

Wstęp.

Poznanie narzędzi do obserwacji działania protokołu HTTP.

Komenda curl

Jedną z możliwości obserwacji danych przesyłanych w protokole HTTP jest komenda `curl`. Pozwala na wysyłanie dowolnych żądań (domyślnie rodzaju GET, ale parametrami można to zmienić). Celem poniższych zadań jest poznanie tego konsolowego narzędzia.

Przykład uruchamiania celem sprawdzenia opcji:

```
Wiersz polecenia

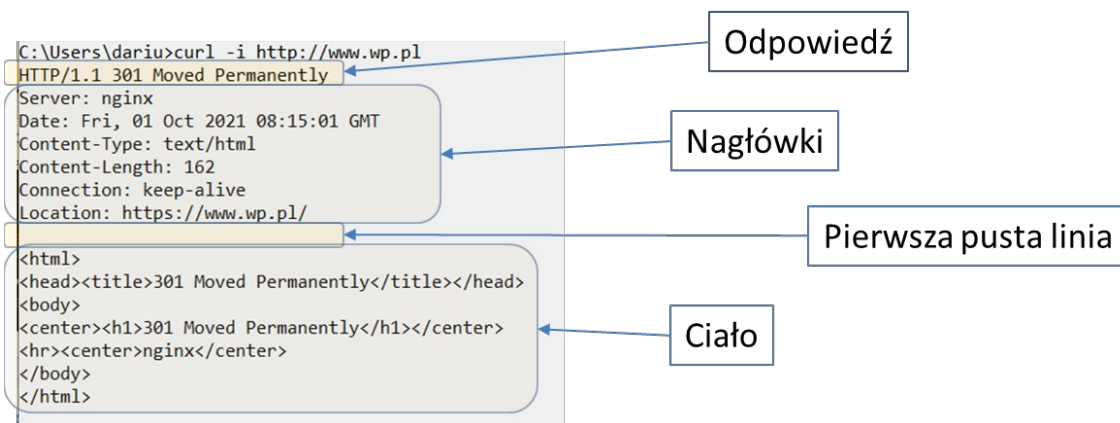
C:\Users\dariu>curl
curl: try 'curl --help' for more information

C:\Users\dariu>curl -h
Usage: curl [options...] <url>
  --abstract-unix-socket <path> Connect via abstract Unix domain socket
  --anyauth             Pick any authentication method
  -a, --append          Append to target file when uploading
  --basic               Use HTTP Basic Authentication
  --cacert <CA certificate> CA certificate to verify peer against
  --capath <dir>        CA directory to verify peer against
  -E, --cert <certificate[:password]> Client certificate file and password
  --cert-status         Verify the status of the server certificate
  --cert-type <type>    Certificate file type (DER/PEM/ENG)
  --ciphers <list of ciphers> SSL ciphers to use
  --compressed         Request compressed response
  -K, --config <file>   Read config from a file
  --connect-timeout <seconds> Maximum time allowed for connection
  --connect-to <HOST1:PORT1:HOST2:PORT2> Connect to host
  -C, --continue-at <offset> Resume transfer offset
```

Uruchomienie z wybranym adresem URL (`http://www.wp.pl`) i otrzymana odpowiedź:

```
C:\Users\dariu>curl http://www.wp.pl
<html>
<head><title>301 Moved Permanently</title></head>
<body>
<center><h1>301 Moved Permanently</h1></center>
<hr><center>nginx</center>
</body>
</html>
```

W wielu przypadkach warto użyć opcji `-i` (Include protocol response headers in the output) pokazującej nagłówki odpowiedzi, który może zawierać istotne informacje. Przykład użycia:



Można z niego wyczytać, że strona została na stałe przeniesiona po inny adres (<https://www.wp.pl>). Zatem po użyciu nowego adresu otrzymujemy dostęp do strumienia dla właściwej strony:

Status	Method	Domain	File	Initiator	Type	Transferred	Size	
200	GET	www.wp.pl	/	BrowserTabChild:jem:93 (document)	html	service worker	455.59 KB	
200	GET	ads.googleadsyndication.com	dc_oe=ChMy7OjOy8wIVF9j...	https://www.wp.pl/	gif	713 B	42 B	0 ms
200	GET	pagead2.googleadsyndication.com	activeviewhai=AKAQjvduh3Q...	https://www.wp.pl/	gif	725 B	42 B	80 ms
200	POST	www.wp.pl	aZhm80TMTC9eUdRmSeuK9...	MDaZOTeYfUgGDnagjMGNAb7c...				NS_BINDING_ABORTED
200	GET	pagead2.googleadsyndication.com	activeviewhai=AKAQjvduh3Q...	https://www.wp.pl/	gif	725 B	42 B	32 ms
200	GET	www.wp.pl	b3YyMGpnTVI4Gz0XvAbHTD...	script	js	1.75 KB	2.19 KB	24 ms
200	GET	www.wp.pl	cDnaDNSUCFD8d3YU58GQZU...	script	js	20.73 KB	51.20 KB	28 ms
200	GET	www.wp.pl	eDyamQ25UNGid8fHME5DQz6...	script	js	3.75 KB	7.17 KB	60 ms
200	GET	www.wp.pl	NhZnOxwYsK7DDraQsF3HhQ...	stylesheet	css	3.05 KB	9.45 KB	24 ms
304	GET	www.wp.pl	MTivOdc2YUuGz0T5wGid65...	script	js	cached	305.65 KB	12 ms

Wybierając zakładkę Network (sieć) otrzymujemy tabelarycznie przedstawioną komunikację przeglądarki z serwerem, również tą spowodowaną przekierowaniem strony, ściąganiem obrazków występujących na stronie, skryptów języka JavaScript itd. Gdy wybierzemy podgląd jednego z takich żądań, np. <https://www.wp.pl> otrzymamy widok przedstawiający dane, które mogliśmy odczytać w narzędziu konsolowym, przedstawione w bardziej przejrzystej formie (trochę zależnej od przeglądarki):

The screenshot shows the Chrome DevTools Network tab. A request to <https://www.wp.pl/> is selected. The response is expanded, showing the following details:

- Status:** 200 OK
- Version:** HTTP/1.1
- Transferred:** 455.59 KB (455.59 KB size)
- Response Headers (1.394 KB):**
 - HTTP/1.1 200 OK
 - server: nginx
 - date: Fri, 01 Oct 2021 08:57:41 GMT
 - content-type: text/html; charset=utf-8
 - vary: Accept-Encoding
 - content-security-policy: report-uri /v1/csplog; block-all-mixed-content
 - x-dns-prefetch-control: on
 - x-frame-options: DENY
 - strict-transport-security: max-age=0; includeSubDomains
 - x-download-options: noopen
 - x-content-type-options: nosniff
 - referrer-policy: no-referrer-when-downgrade
 - x-xss-protection: 1; mode=block
 - cache-control: private, no-store, no-cache
 - link: <https://www.wp.pl/b3YyMGpnTVI4Gz0XvAbHTD...

Arrows in the image point to the following elements:

- Request:** Points to the request entry in the Network tab.
- Response:** Points to the response details in the expanded view.
- Headers (raw):** Points to the raw header text in the expanded view.
- Headers (interpreter and sorted):** Points to the interpreted and sorted header list in the expanded view.

Linia odpowiedzi jest rozbita na składowe, nagłówki można widzieć w postaci przetworzonej, co ułatwia ich analizę (lub nieprzetworzonej, gdy np. szukamy specyficznego błędu). Pewne części nagłówka i samej komunikacji są osobnych zakładkach, co np. ułatwia sprawdzanie tzw. ciasteczek. Można również analizować czasy komunikacji i wiele innych elementów protokołu http(s).

Dzięki takiej formie prezentacji łatwiej jest obserwować przekazywanie danych do i z serwera WWW.

W odpowiednich zakładkach, dla zapytania typu GET, można znaleźć parametry przekazane poprzez tzw. **query string**, czyli ciąg danych w odpowiednim formacie, który może być obecny w adresie URL po znaku zapytania '?'.
 W tym celu należy kliknąć na zakładkę Headers (nagłówki) i przejść do sekcji Request Headers (nagłówki żądania).

Analogicznie można znaleźć dane w ciele zapytania typu POST. W tym przypadku w zależności od wybranego formatu może być to zarówno format jak dla query string jak i format JSON, XML itd.

W zakładce odpowiedzi (ang. response) znajdujemy ciało odpowiedzi na zapytanie. Najczęściej jest to strona w formacie HTML, ale mogą tam być również dane w postaci JSON czy XML.

Pierwsze elementy formatu HTML

Na wykładzie przedstawione były podstawowe składowe elementy dokumenty HTML, jak:

- `<html>` - główny element dokumentu HTML
- `<head>` - nagłówek dokumentu i jego podstawowe składowe:
 - o `<meta>` - element opisu dokumentu, nie prezentowane przez przeglądarkę
 - o `<title>` - tytuł dokumentu
- `<body>` - ciało dokumentu, które zawiera całą pozostałą informację.

W ciele dokumentu jest bardzo duża różnorodność elementów, które albo występują jedno za drugim albo jedno w drugim. Podstawowym jest paragraf `<p>` zawierający tekst pokazywany w przeglądarce. Można też wstawiać komentarze za pomocą `<!-- -->`.

The diagram illustrates the structure of a `Welcome.html` file. The code is shown in a text editor with line numbers 1 to 13. Annotations in yellow boxes point to specific parts of the code:

- Element head** points to the `<head>` tag on line 3.
- Element title zawierający tekst Welcome** points to the `<title>Welcome</title>` tag on line 7.
- Komentarz HTML, nie interpretowany przez przeglądarkę** points to the `<!-- This is a comment. -->` on line 10.
- Element p wewnątrz body** points to the `<p>This is a paragraph in Html5.</p>` on line 11.

The diagram shows a web browser window titled "Welcome". The address bar displays the file path: `file:///C:/Users/darius/Dysk Google/Dyd`. The main content area of the browser shows the text: "This is a paragraph in Html5." A yellow box with the text "Karta pokazuje zawartość elementu title" has an arrow pointing to the browser window, indicating that the browser displays the content of the `<title>` element.

Do podstawowych elementów można też zaliczyć nagłówki o różnych poziomach od `<h1>` do `<h6>` zawierające tekst nagłówka. Domyślnie posiadają one ustawione pewne wielkości czcionki od większej do coraz mniejszej. Możliwość Internetu pokazuje dopiero możliwość przenoszenia się z jednych dokumentów do innych, co osiąga się za pomocą tzw. linków. Tak naprawdę w

dokumentcie istnieją elementy kotwiczące (ang. anchor) <a>, które za pomocą atrybutu href pozwalają podać do jakiego dokumentu ma przejść przeglądarka po kliknięciu tekstu, który jest w środku tego elementu. W dokumencie można również wstawiać obrazki za pomocą elementu uzupełniając odpowiednim adresem atrybut src.

Walidacja dokumentu HTML

Oprócz poprawności elementów samych w sobie, czyli np. istnienia dla każdego znacznika otwierającego (np. <body>) znacznika zamykającego (np. </body>), zawierania się znaczników w całości w innych znacznikach, istnieją dodatkowe reguły poprawności. Jeśli je nie zastosujemy, przeglądarka będzie starała się naprawić nasz dokument podczas prezentacji na ekranie. Może się jednak okazać, że inna przeglądarka będzie prezentować to inaczej. Warto zatem pisać poprawne składniowo dokumenty. W tym celu można albo wykorzystać wtyczki do programistycznych środowisk (IDE), albo sprawdzić za pomocą walidatorów na stronach WWW, np. validator.w3.org/#validate-by-upload.

List zadań

1. Dla wybranej strony WWW pokaż działanie komendy curl. Pokaż zawartość nagłówków i ciała odpowiedzi.
2. Dla wybranej strony WWW pokaż działanie narzędzi programistycznych wybranej przeglądarki. Pokaż jakie żądania wysłała przeglądarka i jakie otrzymała odpowiedzi. Zaprezentuj rodzaj żądania, jego nagłówek i ciało. Podobnie dla odpowiedzi.
3. Znajdź stronę używającą query string w adresie URL i pokaż działanie, wpisując nowy query string z klawiatury.
4. Znajdź stronę z formularzem, która przesyła dane w ciele żądania rodzaju POST. Zaprezentuj, że rzeczywiście tak się dzieje.
5. Stwórz własną poprawną stronę (lub więcej stron) WWW, która będzie zawierać:
 - a. tytuł,
 - b. znaczniki dla różnych rodzajów nagłówków,
 - c. tekst w paragrafach,
 - d. komentarze,
 - e. Podlinkowane:
 - i. inne strony WWW,
 - ii. obrazki,
 - iii. jakiś spakowany plik,
 - iv. adres email
6. Pokazać wybranym walidatorem, że strona jest poprawna.

Termin oddania: Spotkanie 2 (0 punktów)