



HTTP

Trener: Michał Michalczuk

Gdańsk, 21 listopada 2018 roku

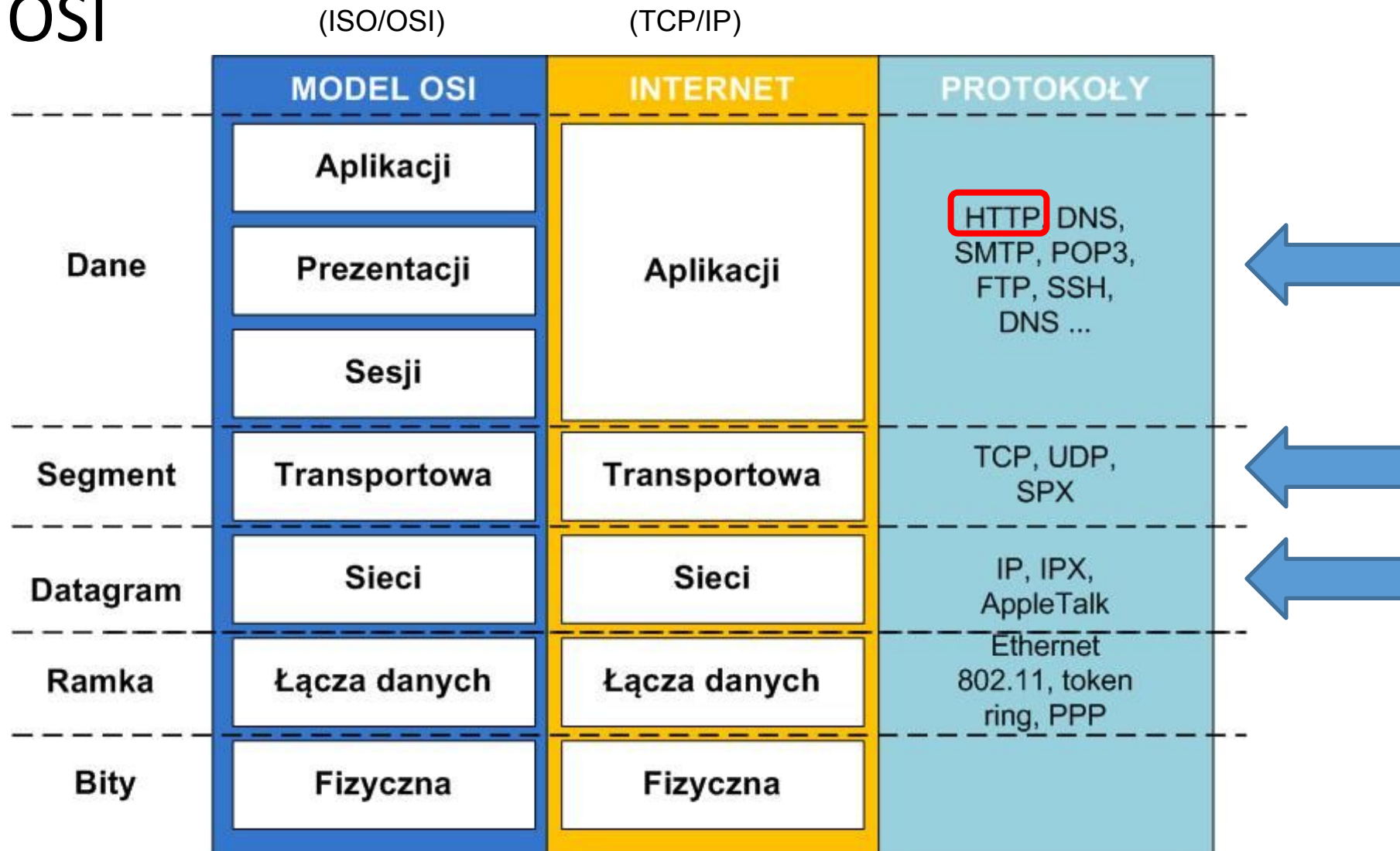
www.infoshareacademy.com

Plan na dzisiaj

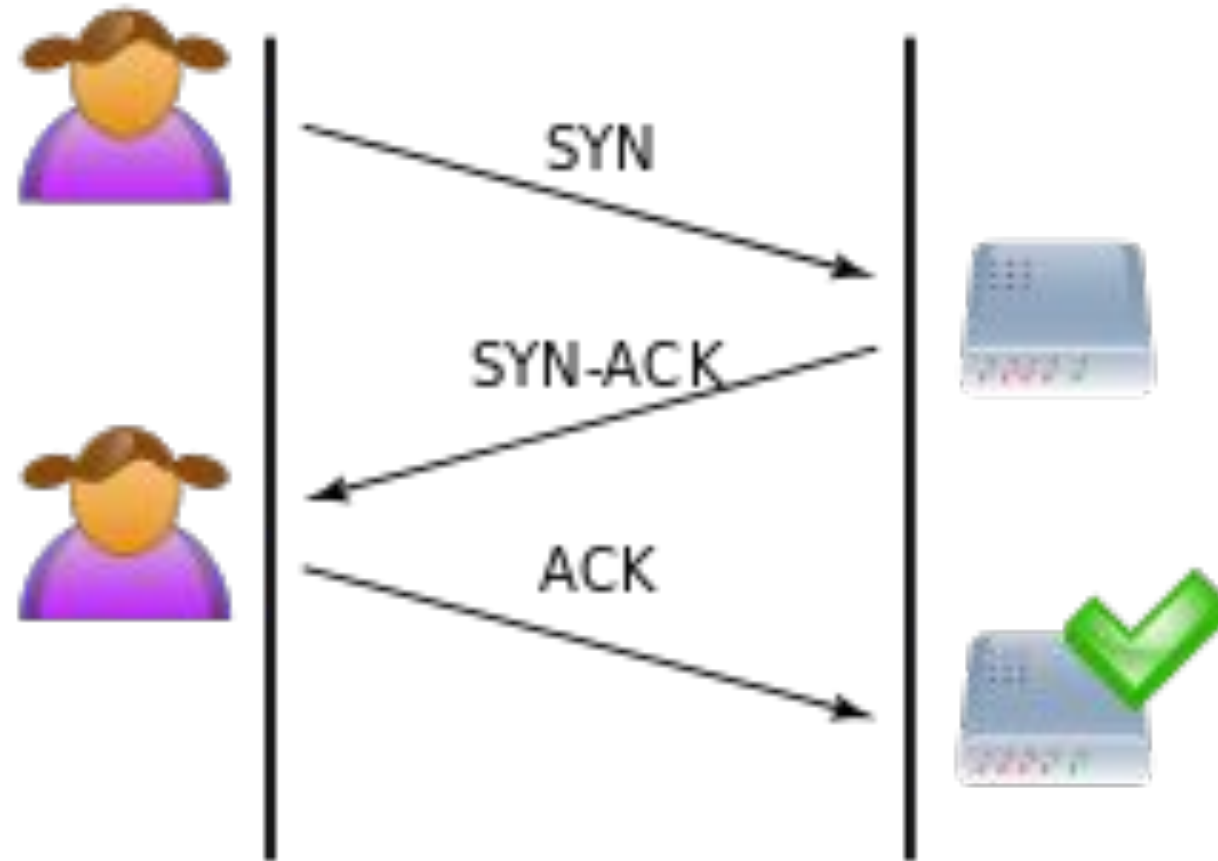
- TCP/IP OSI/ISO
- Czym jest protokół HTTP
- Request / Response
- HTTP a stan
- Uwierzytelnianie
- HTTPS

TCP/IP a ISO/OSI

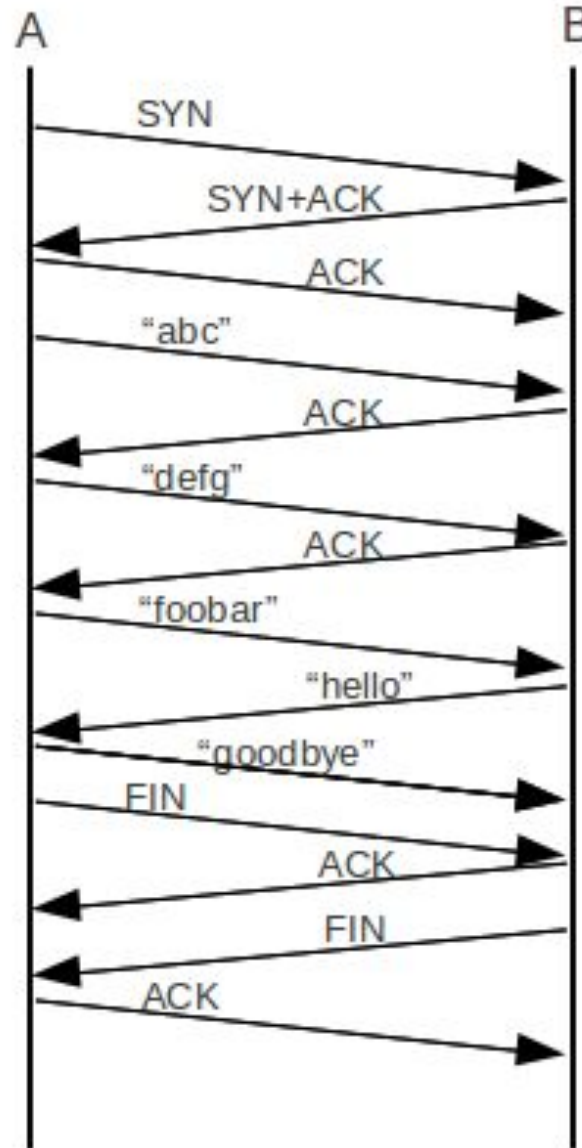
ISO / OSI



TCP - handshake



TCP - cały flow



TCP (Transmission Control Protocol) **UDP** (User Datagram Protocol)

TCP

You ask a friend if he can play with the toy.

Your friend asks you if you actually asked him for the toy.

You tell your friend that you asked for that toy.

He gives you the toy.

UDP

Your friend throws a toy at you and walks away.

Explain TCP like I'm five

You and a friend need to share a toy

TCP

1. You ask a friend if you can play with the toy.
2. Your friend asks you if you actually asked him for the toy.
3. You tell your friend that you asked for that toy.
4. He gives you the toy.

UDP

1. Your friend throws a toy at you and walks away.

<https://dev.to/ben/explain-tcp-like-im-five>

Dlaczego to istotne?

- Złożoność
- Wiele protokołów

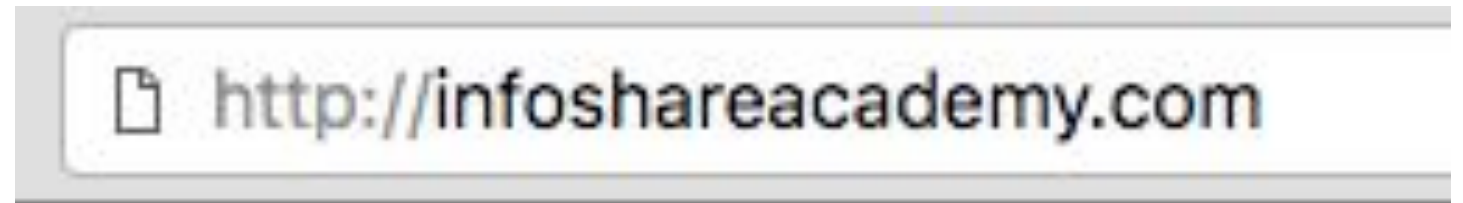


HTTP

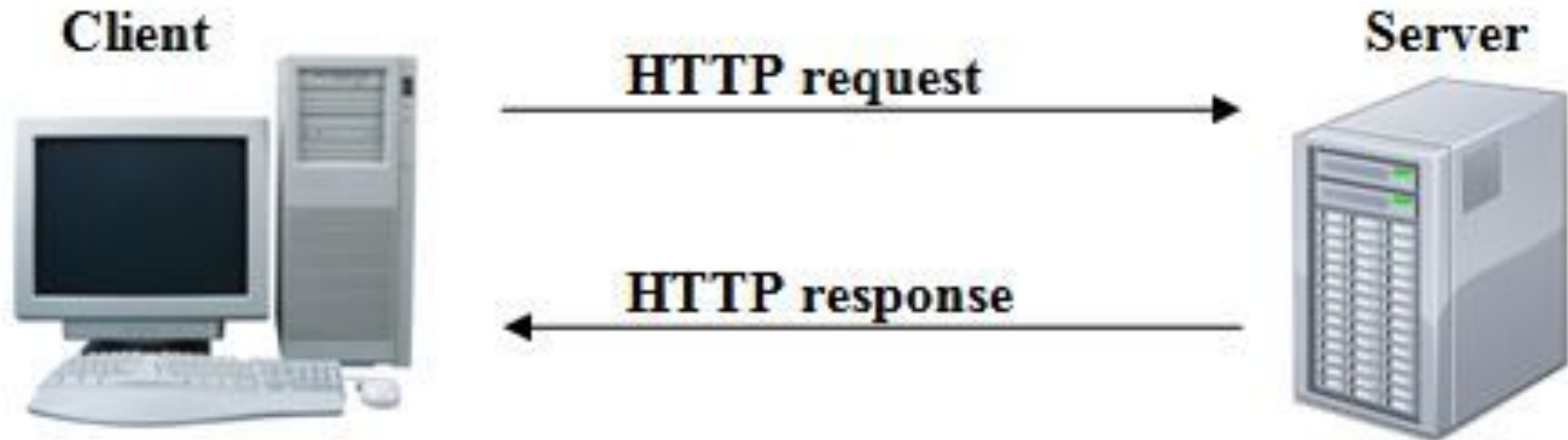
 <http://infoshareacademy.com>

Hypertext Transfer Protocol (HTTP)

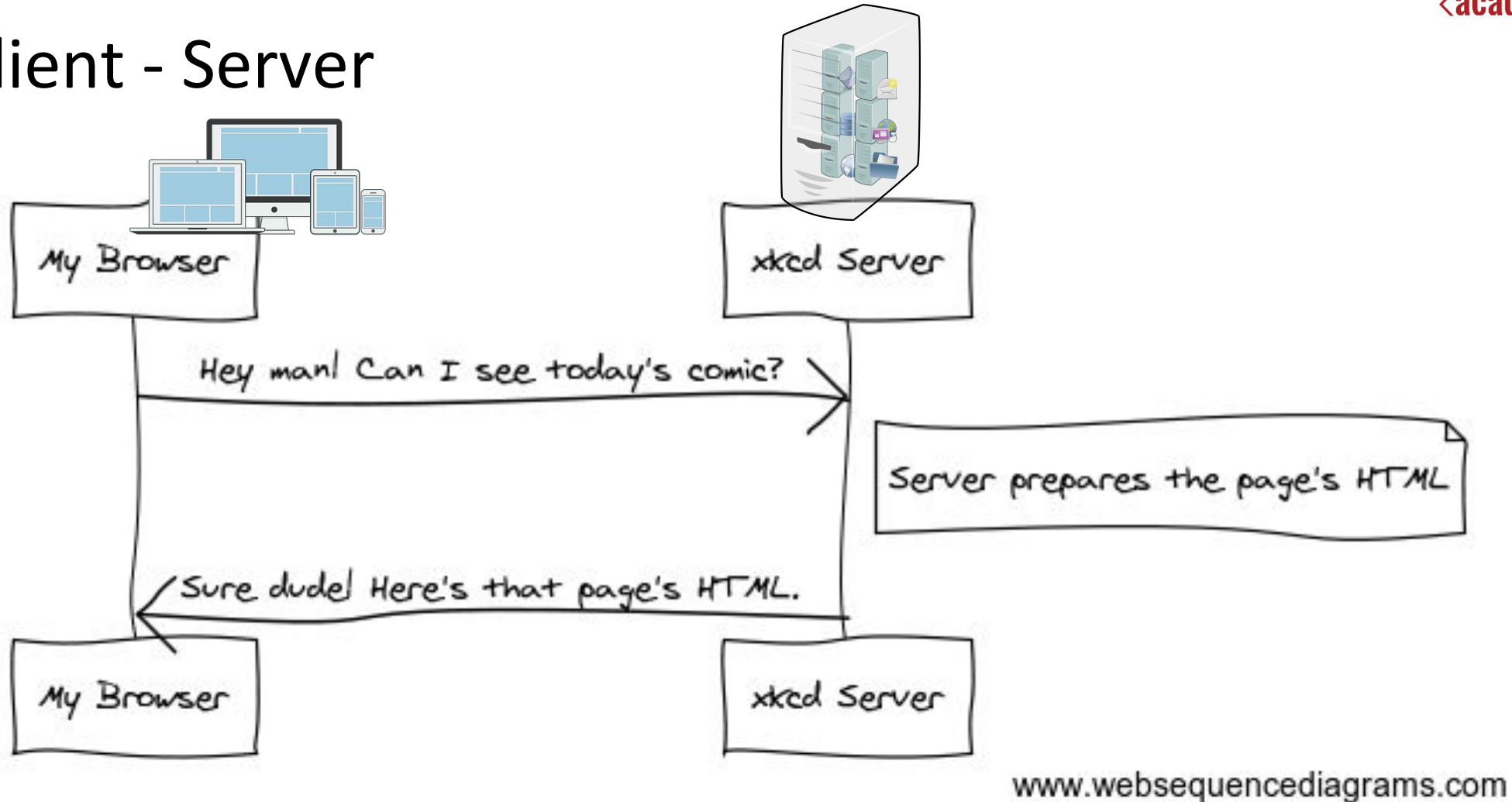
- Tekstowy
- Bezstanowy
- Request-Response (client-server)
- Znormalizowany



Client - Server



Client - Server



Web server / Serwer WWW

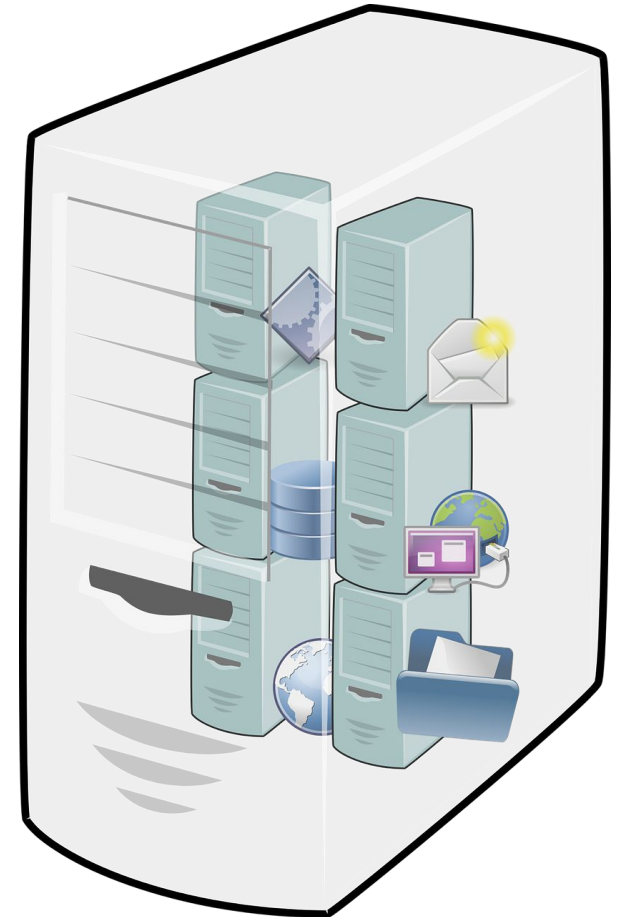
Program obsługujący **żądania HTTP**.

Gdzie?

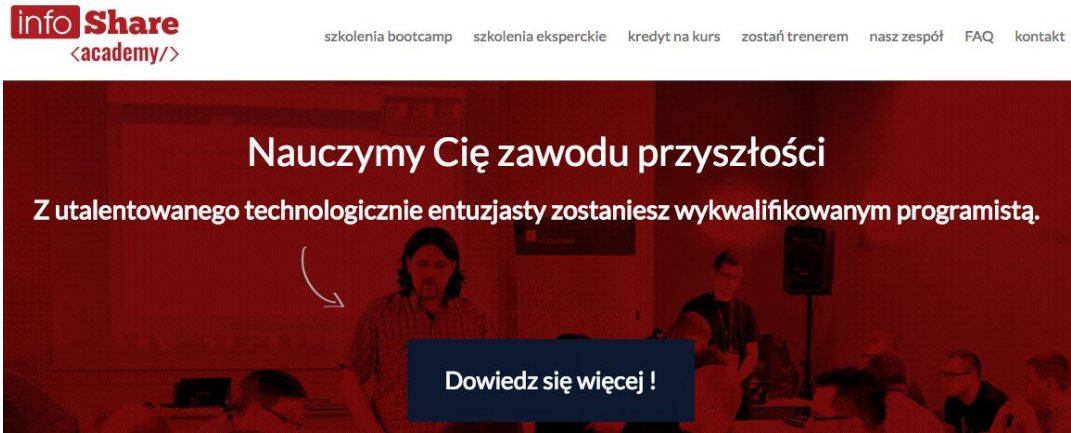
Np na serwerze

(komputer klasy serwer/dedykowany)

albo na naszym komputerze



[EX] Jak to wygląda?



Przyjrzyjmy się stronie akademii - jakie dane klient

(nasza przeglądarka) pobiera z serwera.

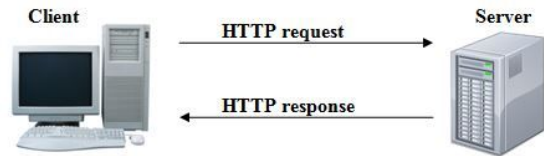
- Jakich danych żąda (request)
- Jakie dane dostaje (response)

Przydatne narzędzia:

Firefox: <https://addons.mozilla.org/pl/firefox/addon/restclient/>

Chrome: <https://chrome.google.com/webstore/detail/postman/fhbjgbiflinjbdggehcddcbncdddomop>

HTTP request HTTP response



Z czego składa się request?

- URL
- HTTP verb (method)
- headers

URL – jak go czytać?



Źródło: <http://antezeta.com/news/campaign-tracking>

URL - przykładowo

<http://infohareacademy.com>

<http://api.twitter.com/1.1/search/tweets.json>

https://api.twitter.com/1.1/search/tweets.json?q=%23superbowl&result_type=recent

<https://my-app.com/api/clients>

<http://my-app.com/api/clients/2>

HTTP verb (method)

GET <http://infoshareacademy.com>

GET <http://api.twitter.com/1.1/search/tweets.json>

GET https://api.twitter.com/1.1/search/tweets.json?q=%23superbowl&result_type=recent

POST <https://my-app.com/api/clients>

PUT <http://my-app.com/api/clients/2>

DELETE <http://my-app.com/api/clients/2>

HTTP verb (method)

Verb	Znaczenie	Przykład
GET	Pobierz dane	GET <code>http://my-app.com/api/clients</code>
POST	Wyślij dane. Utwórz nowy obiekt. Możemy przesłać dane w ciele ("body"/"payload")	POST <code>http://my-app.com/api/clients</code> body: { <i>name: "Evil corp"</i> }
PUT	Wyślij dane i stwórz albo uaktualnij obiekt Możemy przesłać dane w ciele ("body"/"payload")	PUT <code>http://my-app.com/api/clients/2</code> body: { <i>name: "Best corp"</i> }
DELETE	Usuń wskazany obiekt	DELETE <code>http://my-app.com/api/clients/2</code>

Headers (nagłówki) - w żądaniu (od klienta)

Meta informacje o żądaniu.

Wybrane / Popularne

- Accept
- Accept-Language
- Content-Type (np. *application/json* , *application/xml*)
- Cookie
- User-Agent



Headers (nagłówki) - mogą być też w odpowiedzi

Meta informacje o odpowiedzi.

Wybrane / Popularne

- Content-Encoding
- Content-Type (np. *application/json* , *application/xml*)
- Set-Cookie
- Access-Control-Allow-Origin

Lista **standardowych** nagłówków:

https://en.wikipedia.org/wiki/List_of_HTTP_header_fields



Pełen request - przykłady

GET <http://my-address.com/path>

Headers:

User-Agent: *"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_2) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/63.0.3239.132 Safari/537.36"*

Accept: *"text/plain,application/json"*

DELETE <http://my-address.com/path>

Headers:

User-Agent: *"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_2) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/63.0.3239.132 Safari/537.36"*

Accept: *"text/plain,application/json"*

Pełen request - przykłady

POST <http://my-address.com/path>

Body:

```
{  
  "content": "Mine data",  
  "anotherProperty": "Also mine data"  
}
```

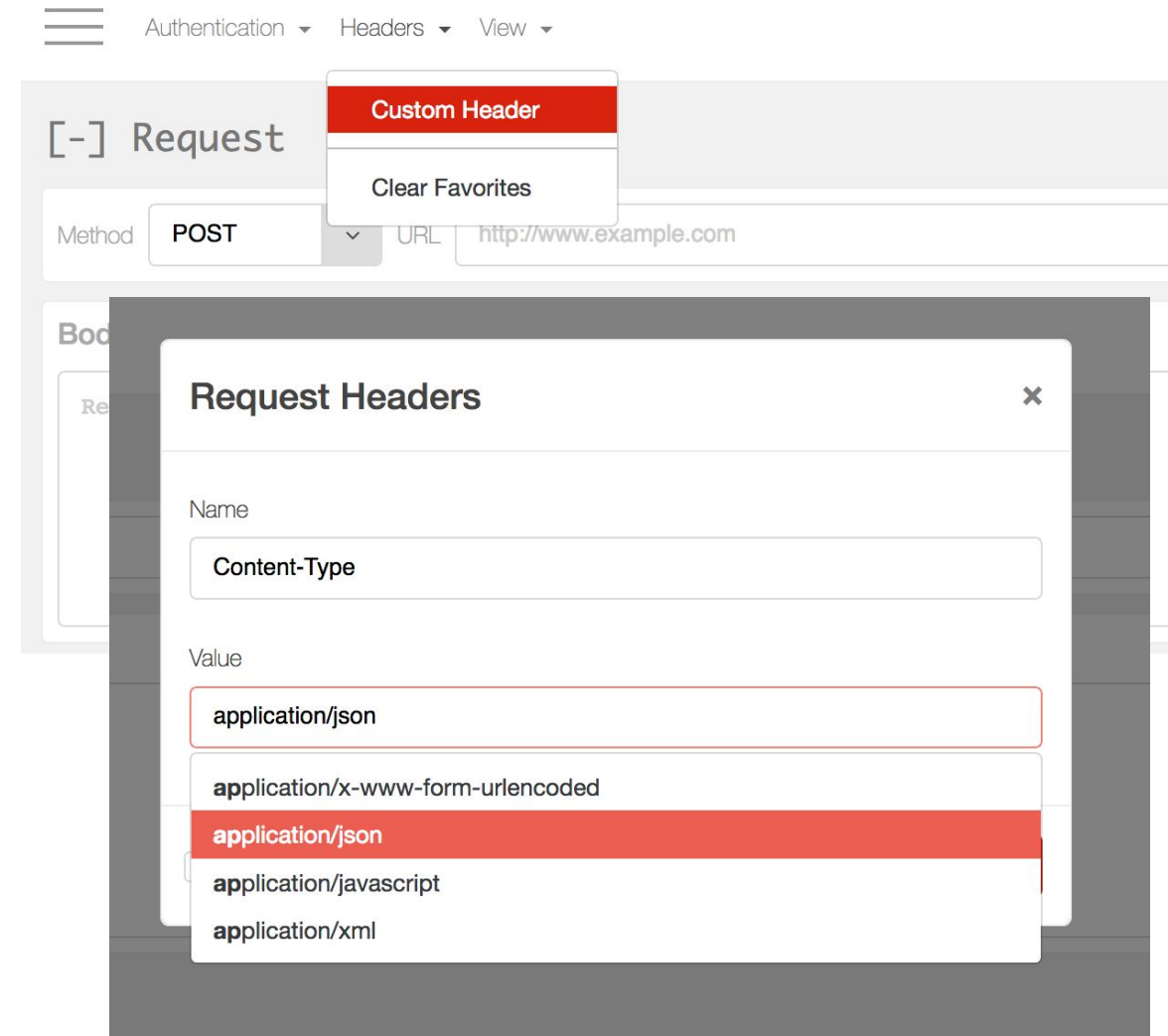
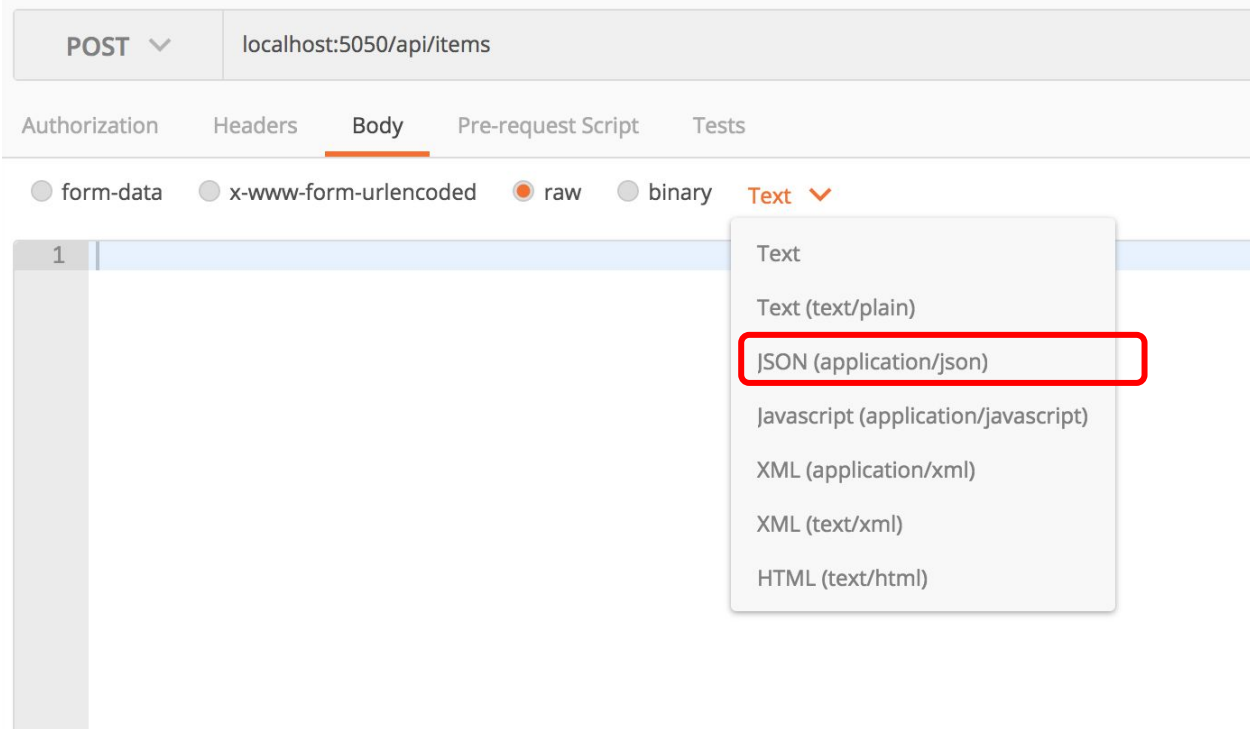
Headers:

User-Agent: *"Mozilla/5.0 (Macintosh; Intel Mac OS X 10_13_2) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/63.0.3239.132 Safari/537.36"*

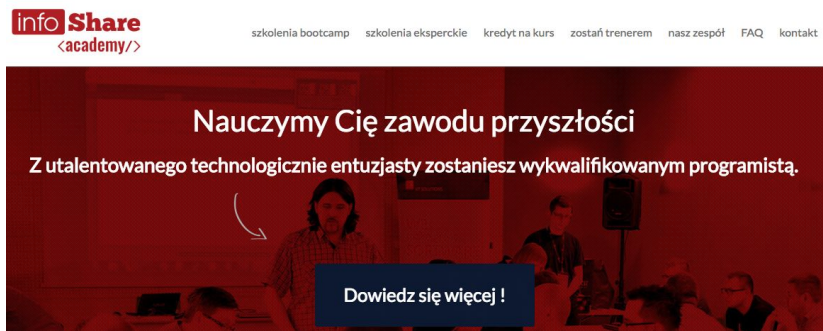
Content-type: *"application/json"*

Accept: *"text/plain,application/json"*

Jak ustawić nagłówek (header) Content-Type



[EX] Wyślijmy parę requestów



Korzystając z POSTMAN lub Advanced REST Client:

Wykonaj requesty do strony ISA:

- GET <http://infoshareacademy.com>
- POST <http://infoshareacademy.com>

body:

```
{ "courseName": "JJDZ4", "id": 15 }
```

- DELETE <http://infoshareacademy.com/courses/11>

Co się udało?

Udało === kod odpowiedzi 200

Przydatne narzędzia:

Firefox: <https://addons.mozilla.org/pl/firefox/addon/restclient/>

Chrome: <https://chrome.google.com/webstore/detail/postman/fhbjgbiflinjbdgggehcdcbncdddomop>

Ok, dostajemy HTML. Ale słyszałem że możemy też wysyłać/pobierać/usuwać dane

Dokładnie.

Usługi/Serwery które mają taką odpowiedzialność wystawiają nam **API** do komunikacji.



Czym jest API

Application Program Interface.

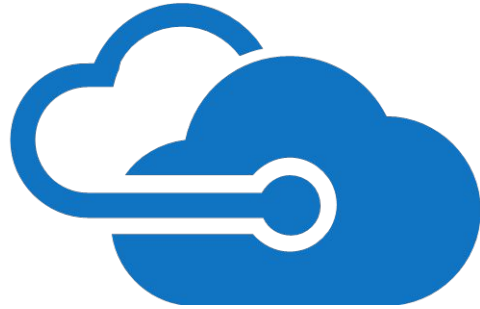
Opisuje jak elementy się ze sobą komunikują, wymieniają dane, jakie dane, jak sformatowane.

Jest takim **kontraktem**.

W naszym kontekście: zbiorem URL, metod HTTP oraz formatów danych.



[EX] Wyślijmy parę requestów do mnie



Hint: moje API przyjmuje dane w formacie JSON

(nagłówek **Content-Type** jest istotny)

Co się udało?

Korzystając z POSTMAN lub Advanced REST Client:

Wykonaj requesty do mojego API

<https://isa-simple-rest-api.herokuapp.com/>

- GET ~/api/users
- POST ~/api/users

body:

```
{ "username": "whatever you want :)" }
```

- GET ~/api/lists?userId=2
- GET ~/api/users/3
- PUT ~/api/users/1 (najlepiej użyj id usera którego sam stworzyłeś)
- DELETE ~/api/users/1 (spróbuj różnych *id*)

Niespodzianka. Moje API ma dokumentację



Dokumentacja:

<https://isa-simple-rest-api.herokuapp.com/api/documentation>

OpenAPI Specification - jedna specyfikacja którą opisujemy metody w API.

<https://swagger.io/>

Response – co dostaliście w odpowiedzi

- Kod http

Status: 200 OK

- Body (treść)

```
{
  "id": 1,
  "content": "First item"
},
```

- Headers (nagłówki)

Content-Type → application/json; charset=utf-8

Date → Sat, 17 Dec 2016 15:38:48 GMT

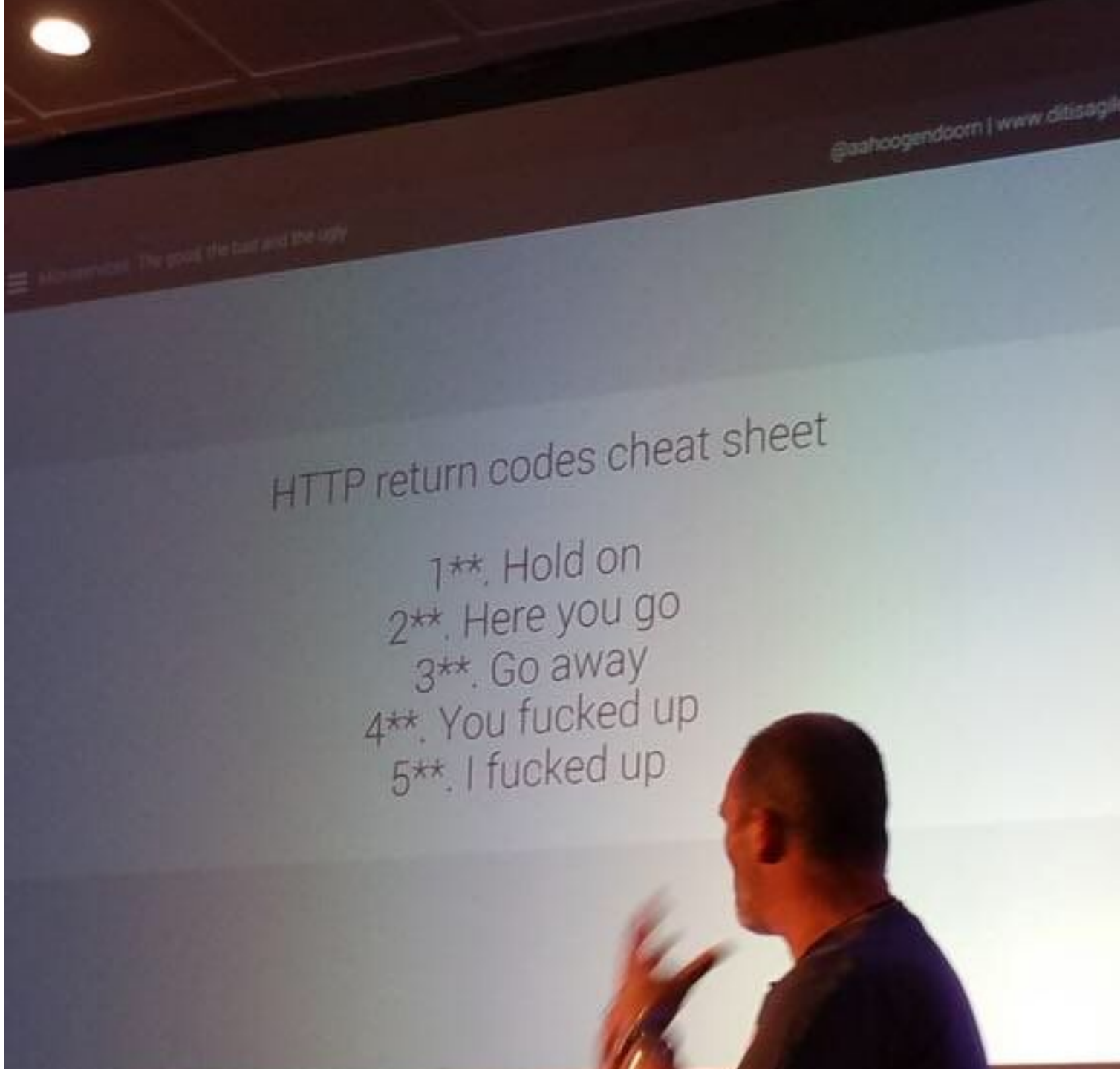
- Cookies (ciasteczka ... nie tym razem)



Kody HTTP - główny podział

- 1xx – informacyjne
- 2xx – sukces
- 3xx – przekierowania
- 4xx - błąd po stronie klienta (np 404)
- 5xx - błąd po stronie serwera

Status: **200 OK**



Kody HTTP - najpopularniejsze

- 200 – OK
- 201 - Created
- 400 – Bad Request
- 401 – Authentication Required
- 403 – Forbidden
- 404 – Not Found
- 500 – Internal Server Error
- 503 – Service Unavailable

Status: **200 OK**

Body - treść odpowiedzi

JSON

```
[
  {
    "id": 1,
    "content": "First item"
  },
  {
    "id": 2,
    "content": "Another item"
  }
]
```

XML

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
  <element>
    <content>First item</content>
    <id>1</id>
  </element>
  <element>
    <content>Another item</content>
    <id>2</id>
  </element>
</root>
```



Response – GET

The screenshot displays the 'Body' tab of a web browser's developer tools. The request method is 'GET' and the URL is 'https://isa-example-api.azurewebsites.net/api/items'. The status is '200 OK' and the response time is '978 ms'. The response body is a JSON array containing three objects, each with an 'id' and a 'content' field. The JSON is formatted in a 'Pretty' view.

GET ▼ https://isa-example-api.azurewebsites.net/api/items

Body Cookies Headers (7) Tests

Status: 200 OK Time: 978 ms

Pretty Raw Preview JSON ▼  

```
1 [
2   {
3     "id": 1,
4     "content": "First item"
5   },
6   {
7     "id": 2,
8     "content": "Another item"
9   },
10  {
11    "id": 3,
12    "content": "One more item"
13  }
14 ]
```

Response – GET

GET ▾

https://isa-example-api.azurewebsites.net/api/items

Body

Cookies

Headers (7)

Tests

Status: 200 OK

Content-Encoding → gzip

Content-Type → application/json; charset=utf-8

Date → Sat, 17 Dec 2016 15:07:32 GMT

Server → Microsoft-IIS/8.0

Transfer-Encoding → chunked

Vary → Accept-Encoding

X-Powered-By → ASP.NET

Response – POST

POST ▾

https://isa-example-api.azurewebsites.net/api/items

Body

Cookies

Headers (6)

Tests

Status: 201 Created

Pretty

Raw

Preview

JSON ▾



```
1 {  
2   "id": 4,  
3   "content": "New item"  
4 }
```

Response – POST

POST ▼

https://isa-example-api.azurewebsites.net/api/items

Body

Cookies

Headers (6)

Tests

Status: 201 Created

Content-Type → application/json; charset=utf-8

Date → Sat, 17 Dec 2016 15:22:05 GMT

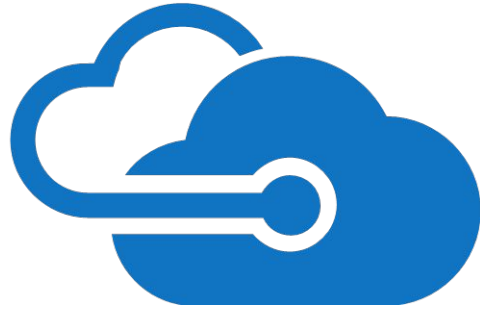
Location → https://isa-example-api.azurewebsites.net/api/Items/4

Server → Microsoft-IIS/8.0

Transfer-Encoding → chunked

X-Powered-By → ASP.NET

[EX] Wyślijmy znowu parę requestów do mnie



Korzystając z POSTMAN lub Advanced REST Client:

Wykonaj requesty do mojego API

<https://isa-simple-rest-api.herokuapp.com>

- POST ~/api/alwaysbad
- GET ~/api/alwaysbad
- PUT ~/api/users/1000
body: { "username": "foo" }
- DELETE ~/api/users/1001
- POST ~/api/users (z pustym body, kompletnie pustym)
- POST ~/api/users
body: { "username": false }

**Zwróćcie uwagę na odpowiedzi,
zwłaszcza na kody.**

Pamiętajcie o nagłówku **Content-Type**?

Co się udało, jakie kody dostaliście?

GET vs POST - formularze

```
<form action="/users">
```

First name:

```
<input type="text" name="firstname" value="Mickey">
```

Last name:

```
<input type="text" name="lastname" value="Mouse">
```

```
<input type="submit" value="Send">
```

```
</form>
```

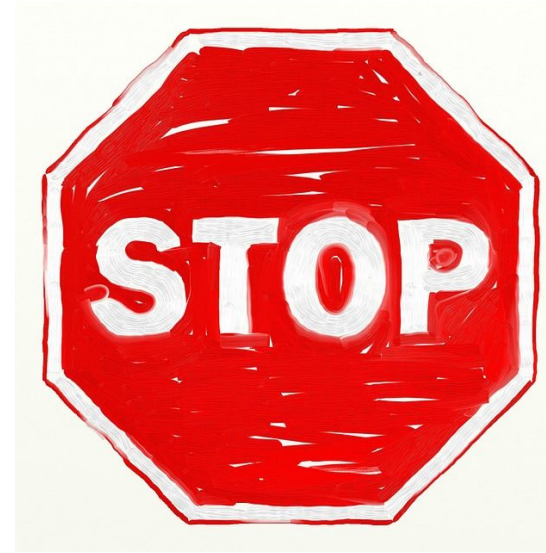


POST *~/users*

body: *firstname=Mickey&lastname=Mouse*



GET vs POST - formularze



```
<form onsubmit="sendAsGet()">
```

First name:

```
<input type="text" name="firstname" value="Mickey">
```

Last name:

```
<input type="text" name="lastname" value="Mouse">
```

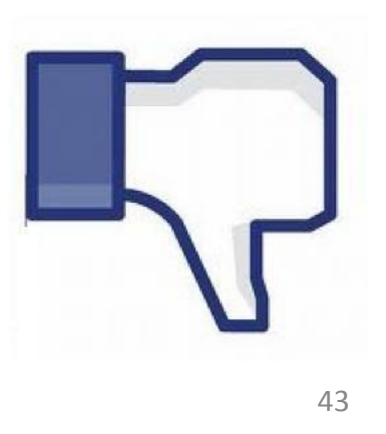
JavaScript magic

GET

~/users/create?firstname=Mickey&lastname=Mouse

```
<input type="submit" value="Send">
```

```
</form>
```



GET **vs** POST

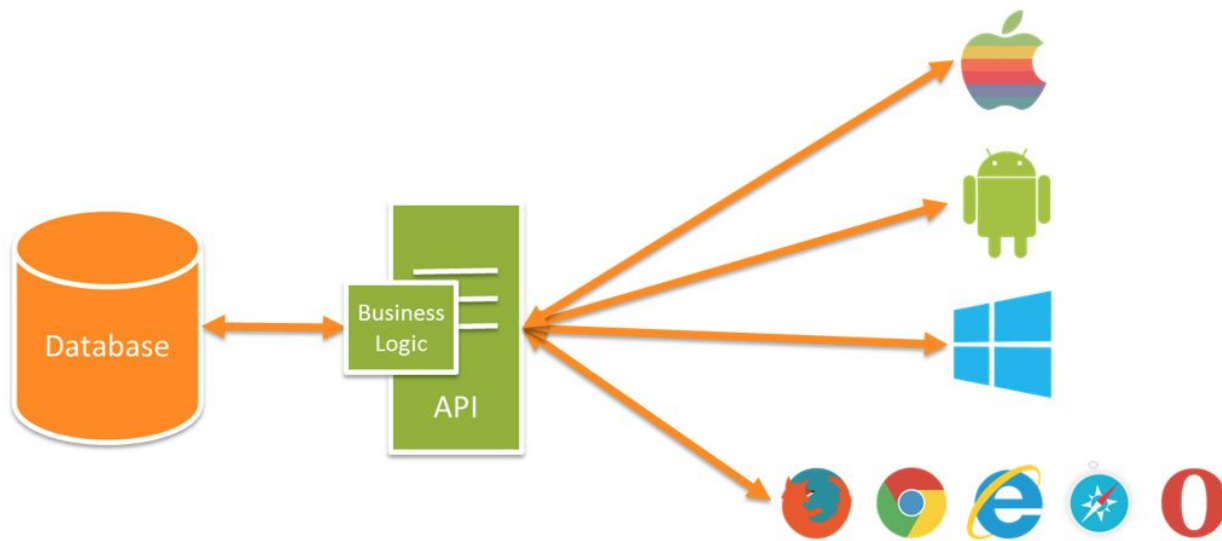
GET

- idempotentny
- n wywołań - żadnego efektu
- n wywołań - zawsze ten sam wynik

POST/PUT/DELETE

- zmiana stanu serwera
- dodanie/zmiana/usunięcie danych
- n wywołań - różne wyniki, lub błędy

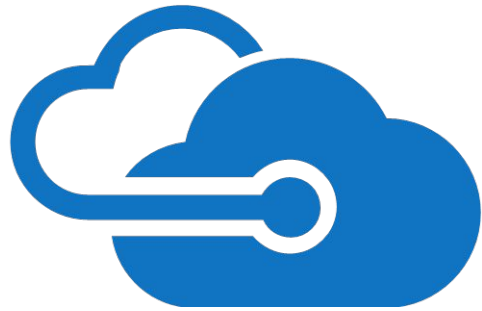
Jacy klienci mogą “gadać” z API?



DOWOLNI

- przeglądarka
- inny serwer
- aplikacja mobilna
- aplikacja konsolowa
- ...

Mały przykład



- Zapytanie bezpośrednio z konsoli
- Zapytanie z konsolowej aplikacji
Java/JavaScript

Przełączam się na Konsolę/IDE.

Bezstanowość HTTP a ciasteczka



Bezstanowość HTTP

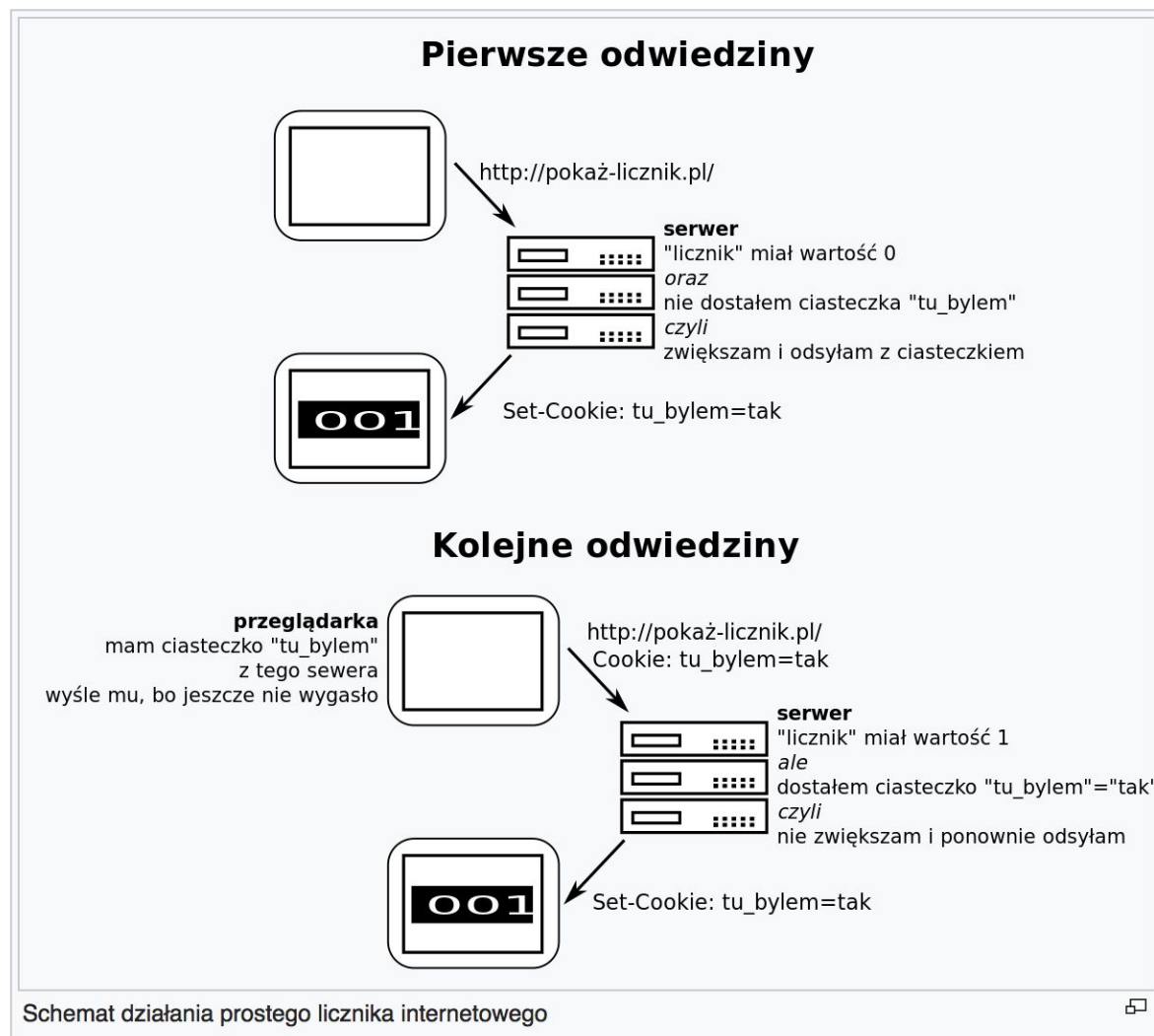
- HTTP nie wie, że ten sam klient wysyła do niego żądania
- Każde żądanie musi być kompletne – serwer nic sobie nie "dopowie"

Ciasteczka i "stan"



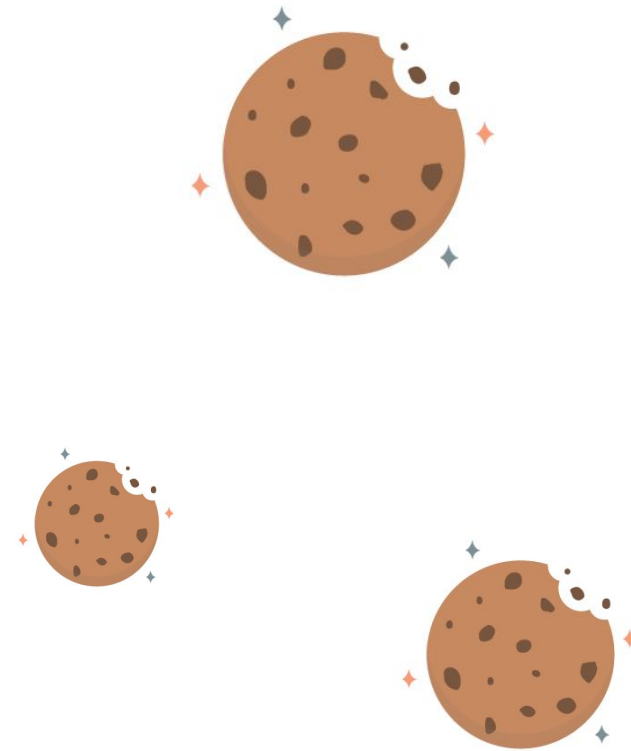
- Ciasteczka
- Sesje po stronie serwera
- Nagłówki - I dane w nich
- Ukryte zmienne w POST
- Parametry żądania np.: <http://my-app.com/?userId=5>

Ciasteczka i "stan"

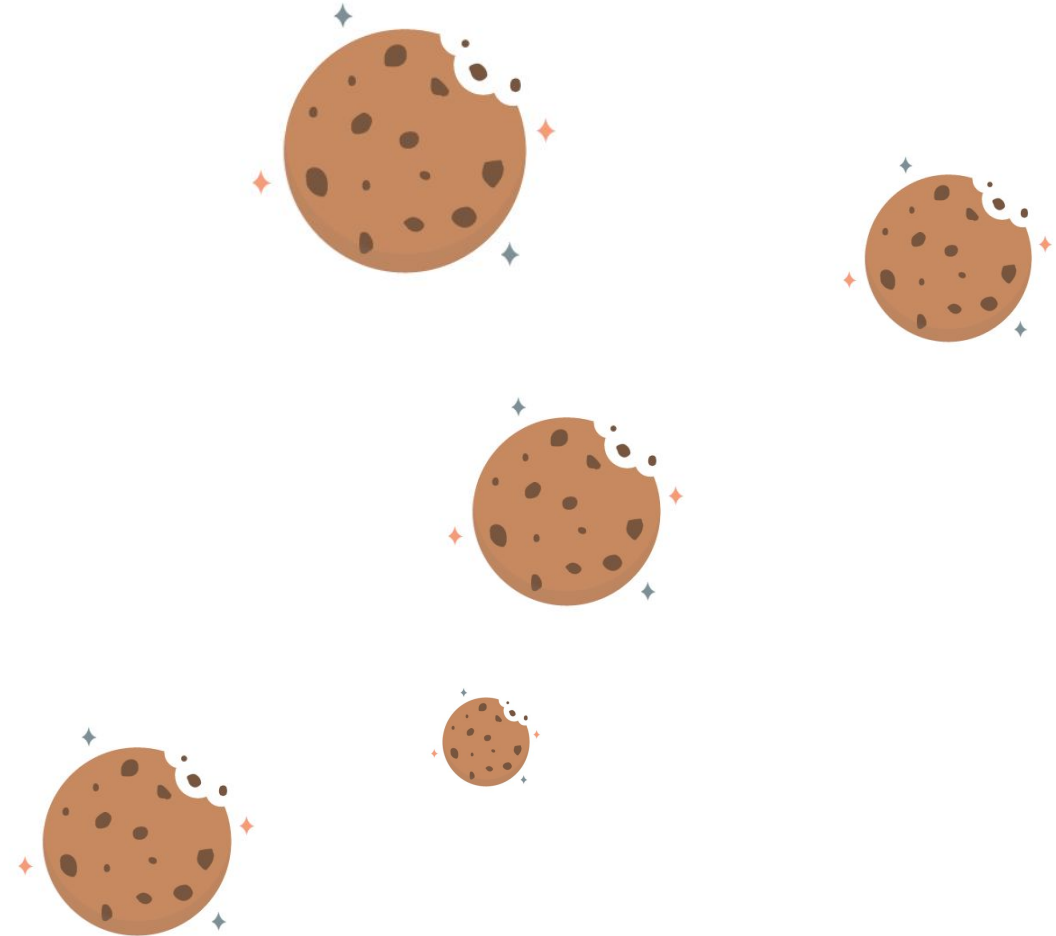
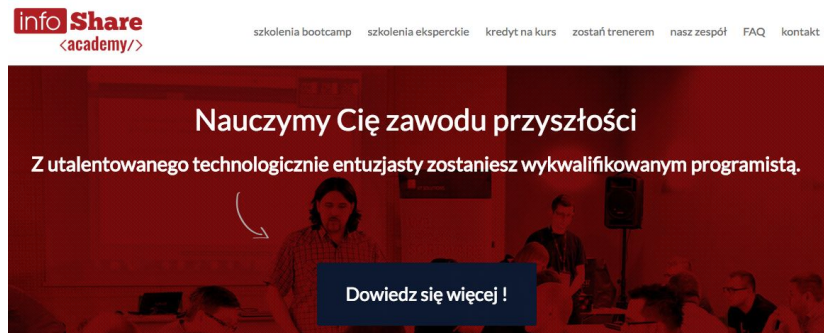


Źródło: https://pl.wikipedia.org/wiki/HTTP_cookie

Ciasteczka i JEE



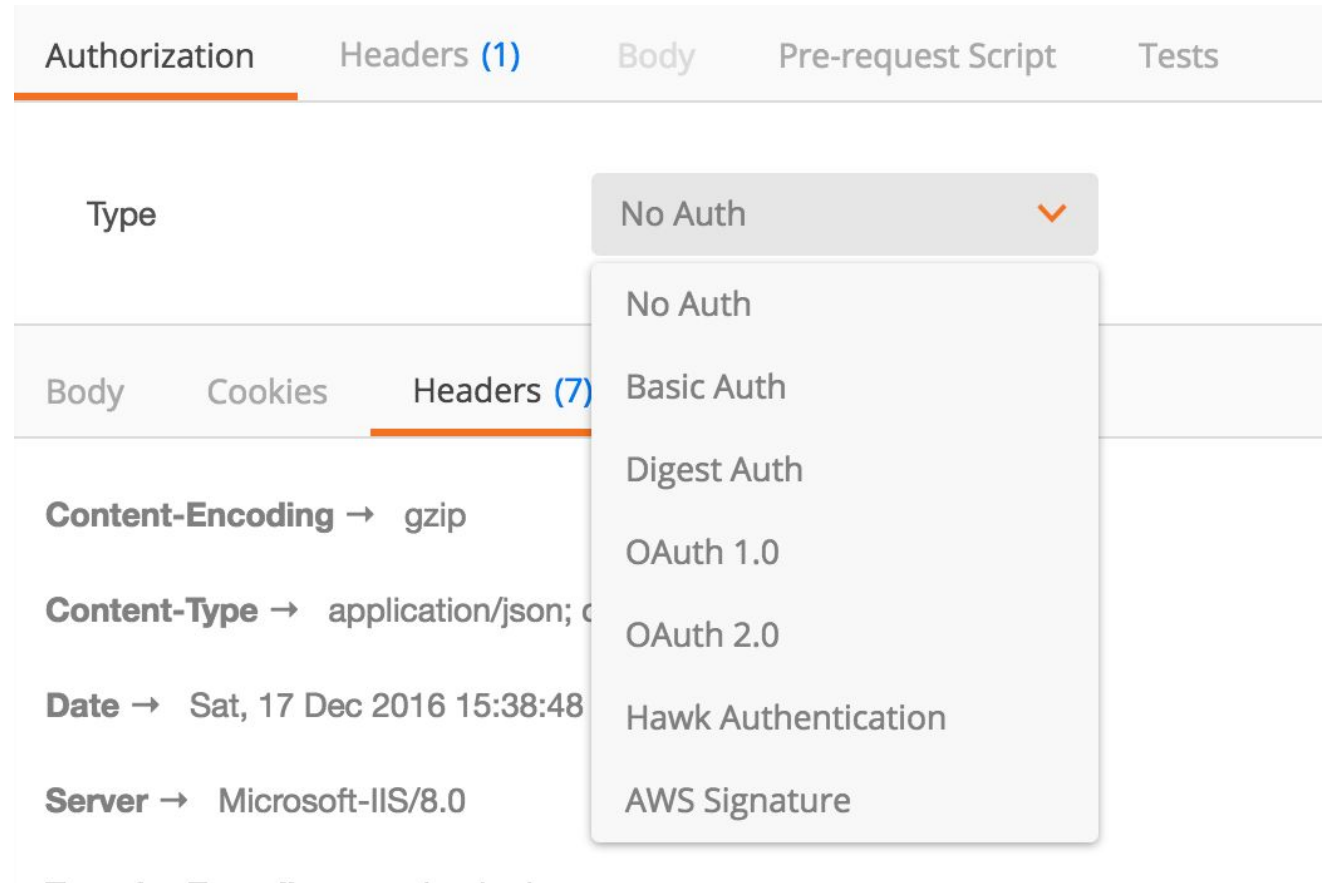
[EX] Zobaczmy ciastka



Uwierzytelnianie



Gdzie, jak ?



Po co ?



An authentication dialog box titled "Authentication" with a blue header bar and a close button (X) in the top right corner. The dialog has a light beige background. On the left, there is a yellow padlock icon. To its right, the text "Enter user name and password for 'example.com'" is displayed. Below this, there are two input fields. The first is labeled "User Name:" and contains the text "Joseph.Smith". The second is labeled "Password:" and contains ten black dots, with a cursor at the end. Below the password field is a checkbox labeled "Save password", which is currently unchecked. At the bottom right of the dialog are two buttons: "OK" and "Cancel".

Authentication

Enter user name and password for 'example.com'

User Name: Joseph.Smith

Password: ●●●●●●●●●●

☐ Save password

OK Cancel

[EX] Wyślijmy znowu parę requestów do mnie



Autoryzacja: **Basic-Auth**

Pamiętajcie o nagłówku **Content-Type**?

Userzy:

```
{
  username: 'michalczukm',
  password: 'michalczukm-secret-pass'
},
{
  username: 'kowalskik',
  password: 'kowalskik-secret-pass'
},
{
  username: 'nowakp',
  password: 'nowakp-secret-pass'
}
```

Część endpointów u mnie wymaga autoryzacji.

Są oznaczone jako **v2**.

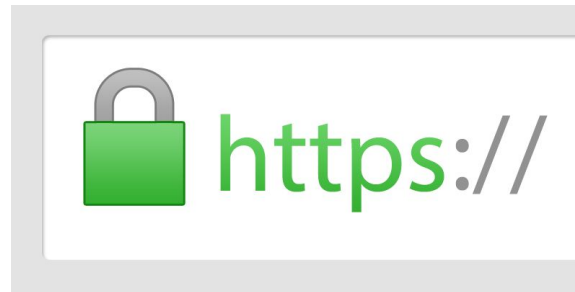
<https://isa-simple-rest-api.herokuapp.com/api/documentation>

Wyślijmy do nich zapytania.

<https://isa-simple-rest-api.herokuapp.com/api/>

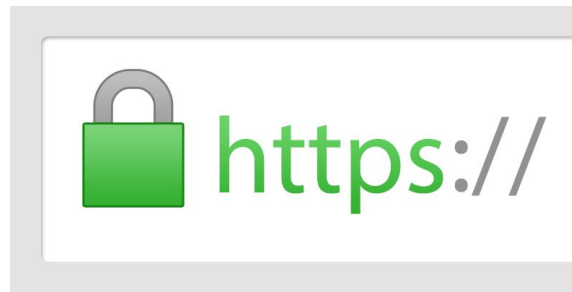
- POST ~/api/v2/items
body: { "content": "some content", "listId": "1" }
- GET ~/api/v2/items

HTTPS a SSL



HTTPS

- HTTP wysyła tekst. **Jawny tekst**
- Warto go zaszyfrować
- SSL (Secure Sockets Layer) szyfruje ruch sieciowy
- Czy HTTPS oznacza że strona na którą wchodzimy jest bezpieczna (np nie wyłudzi od nas danych) ?

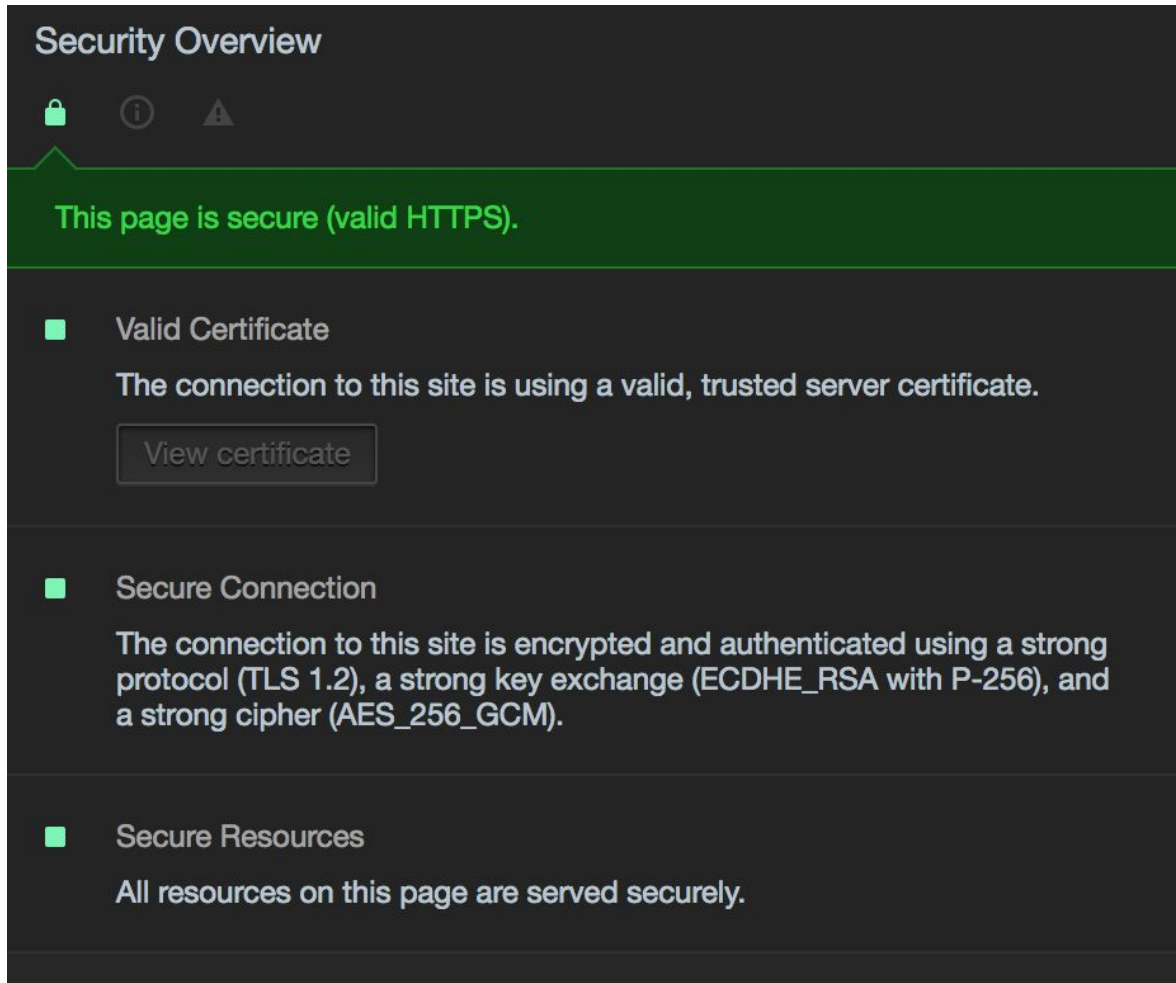


Jak to działa ?



Źródło: <https://www.awardspace.com>

[EX] Zobaczmy taki certyfikat



<https://trello.com>

Dev Tools -> Security

Certyfikaty kosztują ... ale



<https://letsencrypt.org/>

Podsumowanie – protokół HTTP

- Request to HTTP verb + URL + headers + opcjonalnie body
- Response to status + headers + opcjonalnie body
- Do pobierania danych używamy GET
- Do zapisywania danych używamy POST
- HTTP jest bezstanowy
- HTTP wysyła wszystko jawnym tekstem !
- Ciasteczka to takie "udawanie stanu".
- Ciasteczka są wysyłane za każdym requestem
- Ruch po HTTPS jest zaszyfrowany (opisuje to protokół SSL)
- HTTPS wymaga użycia zaufanego certyfikatu
- **Nigdy nie loguj się ani nie wysyłaj wrażliwych danych na strony bez HTTPS**

Parę przydatnych linków

- Mała zajawka o API <https://www.mkyong.com/java/how-to-send-http-request-getpost-in-java/>
- <https://letsencrypt.org/>
- <http://wszystkoociasteczkach.pl/>
- REST API concepts: <https://www.youtube.com/watch?v=7YcW25PHnAA>
- Go deep into REST: <http://www.restapitutorial.com/index.html>

Postman standalone



Dlaczego?

Postman przeniósł się z rozszerzenia Chrome na dedykowaną aplikację w [Electron'ie](#).

<https://www.getpostman.com/>

On Ubuntu: [How to install Postman on Ubuntu](#)

Dziękuję za uwagę



michalczukm



michalczukm@gmail.com