Mechanizmy pamięci podręcznej i optymalizacja

Projektowanie i programowanie systemów internetowych I

mgr inż. Krzysztof Rewak

19 maja 2018

Wydział Nauk Technicznych i Ekonomicznych
Państwowa Wyższa Szkoła Zawodowa im. Witelona w Legnicy

Plan prezentacji

1. Pamięć podręczna
2. Cache przeglądarki internetowej
3. Cache HTTP
4. Cache bazy danych
5. Podsumowanie

Pamięć podręczna

Z definicji

Uogólniając, **pamięć podręczna** to pamięć o krótszym czasie dostępu niż standardowo wykorzystywana pamięć.

Z definicji

Podstawowym celem wykorzystania pamięci podręcznej jest zatem zwiększenie szybkości dostępu do *pewnych* danych.

Pewne dane?

Jakie informacje powinny być przechowywane w pamięci podręcznej?

Dowolne.

Pewne dane?

Jakie informacje powinny być przechowywane w pamięci podręcznej?

Dowolne, ale najsensowniejszym wyjściem jest przechowywanie danych, które najprawdopodobniej według wybranych wskaźników będą wykrozystane w najbliższym czasie.

Gdzie można znaleźć pamięć podręczną?

- procesor
- dysk twardy
- system plików

Gdzie można znaleźć pamięć podręczną?

- przeglądarka internetowa
- pomiędzy przeglądarką a serwerem HTTP
- bazy danych

Cache przeglądarki internetowej

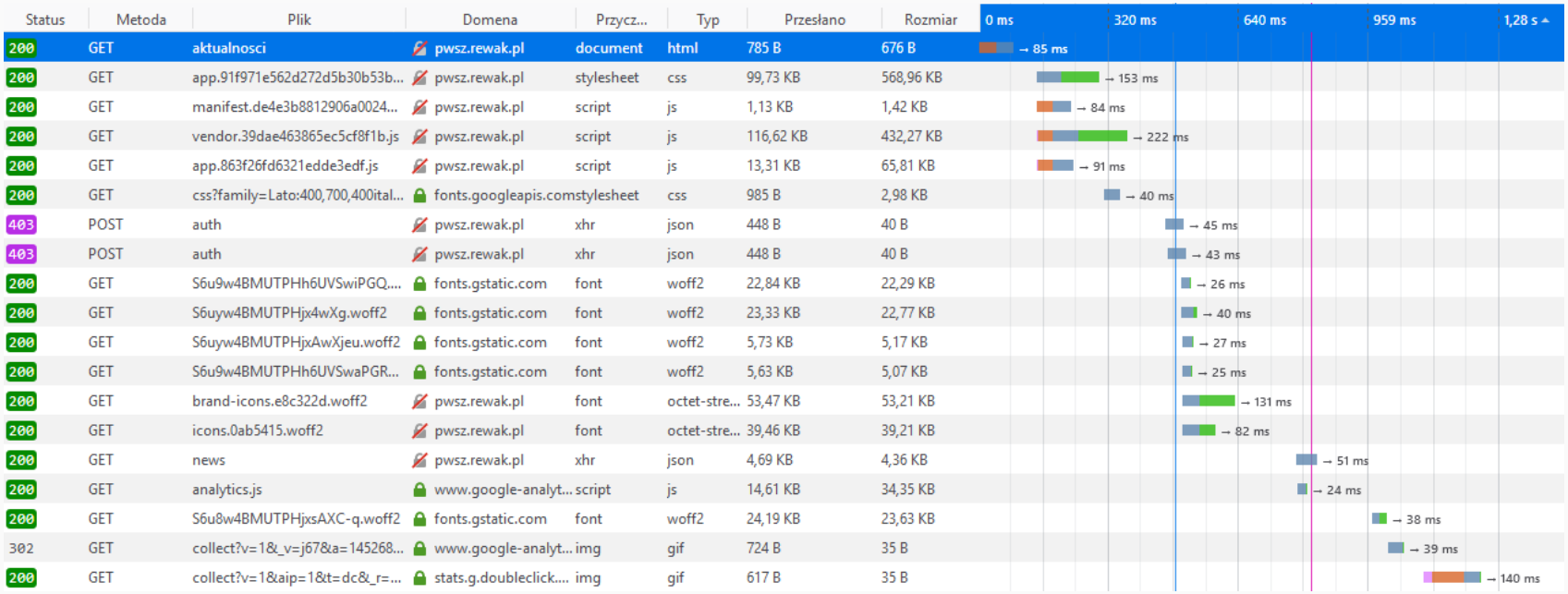
Okno na internet

Przeglądarka internetowa służy do łączenia się poprzez protokół HTTP(S) z serwerem i wyświetlania jego odpowiedzi. Umożliwia również interakcję z serwerem w sposób zaprojektowany przez programistę.

Okno na internet

Korzystając z narzędzi deweloperskich łatwo można podejrzeć ile zapytań wykonuje wejście na dowolną stronę w internecie.

Pod maską



Status	Metoda	Plik	Domena	Przycz...	Typ	Przesłano	Rozmiar	0 ms	320 ms	640 ms	959 ms	1,28 s
200	GET	aktualnosci	pwsz.rewak.pl	document	html	785 B	676 B	→ 85 ms				
200	GET	app.91f971e562d272d5b30b53b...	pwsz.rewak.pl	stylesheet	css	99,73 KB	568,96 KB	→ 153 ms				
200	GET	manifest.de4e3b8812906a0024...	pwsz.rewak.pl	script	js	1,13 KB	1,42 KB	→ 84 ms				
200	GET	vendor.39dae463865ec5cf8f1b.js	pwsz.rewak.pl	script	js	116,62 KB	432,27 KB	→ 222 ms				
200	GET	app.863f26fd6321edde3edf.js	pwsz.rewak.pl	script	js	13,31 KB	65,81 KB	→ 91 ms				
200	GET	css?family=Lato:400,700,400ital...	fonts.googleapis.com	stylesheet	css	985 B	2,98 KB	→ 40 ms				
403	POST	auth	pwsz.rewak.pl	xhr	json	448 B	40 B	→ 45 ms				
403	POST	auth	pwsz.rewak.pl	xhr	json	448 B	40 B	→ 43 ms				
200	GET	S6u9w4BMUTPHh6UVSwiPGQ....	fonts.gstatic.com	font	woff2	22,84 KB	22,29 KB	→ 26 ms				
200	GET	S6uyw4BMUTPHjx4wXg.woff2	fonts.gstatic.com	font	woff2	23,33 KB	22,77 KB	→ 40 ms				
200	GET	S6uyw4BMUTPHjxAwXjeu.woff2	fonts.gstatic.com	font	woff2	5,73 KB	5,17 KB	→ 27 ms				
200	GET	S6u9w4BMUTPHh6UVSwaPGR...	fonts.gstatic.com	font	woff2	5,63 KB	5,07 KB	→ 25 ms				
200	GET	brand-icons.e8c322d.woff2	pwsz.rewak.pl	font	octet-stre...	53,47 KB	53,21 KB	→ 131 ms				
200	GET	icons.0ab5415.woff2	pwsz.rewak.pl	font	octet-stre...	39,46 KB	39,21 KB	→ 82 ms				
200	GET	news	pwsz.rewak.pl	xhr	json	4,69 KB	4,36 KB	→ 51 ms				
200	GET	analytics.js	www.google-analyt...	script	js	14,61 KB	34,35 KB	→ 24 ms				
200	GET	S6u8w4BMUTPHjxsAXC-q.woff2	fonts.gstatic.com	font	woff2	24,19 KB	23,63 KB	→ 38 ms				
302	GET	collect?v=1&_v=j67&a=145268...	www.google-analyt...	img	gif	724 B	35 B	→ 39 ms				
200	GET	collect?v=1&aip=1&t=dc&_r=...	stats.g.doubleclick....	img	gif	617 B	35 B	→ 140 ms				

Pod maską

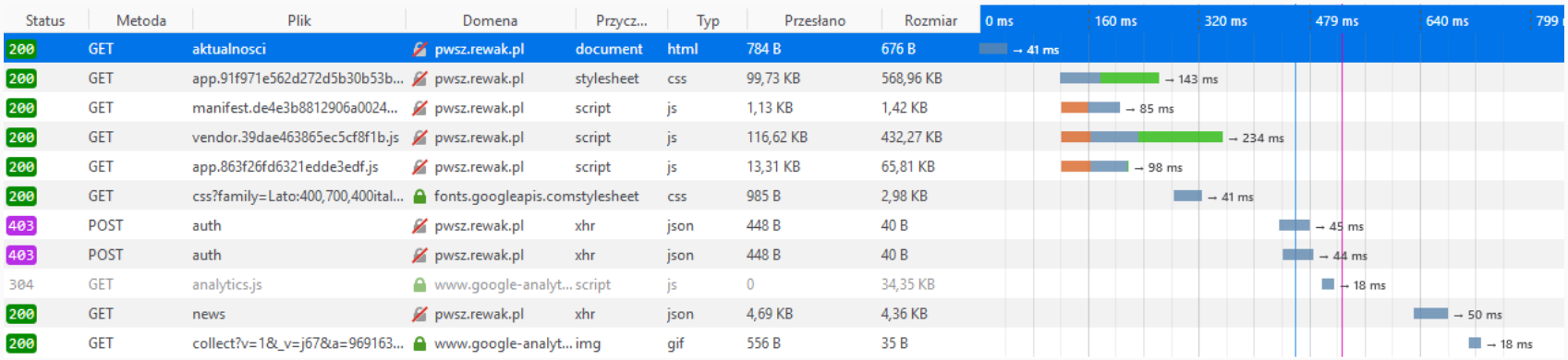
Kolejno wczytywane są:

- strona `/aktualnosci`, czyli HTML
- załączone w HTML-u skrypty JS i style CSS
- odpytanie o status uwierzytelnienia
- fonty, ikony
- odpytanie o listę aktualności
- zewnętrzy skrypt Google Analytics
- odpytania do Google Analytics

Pod maską

Po odświeżeniu strony (F5) widać, że lista się zmniejszyła:

Pod maską



Status	Metoda	Plik	Domena	Przycz...	Typ	Przesłano	Rozmiar	0 ms	160 ms	320 ms	479 ms	640 ms	799
200	GET	aktualnosci	pwsz.rewak.pl	document	html	784 B	676 B	→ 41 ms					
200	GET	app.91f971e562d272d5b30b53b...	pwsz.rewak.pl	stylesheet	css	99,73 KB	568,96 KB		→ 143 ms				
200	GET	manifest.de4e3b8812906a0024...	pwsz.rewak.pl	script	js	1,13 KB	1,42 KB		→ 85 ms				
200	GET	vendor.39dae463865ec5cf8f1b.js	pwsz.rewak.pl	script	js	116,62 KB	432,27 KB		→ 234 ms				
200	GET	app.863f26fd6321edde3edf.js	pwsz.rewak.pl	script	js	13,31 KB	65,81 KB		→ 98 ms				
200	GET	css?family=Lato:400,700,400ital...	fonts.googleapis.com	stylesheet	css	985 B	2,98 KB			→ 41 ms			
403	POST	auth	pwsz.rewak.pl	xhr	json	448 B	40 B				→ 45 ms		
403	POST	auth	pwsz.rewak.pl	xhr	json	448 B	40 B				→ 44 ms		
304	GET	analytics.js	www.google-analyt...	script	js	0	34,35 KB				→ 18 ms		
200	GET	news	pwsz.rewak.pl	xhr	json	4,69 KB	4,36 KB					→ 50 ms	
200	GET	collect?v=1&_v=j67&a=969163...	www.google-analyt...	img	gif	556 B	35 B						→ 18 ms

Pod maską

Kolejno wczytywane są tym razem:

- strona `/aktualnosci`, czyli HTML
- załączone w HTML-u skrypty JS i style CSS
- odpytanie o status uwierzytelnienia
- odpytanie o listę aktualności
- zewnętrzy skrypt Google Analytics
- odpytania do Google Analytics

Co się stało?

Przeglądarka jest w stanie zapamiętać jaka strona prosiła o jakie zasoby. Pewne pliki mogą zostać zapisane w pamięci komputera, aby w przyszłości nie było wymagane pobieranie ich na nowo z internetu.

Co cache'ować?

Do *cache* trafiają najczęściej arkusze stylów, skrypty, fonty i grafiki.

Nowy wspaniały świat

W idealnie skonstruowanym frontendzie zapamiętane zostałyby zatem również:

- wszystkie załączone w HTML-u skrypty JS i style CSS
- zewnętrzy skrypt Google Analytics

aby odpytania wyglądały następująco:

- strona `/aktualnosci`, czyli HTML
- odpytanie o status uwierzytelnienia
- odpytanie o listę aktualności
- odpytania do Google Analytics

Zawsze?

Czy wszystkie wcześniej wymienione zasoby powinny zawsze być umieszczane w pamięci podręcznej przeglądarki?

Można wykorzystać nagłówek `Cache-Control: no-cache` zapytania HTTP, aby ograniczyć *cache'owanie* zasobów.

Tips'n'tricks

Kombinacja klawiszy Ctrl+F5 zazwyczaj czyści pamięć podręczna przeglądarki na aktualnie wyświetlanej stronie.

Cache HTTP

Pośrednik HTTP

Pośredniczenie HTTP nie jest *de iuro* systemem pamięci podręcznej, ale działa w podobny sposób.

Pośrednik HTTP

Idea polega na rozmieszczeniu serwerów w taki sposób, aby informacje były przesyłane klientowi z najbliższego (również geograficznie!) serwera pośredniczącego.

CDN

Na takiej zasadzie działają CDN-y, (ang. *Content Delivery Network*), czyli rozproszone systemy dostarczania treści. Na takich serwerach przetrzymuje się przede wszystkim pliki CSS, JS i wszelakie pliki graficzne.

CDN

Przykładowy link do biblioteki jQuery hostowanej na Cloudflare Amazonu:

`https://cdnjs.cloudflare.com/ajax/libs/jquery/3.3.1/core.js`

Cache bazy danych

Cache danych

Pamięć podręczna może również zostać wykorzystana po stronie backendu, a najczęściej używana jest przy optymalizacji względem czasu pobierania danych.

Redis

Ilustrującym przykładem będzie aplikacja korzystająca bazowo z relacyjnej bazy danych MySQL, ale *cache'ująca* pewne wyniki za pomocą *key-value database* Redis.

Redis

Redis, w przeciwieństwie do MySQL-a, opiera się na zapisywaniu danych w pamięci komputera.

Dzięki temu dostęp do danych jest o wiele szybszy, ale ceną jest nietrwałość przechowywanych informacji.

Przykład

Wiemy, że:

- system to sklep internetowy, w którym sprzedawana jest bardzo duża liczba produktów;
- aby pobrać produkt z bazy danych należy połączyć kilka tabel z wieloma warunkami;
- zmiany w produktach zachodzą stosunkowo rzadko;
- na stonie głównej sklepu pojawia się n wybranych przez administratora produktów.

Przykład

Można oczywiście zbudować w kontrolerze `HomeController` skomplikowane zapytanie SQL lub długi łańcuch metod ORM-a, które odpytają bazę danych o ten konkretny zestaw danych. Generuje to przynajmniej dwa problemy:

- skomplikowane zapytanie będzie trwało dłuższy niż krótszy czas;
- zapytanie odpyta bazę danych przy każdym jednym wejściu każdego użytkownika.

Przykład

Innym rozwiązanem jest zbudowanie prostego systemu pamięci podręcznej, który będzie *cache'ował* interesujące nas produkty.

Przykład

`HomeController` nie wykona wówczas żadnego zapytania do bazy danych, a jedynie odpyta Redisa o kolekcję obiektów typu `CachedHomeProduct`.

Każdy produkt będzie zebranym uprzednio zestawem informacji istotnych tylko w kontekście strony głównej: nazwa, cena, link do grafiki promocyjnej.

Przykład

Kiedy Redis będzie uaktualniany?

Najlepiej przy każdej edycji produktu. Najsprytniej będzie podpiąć *listener* na akcję zapisu wybranego modelu i w tymże *listenerze* uruchomić proces *cache'owania*.

Przykład

Co się stanie jak serwer padnie i wstanie po chwili?

Dane w pamięci podręcznej mogą wówczas zniknąć. Warto dla takich przypadków stworzyć mechanizm, który sprawdzi, że *cache* ma jakiekolwiek dane; jeżeli nie, wówczas można wykonać polecenie ponownego uzupełnienia systemu pamięci podręcznej lub pobrania danych bezpośrednio z bazy danych.

Inne przykłady

Gdzie jeszcze można wykorzystać *cache'owanie* danych?

Wszędzie, gdzie będzie miało to sensowne uzasadnienie. Sztandarowymi przykładami są wyszukiwarki, ale zda się doskonale do zbierania w konkretne zestawy rzadko aktualizowane modele bazodanowe.

Podsumowanie

Bibliografia i ciekawe źródła

- https://developer.mozilla.org/pl/docs/Web/HTTP/Headers/Cache-Control
- https://cdnjs.com/libraries/jquery

Pytania?

Kod prezentacji dostępny jest w repozytorium git pod adresem
`https://bitbucket.org/krewak/pwsz-ppsi`

Wszystkie informacje dot. kursu dostępne są pod adresem
`http://pwsz.rewak.pl/kursy/4`