

Mini-MVC: front controller i routing

Czas: 2 × 45 min

praca indywidualna

Cel: Przebudowa prostej aplikacji CRUD na architekturę MVC z jednym punktem wejścia, routerem i przyjaznymi adresami URL.

Po zajęciach potrafisz:

- wyjaśnia rolę modelu, widoku i kontrolera oraz front controllera
- kieruje wszystkie żądania do `index.php` regułą `mod_rewrite` w `.htaccess`
- pisze router dopasowujący metodę HTTP i wzorec adresu z parametrem `{id}`
- przenosi zapytania SQL do klasy modelu, a HTML do widoków ze wspólnym układem
- odróżnia adresy zasobów (`/filmy/5`) od plików skryptów (`edytuj.php?id=5`)

Sprzęt i materiały

- komputer z Windows 11, XAMPP (Apache z `mod_rewrite`), Composer
- aplikacja z zadania 9 albo folder `r2_17_start` i plik `r2_17_katalog.sql`

Oprogramowanie

- **XAMPP** — `mod_rewrite` i `AllowOverride All` dla `htdocs` są włączone w domyślnej konfiguracji — sprawdź w `httpd.conf`, jeśli adresy nie działają
apachefriends.org/download.html
- **Composer** — tylko autoloader (bez pakietów)
getcomposer.org/download
- **Visual Studio Code**
code.visualstudio.com

Przygotowanie przed zajęciami

- Przypomnij sobie autoloader PSR-4 z zadania 16 i działanie aplikacji z zadania 9.

Bezpieczeństwo

- Pliki konfiguracji, modele i widoki umieść poza folderem `public/` — z przeglądarki dostępny jest tylko `public/`.
- Usuwanie wykonuj wyłącznie metodą POST; zachowaj zapytania przygotowane i escapowanie z zadania 9.

Przebieg zadania

1 Analiza i struktura projektu

Zaimportuj `r2_17_katalog.sql`, skopiuj `r2_17_start` (lub zadanie 9) do `htdocs\katalog_stary` i sprawdź działanie. Nowy projekt utwórz w `htdocs\katalog_mvc`.

```
katalog_mvc/  
  app/Controllers/FilmController.php  app/Models/Film.php  
  app/Router.php  app/View.php  app/helpers.php  
  views/layout.php  views/404.php  views/filmy/{lista,pokaz,formularz}.php  
  public/index.php  public/.htaccess  public/style.css  
  config.php  composer.json
```

1. W tabeli w sprawozdaniu przypisz każdy fragment starej aplikacji (zapytania SQL, walidacja, HTML, obsługa POST, przekierowania) do warstwy: model, widok, kontroler.
2. Utwórz strukturę z ramki. W `composer.json` ustaw autoload `"App\\": "app/"` oraz `"files": ["app/helpers.php"]`, uruchom `composer dump-autoload`. W `config.php` zwróć tablicę z danymi połączenia (`return [...]`).

2 Front controller, .htaccess i router

Docelowe trasy aplikacji:

GET /	→ FilmController::index	lista
GET /filmy/dodaj	→ create	formularz
POST /filmy	→ store	zapis nowego
GET /filmy/{id}	→ show	szczegóły
GET /filmy/{id}/edytuj	→ edit	formularz edycji
POST /filmy/{id}	→ update	zapis zmian
POST /filmy/{id}/usun	→ destroy	usunięcie

1. Utwórz `public/.htaccess`: `RewriteEngine On`, dwa warunki `RewriteCond %{REQUEST_FILENAME} !-f` i `!-d`, reguła `RewriteRule ^ index.php [QSA,L]`. Wyjaśnij, po co są warunki.
2. W `app/helpers.php` napisz `baza_url()` (katalog z `$_SERVER['SCRIPT_NAME']`), np. `/katalog_mvc/public`), `url($sciezka)`, `e()` i `przekieruj($sciezka, $flash)` (nagłówek Location + `exit`).
3. Napisz klasę `App\Router` z metodami `get()`, `post()` i `dispatch($metoda, $uri)`: usuń z adresu część bazową i zapytanie (`parse_url`), zamień `{id}` we wzorcu na `(\d+)`, przy dopasowaniu wywołaj akcję z parametrami jako `int`; brak dopasowania → kod 404 i widok `404`.
4. W `public/index.php` (front controller) dołącz autoload, uruchom sesję, utwórz PDO z `config.php`, model i kontroler, zarejestruj trasy z ramki i wywołaj `dispatch()`.

3 Model, widoki i kontroler

1. Klasa `App\Models\Film` dostaje PDO w konstruktorze i ma metody `wszystkie()`, `znajdz(int $id): ?array`, `dodaj(array): int` (zwraca `lastInsertId`), `zmien()`, `usun()` oraz statyczną `waliduj(array $d): array` zwracającą `[dane, błędy]` (przenieś walidację ze starej aplikacji). Gatunki jako stała klasy.
2. Klasa `App\View` z metodą statyczną `render($nazwa, $dane): extract($dane)`, bufor `ob_start()` dla widoku, a wynik (`$trec`) wstawiony do `views/layout.php`. Układ zawiera menu, komunikat flash i `<link>` do `url('/style.css')`.
3. `FilmController` ma 7 akcji z tabeli tras. Po udanym zapisie — przekierowanie do `/filmy/{id}` z komunikatem flash; przy błędach walidacji formularz z błędami i wpisanymi danymi; nieistniejący film — 404.
4. Widoki: lista (tytuł jako odnośnik do szczegółów, Edytuj, przycisk Usuń w formularzu POST), szczegóły, jeden wspólny formularz dla dodawania i edycji (atrybut `action` z kontrolera), strona 404. Wszystkie dane przez `e()`.

4 Testy i dokumentacja

Aplikację otwieraj pod `http://localhost/katalog_mvc/public/`.

1. Przetestuj wszystkie trasy, adres nieistniejący (`/aktorzy`) i nieistniejący film (`/filmy/999`). W F12 → Network zanotuj kody odpowiedzi (200, 302, 404) dla 5 różnych żądań.
2. Porównaj adresy starej i nowej aplikacji w tabeli (np. `edytuj.php?id=3` → `/filmy/3/edytuj`). Zrób zrzuty listy, formularza z błędami i strony 404, spakuj projekt bez `vendor/` jako `nazwisko_imie_zad17.zip`.

W sprawozdaniu umieść

- ☐ tabela przypisania fragmentów starej aplikacji do warstw M, V, C
- ☐ fragmenty kodu: `.htaccess`, `Router::dispatch()`, jedna akcja kontrolera, `View::render()`
- ☐ tabela adresów: stara aplikacja → nowa aplikacja, z metodą HTTP i kodem odpowiedzi
- ☐ zrzuty: lista, formularz z błędami, 404; archiwum `nazwisko_imie_zad17.zip`
- ☐ odpowiedzi na pytania kontrolne

Pytania kontrolne

1. Jakie zadanie ma front controller i co zyskujemy, gdy wszystkie żądania przechodzą przez jeden plik?
2. Dlaczego model nie powinien zawierać HTML, a widok zapytań SQL?
3. Po co w regule `mod_rewrite` warunki `!-f` i `!-d`? Co by się stało z plikiem `style.css` bez nich?
4. Dlaczego usuwanie jest trasą POST, a nie odnośnikiem GET?

Kryteria oceny

Kryterium	Punkty
Struktura katalogów, autoloading, konfiguracja poza <code>public/</code> , tabela M/V/C	3
<code>.htaccess</code> i front controller z 7 trasami	3
Router: parametr <code>{id}</code> , ścieżka bazowa, metoda HTTP, 404	4
Model z zapytaniami przygotowanymi i walidacją; View z układem i flash	4
Kontroler: 7 akcji, PRG, błędy walidacji, 404 dla brakującego filmu	3
Testy z kodami odpowiedzi, zrzuty, archiwum, odpowiedzi na pytania	3
Razem	20

Sprawozdanie przygotuj na wzorze dołączonym do zadania (plik .docx o tym samym numerze).