontowanie funkcji pośredniej `callback` wyłącznie na ścieżce `ścieżka`, niezależnie od typu żądania HTTP. W tym przypadku, funkcja `callback` stanowi funkcję obsługi routa `ścieżka` dla każdego żądania HTTP.

Z kolei metoda `router.use()` pozwala na zamontowanie funkcji pośredniej (lub funkcji pośrednich):

1) albo dla zadanej podścieżki lub podścieżek;
2) albo dla wszystkich podścieżek (względem ścieżki aplikacji)

obsługiwanych przez OKREŚLONY router. Widać stąd, że działanie metody `router.use()` jest analogiczne do metody `app.use()` z tą różnicą, że metoda `app.use()` operuje na ścieżkach aplikacji, a `router.use()` – na podścieżkach ściezki aplikacji zdefiniowanej za pomocą `app.use()`.
Brak określonej podścieżki oznacza, że jako obowiązująca przyjmowana jest podścieżka domyślna /.