PROGRAMIŚCI DLA SZCZECINA

Zachodniopomorska Szkoła Biznesu w Szczecinie

Technologie internetowe

Protokół HTTP

REST API

Mariusz Łazor / 02.09.2019

Fundusze Europejskie Program Regionalny

Rzeczpospolita Polska

Pomorze Zachodnie

Unia Europejska Europejski Fundusz Społeczny

Charakterystyka protokołu HTTP

- HTTP- *Hypertext Transfer Protocol*
- Protokół to *„zbiór zasad wymiany informacji i współpracy programów i urządzeń komputerowych"*
- Protokół http działa w najwyższej warstwie modelu ISO OSI - warstwie aplikacji
- W HTTP programami są klienci (wysyłają żądania, np. przeglądarki internetowe) i serwery (zwracają odpowiedzi)
- Każde żądanie powiązane jest z chęcią otrzymania konkretnego zasobu, np. strony HTML, obrazka, pliku JavaScript, CSS itd.
- Każdy zasób posiada unikalny identyfikator - URI (*Uniform Resource Identifier*)
- W HTTP komunikacja oparta jest na żądaniach i odpowiedziach, protokół HTTP określa format tych wiadomości (co powinny one zawierać)
- Protokół HTTP jest bezstanowy - oznacza to, że serwer nie pamięta wcześniejszych żądań, każde zapytanie jest interpretowane niezależnie od innych

Adres zasobu - URL

- Adres, pod którym dostępny jest zasób nazywany jest URL (*Uniform Resource Locator*) i wchodzi on w skład zbioru *URI*, w którym znajduje się również *URN* (*Uniform Resource Name*)
- Format adres URL: *scheme:[//[user[:password]@]host[:port]][/path][?query][#fragment]*, np.: *http://user:pass@www.test.com:80/aa/bb/cc?test=1#identyfikator*
- *scheme:* służy do określenia protokołu, najczęściej *http* lub *https* (rozszerzenie *http* o szyfrowanie połączenia między klientem a serwerem)
- *user:password:* dane do uwierzytelnienia, przekazywanie danych w adresie nie jest bezpieczne nawet używając protokołu https, raczej nie stosuje się tej metody
- *host:* nazwa domeny lub adres IP
- *port:* numer portu, na jakim nasłuchuje serwer, domyślnym portem dla *http* jest *80*, dla *https* - *443*, aczkolwiek możemy skonfigurować nasz serwer, aby nasłuchiwał na konkretnym porcie
- *path:* ścieżka do zasobu
- *query:* dodatkowe dane identyfikujące zasób, które oddziela się od ścieżki znakiem *?*, parametry łączy się znakami *&*
- *fragment:* przekierowanie do konkretnego fragmentu zasobu, np. elementu na stronie posiadającego konkretny identyfikator

Żądanie i odpowiedź HTTP

- Żądania są wysyłane do serwera w postaci wiadomości, które składa się z kilku linii:
  - linia zawierająca metodę HTTP, adres do zasobu i wersję protokołu
  - linie zawierające listę nagłówków
  - pusta linia oznaczająca koniec nagłówków
  - ciało wiadomości (jeżeli istnieje)
- Odpowiedź, podobnie jak żądanie, składa się z kilku linii:
  - linia zawierająca wersję protokołu i status odpowiedzi
  - linie zawierające listę nagłówków
  - pusta linia oznaczająca koniec nagłówków
  - ciało wiadomości (jeżeli istnieje)

Metody HTTP

- *GET* - metoda służąca do pobrania zasobu, nie posiada ciała, wszelkie dodatkowe dane identyfikujące zasób przekazywane są w adresie *URL*
- *HEAD* - metoda podobna do *GET*, z tą różnicą, że nie zwraca ona ciała odpowiedzi, tylko status i listę nagłówków - wykorzystywane np. do sprawdzenia, czy dany zasób uległ zmianie
- *POST* - metoda służąca do przesyłania zawartości formularzy, tworzenia nowego zasobu, dodawania danych do istniejącego zasobu - może zawierać ciało, ale nie jest ono obowiązkowe
- *PUT* - metoda podobna do *POST*, która służy do aktualizacji ***całego*** zasobu
- *PATCH* - metoda podobna do *PUT*, która służy do aktualizacji ***części*** zasobu
- *DELETE* - metoda służąca do usunięcie zasobu
- *CONNECT* - metoda służąca do nawiązania połączenia między klientem a serwerem
- *OPTIONS* - metoda służąca do pobrania informacji na temat możliwości komunikacji dla danego zasobu, np. jakie żądania są obsługiwane przez serwer
- *TRACE* - żądanie służące do testowania przetwarzania żądania przez serwer - w odpowiedzi powinniśmy otrzymać zapytanie, które otrzymał

Nagłówki HTTP

- Nagłówki służą do przekazywania metadanych na temat zasobów
- Nagłówki mają one postać *name: value*
- Nagłówki są dołączane zarówno do żądania, jak i odpowiedzi
- Istnieje wiele nagłówków wchodzących domyślnie w skład HTTP, ale w aplikacji możliwa jest również implementacja obsługi „customowych", których nazwę zazwyczaj zaczyna się od *X-*, w celu odróżnienia och od tych standardowych

Przykładowe nagłówki HTTP

- Accept - Klient informuje serwer o tym jaki format jest w stanie zrozumieć, może to być na przykład JSON: application/json
- Accept-Encoding - Klient informuje serwer o tym jakie sposoby kodowania ciała wiadomości rozumie, może być użyty do określenia pożądanego algorytmu kompresji odpowiedzi
- Access-Control-Allow-Methods - W odpowiedzi na zapytanie typu OPTIONS serwer informuje jakie inne czasowniki HTTP są dozwolone
- Access-Control-Allow-Origin - Serwer informuje klienta jakie domeny uprawnione są do użycia odpowiedzi
- Cache-Control - Nagłówek służący do zarządzania cache'owaniem. Dotyczy zarówno żądań jak i odpowiedzi
- Connection - Zawiera informacje na temat połączenia pomiędzy klientem a serwerem
- Content-Encoding - Serwer informuje klienta o sposobie kodowania ciała wiadomości
- Content-Type - Odpowiednik nagłówka Accept wysyłany przez serwer informujący o formacie odpowiedzi
- Cookie - Nagłówek służący do przesłania ciasteczka przez klienty do serwera
- Date - Zawiera datę mówiącą o czasie wygenerowania żądania/odwiedzi
- Host - Zawiera domenę, do której wysyłane jest żądanie
- Location - Zawiera informacje o położeniu zasobu, może być użyty na przykład przy przekierowaniach i tworzeniu nowych zasobów
- Server - Serwer informuje klienty jakiego oprogramowania używa do obsługi odpowiedzi
- Set-Cookie - Nagłówek służący do ustawienia ciasteczka
- User-Agent - Nagłówek dołączany do zapytania informujący o tym jaki klient został użyty do jego wysłania

Ciasteczka (Cookie)

- Ciasteczka to tak naprawdę nagłówek umieszczany w żądaniu, jego wartość jest przechowywana w pamięci przeglądarki w postaci pliku
- Ciasteczka są przypisywane do konkretnej domeny(*host*) oraz ścieżki(*path*)
- HTTP jest protokołem bezstanowym - dzięki ciasteczkom jesteśmy w stanie połączyć szereg żądań w sesję - serwer ustawia dane ciasteczko, które jest dołączane do kolejnego żądania

Statusy odpowiedzi HTTP

- 1xx - informują, że zapytanie zostało otrzymane i jest przetwarzane - rzadko spotyka się ich praktyczne wykorzystanie
- 2xx - informują o prawidłowym przetworzeniu zapytania:
  - 200 OK - żądanie zostało przetworzone prawidłowo
  - 201 Created - zasób został utworzony
  - 202 Accepted - zapytanie zostało przyjęte przez serwer, jego przetwarzanie nie jest jeszcze ukończone
  - 204 No Content - zapytanie zostało przetworzone, ciało odpowiedzi jest puste
- 3xx - informują o konieczności podjęcia dodatkowej akcji w celu skończenia zapytania, służą do ustawiania przekierowań:
  - 301 Moved Permanently - informuje, że zasób został przeniesiony na stałe w inne miejsce
- 4xx - informują o błędzie klienta, że żądanie nie może być przetworzone:
  - 400 Bad Request - nieprawidłowa zawartość żądania
  - 401 Unauthorized - problem w trakcie uwierzytelnienia, np. nieprawidłowe hasło
  - 403 Forbidden - brak dostępu do zasobu
  - 404 Not Found - zasób nie istnieje
- 5xx - informują o błędzie serwera:
  - 500 Internal Server Error - serwer znalazł się w stanie, który uniemożliwia poprawne przetworzenie zapytania
  - 502 Bad Gateway - jeżeli żądanie jest przetwarzane przez wiele węzłów (serwerów pośredniczących) i któryś z nich otrzymał błędną odpowiedź, to zwrócony zostanie kod *502*
  - 503 Service Unavailable - serwer niedostępny

HTTPS

- HTTPS - *Hypertext Transfer Protocol Secure*
- https jest szyfrowaną wersją protokołu http - oznacza to, że osoba, która „wpięła się" w naszą sieć będzie widziała wszystkie przesyłane dane w postaci zaszyfrowanej
- Dane są szyfrowane na poziomie protokołu TSL (*Transport Layer Security*), który jest rozwinięciem SSL *(Secure Socket Layer)* - protokół TSL działa jedną warstwę niżej niż http - na warstwie prezentacji, dzięki czemu może on zabezpieczać protokoły na wyższym poziomie
- Wszelkie aplikacji wymagające naszego uwierzytelnienia powinny wykorzystywać protokół https, aby uchronić użytkowników przed wykradzeniem ich danych (aczkolwiek nie daje to 100% bezpieczeństwa)

REST API

- REST - *Representational State Transfer* - zbiór reguł określających sposób tworzenia architektury systemu informatycznego (tworzenie i pobieranie zasobów)
- API - *Application Programming Interface* - zbiór reguł opisujących sposób komunikacji między programami komputerowymi
- *API* reguluje w jaki sposób użytkownik może uzyskać dostęp do zasobów, natomiast *REST* określa sposób, w jaki te API powinno być zbudowane

REST API - zasady

- Odseparowanie warstwy klienta (front-end) od warstwy serwerowej (back-end) - klient jedynie wysyła wiadomość do serwera (nie ma wpływu na to się dzieje po jego stronie), natomiast serwer jedynie zwraca zbiór danych, ale w żaden sposób nie wymusza na kliencie tego, co ma zostać wyświetlone
- RESTful API jest bezstanowe - każde żądanie jest realizowane oddzielnie
- Odpowiedź API powinna definiować, czy ma zostać ona zacache'owana czy nie (informacja ta jest umieszczona w nagłówkach odpowiedzi)
- Endpoint (adres zasobu) powinien jednoznacznie wskazywać, do jakiego zasobu się odwołujemy

JSON

- JSON (*JavaScript Object Notation*) - sposób przekazywania danych w składni zgodnej z językiem *JavaScript*
- Format JSON jest najczęstszym sposobem przekazywania i przyjmowania danych przez RESTful API
- Dane są przekazywane w formie "key": value, gdzie nazwa klucza koniecznie musi być przekazana w cudzysłowie, dozwolona jest obecność spacji w kluczu, ale nie jest to wskazane (warto zwracać na to uwagę w testach), ponieważ zmniejsza to użyteczność/dostępność API
- Kolejne dane są oddzielane przecinkiem
- W JSON możemy przekazać następujące typy danych:
  - String - musi być umieszczony w cudzysłowie, np. "name": "Jan"
  - Number - np. "age": 21
  - Boolean - np. "married": true
  - Null - np. "secondName": null
  - Obiekt - umieszcza się go w nawiasach klamrowych, np.:

```
{
    "name": "Jan",
    "age": 21,
    "married": true,
    "secondName": null
}
```

  - Tablica, dane umieszczone w nawiasach klamrowych, np.: {"knownLanguages": ["Polish", "English"]}

Polecane materiały

- Nagłówki HTTP: https://pl.wikipedia.org/wiki/Lista_nag%C5%82%C3%B3wk%C3%B3w_HTTP
- Działanie SSL: https://hostovita.pl/blog/jak-dziala-certyfikat-ssl/
- Wstęp do REST API: https://devszczepaniak.pl/wstep-do-rest-api/
- JSON: http://jsdn.pl/json-dla-poczatkujacych-i-kompletnie-poczatkujacych/