Teoria egzamin Damian Strojek

cookies

HTTP cookies to małe bloki danych stworzone przez przeglądarke podczas przeglądania przez użytkownika danej strony internetowej. Ciastka znajdują się na urządzeniu, które jest używane, aby dostać się do danej strony internetowej i więcej niż jedno ciastko może być umieszczone na urządzeniu w trakcie sesji.

Dzięki ciastkom serwery są w stanie przechowywać informacje stanowe (takie jak przedmioty dodane do koszyka w sklepie internetowym) na urządzeniu użytkownika lub śledzenie aktywności przeglądania użytkownika. Mogą być również używane do zapisywania informacji do późniejszego wykorzystania, które użytkownik wprowadził wcześniej w polach np. formularza.

Uwierzytelniające pliki cookie są powszechnie używane przez serwery internetowe do uwierzytelniania, czy użytkownik jest zalogowany i na jakim koncie. Bez pliku cookie użytkownicy musieliby się uwierzytelnić, logując się na każdej stronie zawierającej poufne informacje, do których chcą uzyskać dostęp.

Pliki cookie dzielimy na :

- **Sesyjny plik cookie** - jest to tymczasowy plik cookie istniejący tylko w pamięci tymczasowej, gdy użytkownik przegląda witrynę internetową. Wygasają one lub są usuwane po zamknięciu przeglądarki internetowej. Nie posiadają one daty wygaśnięcia.
- **Trwały plik cookie** - wygasa w określonym dniu lub po upływie określonego czasu. Przez okres istnienia trwałego pliku cookie ustawionego przez jego twórce, jego informacje będą przesyłane do serwera za każdym razem, gdy użytkownik odwiedza witrynę, do której należy, lub za każdym razem, gdy użytkownik przegląda zasób należący do tej witryny.

localStorage, sessionStorage

Obiekty web storage takie jak localStorage i sessionStorage pozwalają na zapisanie klucza/wartości w przeglądarce. Interesujące jest to, że dane te przetrwają mimo odświeżenia strony (sessionStorage), a nawet zresetowania całej przeglądarki (localStorage).

To co odróżnia te dwa obiekty od ciasteczek jest to, że nie są one przesyłane z każdym żądaniem do serwera. Dzięki temu możemy przechowywać znacznie więcej informacji w tych obiektach. Serwer również nie może manipulować tymi obiektami prze nagłówki HTTP. Wszystko jest zrobione przez JavaScript. Dodatkowo obiekty te są przypisane do swoich oryginalnych domen. Przez to inne protokoły czy poddomeny nie mogą dostać informacji z obiektów przypisanych do innych domen.

Wszystkie informacje, które są przechowywane w localStorage i sessionStorage są konwertowane do postaci stringów.

Główne **właściwości localStorage** :

- Jest udostępniana pomiędzy wszystkie zakładki i okna, które pochodzą z tego samego źródła.
- Informacje nie przedawniają się. Zostają nawet po zresetowaniu przeglądarki czy uruchomieniu ponownie systemu operacyjnego.

Główne **właściwości sessionStorage** :

- sessionStorage egzystuje tylko i wyłącznie w ramach konkretnej zakładki w przeglądarce. Kolejna zakładka z tą samą stroną będzie miała swoją osobną sessionStorage.
- Dane przetrwają reset strony, ale nie zamknięcie zakładki.

Protokoły internetowe

URL (Uniform Resource Locator) to ustandaryzowany adres zasobu (dokumentu, obrazu, pliku) w sieci, np. eti.pg.edu.pl.

URN (Uniform Resource Name) to tylko nazwa zasobu.

Na **URI (Uniform Resource Identifier)** składa się URN i/lub URL. URL względny np. `/login` pomija cały początek złożony z protokołu, adresu zasobu itd.

DNS (Domain Name System) odpowiada za translację nazw zasobu na adresy IP w sieci komputerowej.

HTTP (Hypertext Transfer Protocol) to protokół komunikacyjny serwujący usługi na domyślnym porcie nr 80. Serwer WWW to oprogramowanie, które działa po uruchomieniu systemu i oczekuje na żądania klienta na określonym porcie w ramach, których odsyła

potrzebne dokumenty z zasobów lokalnych. Protokół HTTP jest bezstanowy i nie utrzymuje stanu konwersacji z klientem pomiędzy żądaniami. Nie rozróżnia ich sam z siebie.

Budowa żądania i odpowiedzi :

```
Komenda/Status + Nagłówek + pusta linia + ciało
```

W żądaniu i odpowiedziach zawsze musi być określony `Host`. Większość serwerów ma limit długości GET 1024-4096, POST nie ma takich ograniczeń.

Klasy odpowiedzi :

- 1xx - klasa informacyjna
- 2xx - klasa odpowiedzi pozytywnych
- 3xx - klasa przekierowania
- 4xx - klasa błędów klienta
- 5xx - klasa błędów serwera

Charakterystka HTTP w wersji 1.1

- Wprowadzono mechanizm wirtualnych hostów. Na jednej maszynie może się znajdować wiele hostów serwujących, np. różne strony internetowe.
- Trwałe połączenia - zamiast `connection: close` jest `connection: keep-alive`.
- Potokowość z wykorzystaniem trwałych. Jesteśmy w stanie wyprowadzić wiele żądań nie oczekując na odpowiedź.
- Dopracowano mechanizm pamięci podręcznej.
- Zniesiono limit 2 równoczesnych połączeń trwałych od tego samego klienta.
- Można przesyłać fragmenty zasobów / zasoby o nieznanym rozmiarze.

Charakterystyka HTTP w wersji 2.0

- Kompresja wybranych elementów żądania i odpowiedzi.
- Wprowadzono format binarny, a nie tekstowy i zaczęto wykorzystać efektywnie protokół TCP.
- Występuje pełna priorytetyzacja i multipleksacja.
- Serwer jest w stanie wysyłać dodatkowe zasoby za pomocą `server push`.

Przekierowanie żądania można zrealizować za pomocą zmian w pliku konfiguracyjnym serwera włączając rewriteEngine i ustawiając odpowiedni rewriteRule łącznie z kodem przekierowań. Można również dokonać redirect w `.htaaccess`.

Strona klienta

Krokowy sposób działania przeglądarki :

- Pobierany dokument HTML na progu przeglądarki (optymalizacja, naprawa),
- dociągane zasoby za pomocą żądań przewidujących,
- HTML parsowany :
  - Tworzona reprezentacja obiektowa (drzewo DOM) wczytywanego dokumentu,
  - skrypty osadzone w kodzie HTML wykonywane po napotkaniu `<script>` w trakcie interpretacji modyfikują drzewo DOM,
  - zakończone parsowanie HTML → gotowe drzewo DOM co skutkuje zdarzeniem `DOMContentLoaded`.
- dokument CSS jest pobierany i składany (kaskadowanie, dziedziczenie),
- nałożenie na elementy drzewa DOM wyznaczonych styli,
- drzewo obiektów wyświetlanych (render tree) :
  - określenie pozycji i wielkości elementów drzewa obiektów wyświetlanych (layout),
  - wyrysowanie w oknie przeglądarki (painting),
- wyświetlenie obiektów osadzonych np. plików graficznych,
- zdarzenie `load` zakończone (czy tam egzekwuje się w tym momencie, aby zakomunikować, że skończyło się ładować).

MIME Type (Media Type) mówi o tym jakiego typu plików się spodziewamy / odsyła nam przeglądarka - odczytujemy to z sygnatury pliku.

Negocjowanie zawartości (Content Negotiation) od strony przeglądarki polega na wysłaniu żądania z polem `Accept` w nagłówku i zdefiniowaniem listy preferowanych przez przeglądarke `content-type` dokumentów. Serwer w swoich żądaniach wysyła pole `content-type`. `q` w żądaniach przeglądarki określa co przeglądarka woli bardziej (taki priorytet). Domyślnie `q = 1`.

Window to obiekt hosta dla kodów JavaScript wykonywanych w przeglądarce. Udostępnia środowisko i nie jest częścią ECMAScript (standard języka skryptowego, który ma gwarantować kompatybilność między przeglądarkami i językami jscript oraz javascript), dostępne metody są zależne od implementacji w danej przeglądarce.

Kod JavaScript wykonuje się w chwili napotkania `<script>` albo gdy wykona się jakieś zdarzenie, np. naciśnięcie guzika (`onclick`). Jest jeszcze możliwość załączenia pliku z rozszerzeniem `.js`, który wykona się razem z parsowaniem. Gdy damy coś jako `onload` to wykona się to dopiero w przypadku zdarzenia `load`.

JSON (JavaScript Object Notation) to lekki tekstowy format wymiany danych niezależny od platformy.

AJAX (Asynchronous JavaScript and XML) odpowiada za asynchroniczne pobieranie danych z serwera już po załadowaniu strony do przeglądarki. Zamiast przeładowywać całą stronę, pobierany jest jedynie jej fragment, który zostaje wyświetlony w przeglądarce przy użyciu API DOM.

DOM (Document Object Model) to obiektowy model dokumentu - konwencja reprezentacji dokumentu HTML/XML. API DOM to zestaw interfejsów do operowania na drzewie dokumentu. Drzewo DOM nie może być zmieniane przez CSSa.

Strona serwera

MongoDB to otwarty, nierelacyjny system zarządzania bazą danych napisany w języku cpp. Charakteryzuje się brakiem ściśle zdefiniowanej struktury obsługiwanych baz danych. Zamiast tego dane składowane są jako dokumenty w stylu JSON. Narzędzie Composer automatyzuje zarządzanie zależnościami projektu. Buduje ono drzewo zależności projektu, a następnie pobiera wszystki wymagane biblioteki i zostaje zapisany tzw. *lockfile* (`composer.lock`) z wersjami wszystkich pobranych bibliotek itd.

Dokumenty tej samej kategorii są przetrzymywane razem tworząc kolekcję dokumentów. Kilka kolekcji dokumentów z kolei składa się na jedną bazę danych.

Kolekcja w MongoDB zostaje automatycznie zainicjowana przy pierwszym użyciu i może zawierać dokumenty rożniące się strukturą. Jedna konkretna kolekcja nie może należeć do

wielu baz danych, ale nic nie broni przed tym, aby stworzyć drugą taką samą kolekcję w innej bazie danych.

Jeśli chodzi o regex dla MongoDB to przy sortowaniu `1` jest od sortowania rosnąco, a `-1` od sortowania malejąco.

Sesja to ciąg kolejnych zapytań HTTP wysyłanych przez tego samego klienta do serwera. **Stan sesji** to dane przechowywane pomiędzy zapytaniami, które składają się na sesję. Każde żądanie posiada unikalny identyfikator przekazany w momencie otwarcia sesji. Przeglądarka dołącza go automatycznie.

Przekazywanie identyfikatora sesji może się odbywać w sposób niebezpieczny np. za pomocą GETa (zapisywany w linkach, bookmarksach, itd.) lub za pomocą POSTa (w ukrytym polu formularza) i ciasteczek.

```
session.use_cookies=1      // używane cookie do przekazywania id sesji
session.use_trans_sid=1    // używanie parametrów GET/POST
session.use_only_cookies=0 // używanie wyłącznie ciasteczek
```

`session_start` odpowiada za rozpoczęcie nowej sesji lub wczytanie już stworzonej. Musi znajdować się on przed pierwszym odwołaniem do `$_SESSION`. `session_destroy` nie usuwa ciasteczek, ani identyfikatora. Po zniszczeniu sesji dalej można zapisywać dane dla identyfikatora tej sesji. Całkowite zniszczenie sesji wymaga ręcznego usunięcia ciasteczka/ciasteczek.

Lokalizacja danych sesji :

- W pamięci operacyjnej - szybki dostęp, ale ograniczone rozmiar i bardzo kłopotliwe skalowanie w poziomie,
- w plikach na dysku - dowolny rozmiar, ale wolniejszy dostęp,
- w bazie danych - dowolny rozmiar i łatwe skalowanie w poziomie.

Na **parametry konfiguracyjne mechanizmu sesji** w pliku `php.ini` składają się :

- `session.name` - nazwa ciasteczka/parametru z identyfikatorem sesji (domyślnie PHPSESSID),
- `session.cookie_lifetime` - czas trwania sesji (domyślnie do zamknięcia przeglądarki),
- `session.save_handler` - lokalizacja danych sesji (domyślnie w plikach),
- `sesssion.use_strict_mode` - akceptowanie identyfikatora sesji z żądania HTTP dla niezainicjowanej sesji (domyślnie jest wyłączone, czyli akceptuje).

Kontrola dostępu czyli inaczej uwierzytelnianie to weryfikacja tożsamości, a **autoryzacja** to nadanie uprawnień dostępu.

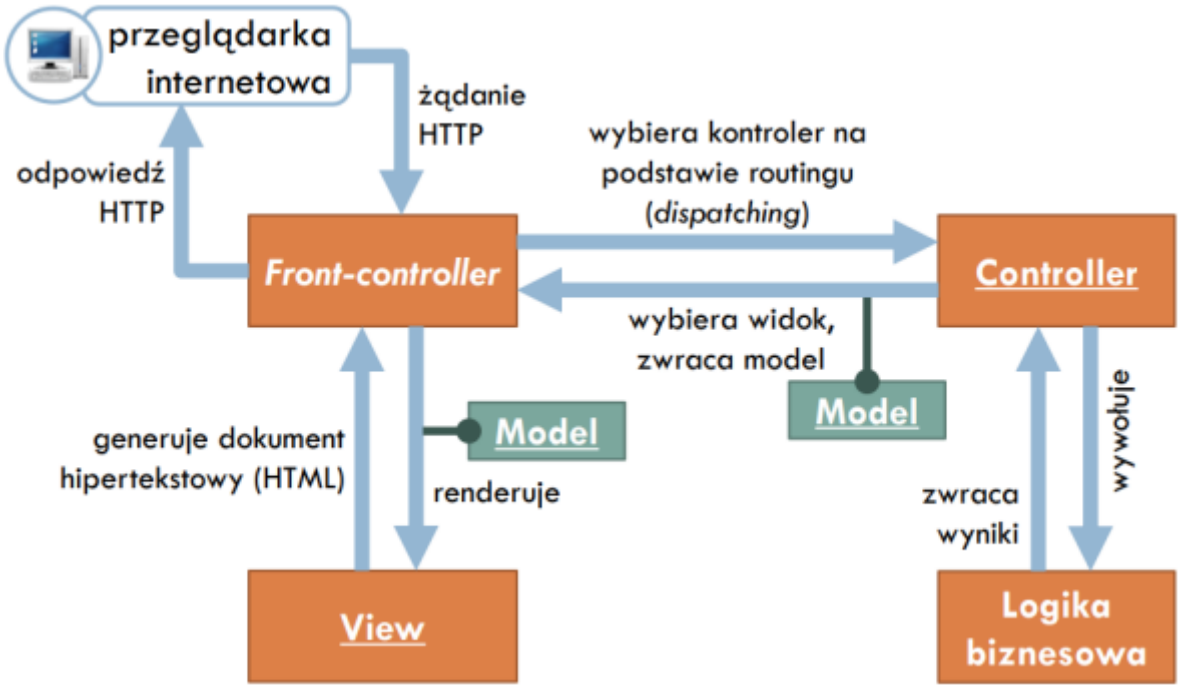
- Funkcja mieszająca/hashująca generuje skrót hasła,
- Kolizja - funkcja mieszająca generuje ten sam hash dla różnych haseł (teoria hashowania).

MVC

Tzw. **Model View Controller,** który kontroluje przepływ sterowania w celu obsługi akcji użytkownika. Front wybiera odpowiedni kontroler i realizuje globalne aspekty (np. sesja). Routing to odwzorowanie adresu na odpowiedni kontroler, który ma obsłużyć dany adres. Model zawiera dane, które kontroler przeznaczył do wyświetlenia w widoku i może zawierać to co z widoku leci do kontrolera, np. przez form.

Mocne powiązanie z architekturą trójwarstwową : prezentacja, biznes i dane.

Kolejność operacji : ***Wywołanie front-controllera → Routing/Dispatching → Wywołanie kontrolera → Uruchomienie logiki biznesowej → Propagacja danych modelu → Wyświetlenie widoku***



JavaScript uzupełnienie

Funkcja Callback to funkcja, która została przekazana do innej funkcji jako argument po to, aby zostać wywołana w środku tej funkcji w odpowiednim momencie. W życiu callbacki są najczęściej używane z funkcjami asynchronicznymi.