HTMX – Lekki framework do dynamicznych interfejsów

Krzysztof W

Wstęp

9 grudnia 2025

Filozofia HTMX: HTML-over-the-wire

Radykalna Różnica w Podejściu

HTMX rozszerza możliwości HTML5 (AJAX, WebSockets) bez pisania kodu JavaScript. Celem jest uproszczenie rozwoju i **powrót logiki renderowania na Serwer**.

Kluczowe Założenia:

- **Cienki Klient:** Przeglądarka ma tylko renderować otrzymany HTML.
- **Deklaratywność:** Zamiast kodu, używamy atrybutów HTML do definiowania akcji.
- **Redukcja złożoności:** Mniej kodu JS = mniej błędów i szybszy czas reakcji.

Cykl Żądania i Kluczowe Atrybuty

Cała logika interakcji jest definiowana w znacznikach, co tworzy architekturę **HTML-over-the-wire**.

Główne Atrybuty (Mechanika):

- **Akcja i Cel:** `hx-post="/api/login"` (co i gdzie wysłać)
- **Wyzwalacz:** `hx-trigger="click"` (kiedy akcja ma się wydarzyć)
- **Odbiorca:** `hx-target="#main-content"` (który element DOM ma być zaktualizowany)
- **Sposób Wstawienia:** `hx-swap="innerHTML"` (jak podmienić treść)

Wszystkie te atrybuty wspólnie tworzą dynamiczną, asynchroniczną interakcję bez konieczności pisania `fetch` w JavaScript.

Backend jako Renderujący Silnik UI

W tym modelu backend pełni funkcję renderującą, co jest kluczowe dla **szybkości i spójności** interfejsu.

Rola Serwera:

- Walidacja wszystkich danych i operacji (np. rejestracji).
- Przetwarzanie logiki biznesowej (np. generowanie JWT).
- Zwracanie **gotowych fragmentów HTML** (widoków), a nie surowych danych.

Logika Danych:

- Zapewnia to natychmiastową reakcję na akcje (rejestracja, logowanie) bez obciążania zewnętrznym API.

Wyzwanie: Uwierzytelnianie w Świecie HTML

Tokeny JWT wymagają obsługi w nagłówkach HTTP, co standardowo wymaga kodu JS. HTMX wymaga minimalnego "kleju".

Mechanizm Wstrzykiwania Tokenów:

1. **Wysyłanie:** Używamy zdarzenia `htmx:configRequest` (tuż przed wysłaniem) do pobrania JWT z `localStorage` i dodania go do nagłówka `Authorization`.
2. **Odbiór:** Używamy `htmx:afterRequest` (po otrzymaniu odpowiedzi) do przechwycenia nowego tokena (z nagłówka odpowiedzi, np. `X-Auth-Token`) i zapisania go.

Wnioski Końcowe

- **Minimalny Narzut (Overhead):** Udało się stworzyć pełnoprawną aplikację (logowanie, rejestracja, chroniony widok) przy ekstremalnie małej ilości kodu JS.
- **Spójność Logiki:** Logika backendowej i logika frontendowej są scalone na serwerze, co ułatwia debugowanie.
- **UX na Poziomie SPA:** Płynna, dynamiczna interakcja bez przeładowań całej strony.

Dziękuję. Przejdźmy teraz do prezentacji kodu i działania aplikacji.