# Obsługa zapytań, protokół HTTP

## Projektowanie i programowanie systemów internetowych I

mgr inż. Krzysztof Rewak

11 marca 2018

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

# Plan prezentacji

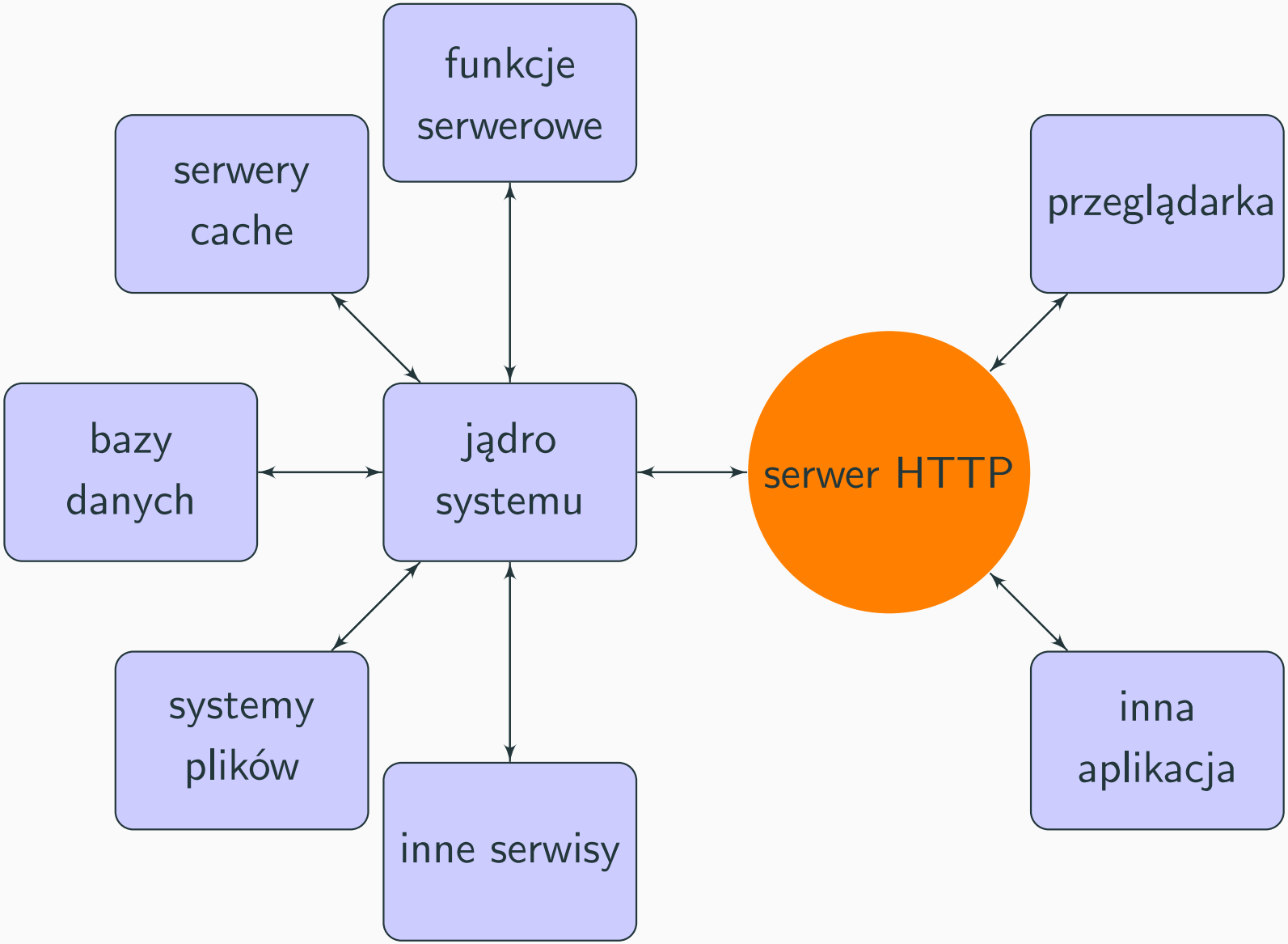
1. Serwery dla systemów internetowych
2. HTTP i jego metody
3. Zapytanie i odpowiedź
4. Kody statusów
5. Routing
6. Podsumowanie

# Serwery dla systemów internetowych

# Serwer? Jaki serwer?

Jak uruchomić system internetowy napisany w dowolnym języku?

# Budowa klasycznego systemu internetowego



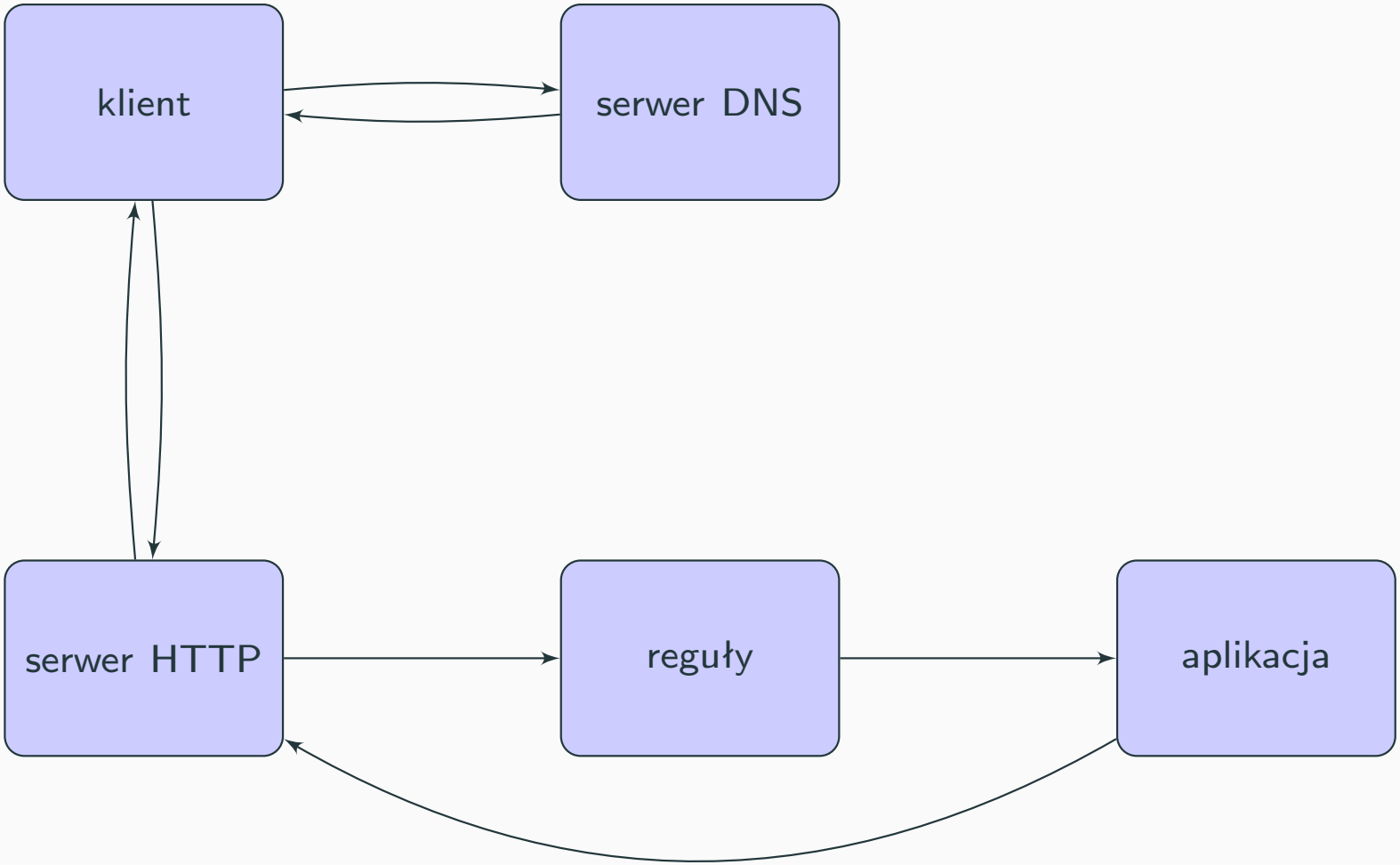
# Obsługa zapytań

Co się dzieje po wpisaniu w pasek adresu przeglądarki internetowej adresu `http://pwsz.rewak.pl`?

# Obsługa zapytań


**Rysunek 1:** Adres wpisany do paska adresu

# Uproszczony cykl życia zapytania



# Funkcje serwera HTTP

Czego powinniśmy wymagać od serwera HTTP:

- obsługi przychodzących zapytań (najczęściej nasłuchując portu :80)
- wyświetlania plików z wyznaczonych katalogów
- możliwości uruchomienia innych serwisów

Każdy z popularnych serwerów oczywiście dostarcza wielu innych ciekawych rozwiązań.

# Przykład *uno*

Po wpisaniu adresu

```
www.pwsz.legnica.edu.pl/test.html
```

przeglądarka wyśle zapytanie

```
GET /test.html HTTP/1.1
Host: www.pwsz.legnica.edu.pl
```

na adres IP 156.17.194.34; serwer Apache obsługujący serwer najpewniej przekierowuje cały ruch na konkretny katalog, na przykład

```
/home/www/pwsz-web/
```

Serwer zwraca błąd 404 (*The requested URL /test.html was not found on this server*) i pokazuje szereg ciekawych informacji o serwerze: `Apache/2.2.9 (Debian) PHP/5.2.6-1+lenny16 with Suhosin-Patch mod_ssl/2.2.9 OpenSSL/0.9.8g Server at www.pwsz.legnica.edu.pl Port 80`

# Przykład *dos*

Po wpisaniu adresu

```
www.pwsz.rewak.pl/test.html
```

przeglądarka wyśle zapytanie

```
GET /test.html HTTP/1.1
Host: www.pwsz.rewak.pl
```

na adres IP 85.255.1.83; serwer Apache obsługujący mój serwer łapie prawie cały ruch z tej domeny i przekierowuje go na plik

```
/home/www/pwsz.rewak.pl/public/index.html
```

Tam załaduje się frontendowa aplikacja, która zbada URL i zwróci odpowiedź, że taki plik nie istnieje.

# Przykład *tré*

Po wpisaniu adresu

```
www.pwsz.rewak.pl/api/courses
```

przeglądarka wyśle zapytanie

```
GET /api/courses HTTP/1.1
Host: www.pwsz.rewak.pl
```

na adres IP 85.255.1.83; serwer Apache obsługujący mój serwer złapie zapytanie do nibykatalogu `api` i przekieruje resztę zapytania do pliku

```
/home/www/pwsz.rewak.pl/public/index.php
```

Przeglądarka otrzyma JSON-a z listą wszystkich prowadzonych przez mnie kursów.

# Najpopularniejsze rozwiązania

- Apache: 47%
- Nginx - 37%
- IIS - 10%
- LiteSpeed - 3%
- Google - 1%
- pozostałe: Tomcat, Node.js, ATS, IWS...

# Magia plików `.htaccess`

Plik umieszczony w katalogu projektu może nadpisać ustawienia Apache. Przykładowo można przekierować cały przekierowany ruch na inny katalog:

```
<IfModule mod_rewrite.c>
        RewriteEngine on
        RewriteRule  ^$ public/    [L]
        RewriteRule  (.*) public/$1 [L]
</IfModule>
```

Warto zapamiętać, że dobrą praktyką jest przekierowanie zapytań HTTP na publiczny folder, w którym nie będzie dostępnych plików aplikacji, repozytorium, itp.

# Magia plików `.htaccess`

Można również przykładowo zablokować dostęp do katalogu wszystkim, którzy nie podadzą poprawnego hasła:

```
AuthType Basic
AuthName "Password Protected Area"
AuthUserFile /path/to/.htpasswd
Require valid-user
```

# Magia plików `.htaccess`

Pliki `.htaccess` pozwolą również na:

- zarządzanie listowaniem zawartości katalogów,
- własne strony błędów,
- obsługę typów plików,
- zarządzanie pamięcią podręczną.

Zmiany są natychmiastowe (ponieważ plik jest wczytywany przy każdym zapytaniu), ale spowalniają serwer.

# Porównanie funkcjonalności serwerów webowych

| Server | Security | | | | Dynamic content[a] | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | basic access authentication | digest access authentication | SSL/TLS https | virtual hosting | CGI | FCGI | SCGI | WSGI | Java Servlets | SSI | ISAPI | SSJS | Runs in user space or kernel space | Administration console | IPv6 | HTTP/2 | QUIC |
| AOLserver | Yes | No | Yes[b][c][d][1] | Yes | Yes | No | Unknown | No | No | Yes | Unknown | Unknown | user | Unknown | Unknown | Unknown | Unknown |
| Apache HTTP Server | Yes | Yes | Yes[e][c][2][f][3] | Yes | Yes | Yes | Yes | Yes[e] | No[g] | Yes | Yes[h] | Unknown | user | Yes[i] | Yes | Yes | Unknown |
| Apache Tomcat | Yes | Yes | Yes[j][4] | Yes | Yes | No | Unknown | No | Yes | Yes | No[k] | Unknown | user | Yes | Yes[l] | Unknown | Unknown |
| Boa | No | No | Yes[m] | Yes | Yes | No | Unknown | No | No | No | No | Unknown | user | Unknown | Yes | Unknown | Unknown |
| Caddy | Yes | No | Yes | Yes | Partial[n] | Yes | No | No | No | No[c] | No | No | user | No | Yes | Yes | Yes |
| Caucho Resin Server | Yes | Yes | paid version[c] | Yes | Yes | Yes | Unknown | No | Yes | Yes | No | Unknown | user | Yes | Yes | Unknown | Unknown |
| Caudium | Yes | Yes | Yes | Yes | Yes | Yes | Unknown | No | Yes | Yes | Unknown | Unknown | user | Yes | Yes[p] | Unknown | Unknown |
| Cherokee HTTP Server | Yes | Yes | Yes | Yes | Yes | Yes | Yes | Yes | No | Yes | No | Unknown | user | Yes | Yes[5] | Unknown | Unknown |
| HFS | Yes | No | Yes via Stunnel[6] | No | No | No | Unknown | No | No | No | Unknown | Unknown | user | Unknown | No | Unknown | Unknown |
| Hiawatha HTTP Server | Yes | Yes | Yes[q][7] | Yes | Yes | Yes | No | No | No | Yes | No | Unknown | user | Yes | Yes | Unknown | Unknown |
| IBM HTTP Server | Yes | Yes | Yes | Yes | Yes | Yes | Unknown | No | No | Yes | No | Unknown | user | Yes | Yes | Unknown | Unknown |
| Internet Information Services | Yes | Yes | Yes | Yes | Yes | Yes | Yes | No | No[r] | Yes | Yes | Yes | kernel and user[8] | Yes | Yes | Yes | Unknown |
| Jetty | Yes | Yes | Yes | Yes | Yes | Unknown | Unknown | No | Yes | Unknown | Unknown | Yes | user | Unknown | Unknown | Yes | Unknown |
| Jexus | No | No | Yes | Yes | No | Yes | No | No | No | No | No | Yes | user | Yes | No | Unknown | Unknown |
| lighttpd | Yes | Yes | Yes[c][9] | Yes | Yes | Yes | Yes | Yes | No[g] | Yes | No | No | user | No | Yes | No | Unknown |
| LiteSpeed Web Server | Yes | Yes | Yes | Yes | Yes | Yes | Unknown | No | No[g] | Yes | No | Unknown | user | Yes | Yes | Yes | Yes |
| Mongoose | Yes | Yes | Yes | Yes | Yes | No | No | No | No | Yes | No | No | user | Yes | Yes | Unknown | Unknown |
| Monkey HTTP Server | Yes | No | Yes[q] | Yes | Yes | Yes | No | No | No | No | No | No | user | No | Yes | Unknown | Unknown |
| NaviServer | Yes | No | Yes | Yes | Yes | No | Unknown | No | No | No | Unknown | Unknown | user | Yes | Unknown | Unknown | Unknown |
| NCSA HTTPd | Yes | Yes | Unknown | Partial[s] | Yes | Unknown | Unknown | No | No | Yes | No | Unknown | user | Unknown | Unknown | Unknown | Unknown |
| nginx | Yes | Yes (module) | Yes | Yes | No | Yes | Yes | Yes | No[10] | Yes | No | Unknown | user | Yes[11] | Yes[12] | Yes[13] | Unknown |
| OpenLink Virtuoso | Yes | Yes | Yes | Yes | No | No | No | No | Yes | Yes | No | No | user | Yes | No | No | Unknown |
| OpenLiteSpeed | Yes | Yes | Yes | Yes | Yes | Yes | Unknown | No | No | No | No | Unknown | user | Yes | Yes | Yes | Unknown |
| Oracle HTTP Server[14] | Yes | Yes | Yes | Yes | Yes | Yes | Unknown | No | No | Yes | No | Unknown | user | Yes[t] | Yes | Unknown | Unknown |
| Oracle iPlanet Web Server | Yes | Yes | Yes | Yes | Yes | Yes | Unknown | No | Yes | Yes | No | Yes | user | Yes | Yes | Unknown | Unknown |
| thttpd | Yes | Unknown | No | Yes | Yes | No | Unknown | No | No | No | No | Unknown | user | No | Yes | Unknown | Unknown |
| TUX web server | No | No | No | Yes | Yes | No | Unknown | No | No | No | No | Unknown | kernel | Unknown | Unknown | Unknown | Unknown |
| Wakanda Server | Yes | Yes | Yes | Yes | No | No | No | No | No | No | No | Yes | user | Yes | Yes | Unknown | Unknown |
| Xitami | Yes | Unknown | paid version | Yes | Yes | Unknown | Unknown | No | Unknown | Yes | Unknown | Unknown | user | Unknown | Unknown | Unknown | Unknown |
| Yaws | Yes | Unknown | Yes | Yes | Yes | Yes | Unknown | No | No | Yes | No | Unknown | user | Unknown | Yes | Unknown | Unknown |
| Zeus Web Server | Yes | Yes | Yes | Yes | Yes | Yes | Unknown | No | No[g] | Yes | Yes | Unknown | user | Yes | No | Unknown | Unknown |

**Rysunek 2:** Tabela porównawcza wg

`https://en.wikipedia.org/wiki/Comparison_of_web_server_software`

# HTTP i jego metody

# Metody zapytań

Specyfikacja protokółu HTTP/1.0 wyróżnia kilka podstawowych metod komunikacji z serwerem. Wystosowane z poziomu klienta zapytanie ma w nagłówku informację na temat wybranej metody, a wyróżniamy m. in.:

- `GET`
- `POST`
- `PUT`
- `DELETE`
- `OPTIONS`
- i inne

# Metoda `GET`

Podstawowa metoda przeznaczona **tylko i wyłącznie** do pobierania danych.

Lista parametrów przekazanych w zapytaniu dodawana jest do URL-a:

```
http://example.com/index.php?query=Test&country_iso=pl
```

# Metoda `POST`

Podstawowa metoda przeznaczona do przesyłania danych, które mają wpłynąć na system. Serwer oczywiście zwróci odpowiedź w identyczny sposób jak przy metodzie `GET`.

Przykładowo `POST`-em należy wysyłać dane z formularzy czy żądania zmiany statusu.

# RESTful API i etody `PUT`, `PATCH` i `DELETE`

Sprytnie zaprojektowana aplikacja rozróżnia wywołania tego samego URL-a z różnymi metodami. REST-owe API może działać następująco:

- `GET /users` zwróci listę użytkowników
- `PUT /users` podmieni listę użytkowników przesłaną nową listą
- `POST /users` doda nowego użytkownika
- `DELETE /users` usunie listę użytkowników

- `GET /users/1` zwróci użytkownika o danym `id = 1`
- `PUT /users/1` podmieni użytkownika lub go utworzy
- `PATCH /users/1` podmieni użytkownika
- `DELETE /users/1` usunie użytkownika

# Zapytanie i odpowiedź

# Zapytanie

Zapytanie (*Request*) wysyłane przez przeglądarkę oczywiście nie składa się tylko z hosta:

```
GET / HTTP/1.1
Host: pwsz.rewak.pl
User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64; rv:58.0)
        Gecko/20100101 Firefox/58.0
Accept: text/html,application/xhtml+xml,
        application/xml;q=0.9,*/*;q=0.8
Accept-Language: pl,en-US;q=0.7,en;q=0.3
Accept-Encoding: gzip, deflate
Cookie: _ga=GA1.2.251979732.1506974412;
        _gid=GA1.2.724276722.1520606158
Connection: keep-alive
Upgrade-Insecure-Requests: 1
```

# Odpowiedź

Natomiast odpowiedź zawsze zawiera w sobie nagłówek (*Response header*) oraz zawartość (*Response body*). Nagłówek wygląda zazwyczaj jak poniżej:

```
Accept-Ranges: bytes
Connection: Keep-Alive
Content-Encoding: gzip
Content-Length: 432
Content-Type: text/html; charset=UTF-8
Date: Sun, 11 Mar 2018 10:02:19 GMT
ETag: "2a4-55b3fcc244d5e-gzip"
Keep-Alive: timeout=5, max=100
Last-Modified: Wed, 11 Oct 2017 06:47:29 GMT
Server: Apache/2.4.18 (Ubuntu)
Vary: Accept-Encoding
```

# Odpowiedź

Zawartość to oczywiście HTML strony internetowej:

```
<!DOCTYPE html>
<html>
<head>
   <meta charset="utf-8">
   <meta http-equiv="X-UA-Compatible" content="IE=edge">
   <meta name="viewport"
     content="width=device-width,initial-scale=1">
   <title>Krzysztof Rewak</title>
   <link rel="stylesheet"
     href="/static/css/app.c0927d6b397cd98ad145970e94ff4c51.css">
</head>

(...)
```

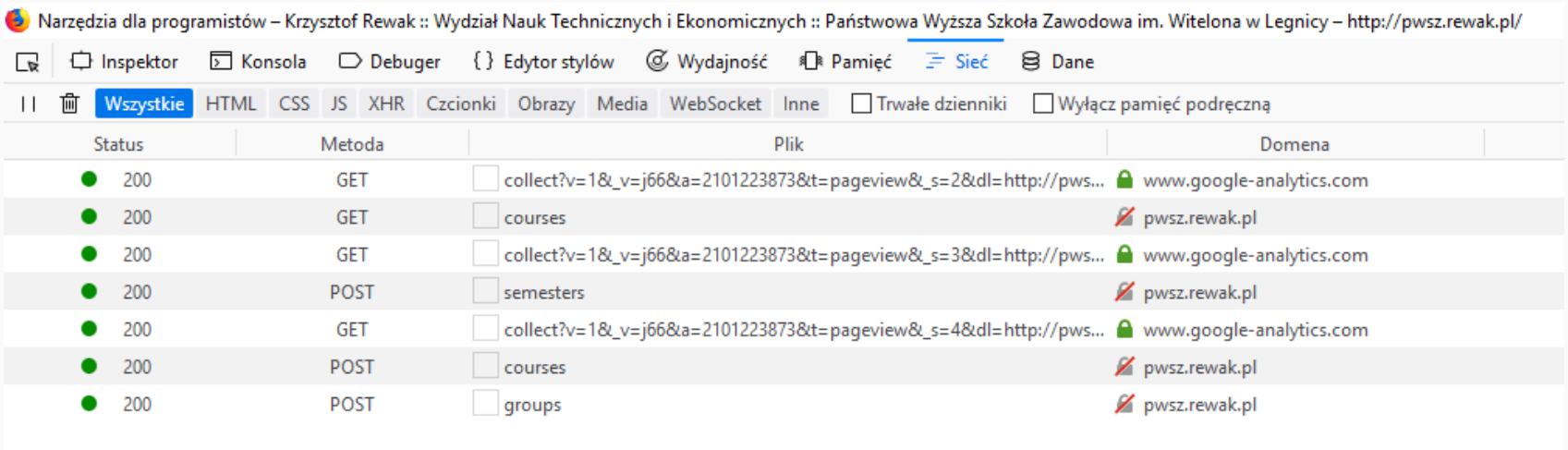
# Kody statusów

# Kody statusów

Każda odpowiedź serwera ma przypisany do siebie specjalny kod statusu, który powinien powiedzieć przeglądarce i użytkownikowi, co stało się z jego zapytaniem.

Kody są pogrupowane według typu odpowiedzi i można je szybko zidentyfikować po pierwszej cyfrze trzycyfrowego kodu.

# Kody statusów




**Rysunek 3:** Narzędzia deweloperskie mocno pomagają przy badaniu komunikacji z serwerem

# Przykładowe kody statusów

- **1xx** informacje zwrotne
- **2xx** informacje o sukcesie:
  - **200** OK
  - **201** Created
  - **202** Accepted
- **3xx** informacje o przekierowaniu
- **4xx** informacje o błędzie po stronie klienta:
  - **400** Bad Request
  - **401** Unauthorized
  - **403** Forbidden
  - **404** Not Found
  - **405** Method Not Allowed
- **5xx** informacje o błędzie po stronie serwera:
  - **500** Internal Server Error
  - **501** Not Implemented
  - **503** Service Unavailable
  - **504** Gateway Timeout

# Routing

# Najprostsza aplikacja w PHP

Najprymitywniejsza aplikacja napisana w PHP mogłaby mieć taką strukturę:

- `styles/`
  - `style.css`
- `dashboard.php`
- `index.php`
- `login.php`
- `logout.php`
- `register.php`

Jeżeli Apache lub Nginx wskazuje folder projektu, wówczas wystarczy w pasku adresu wpisać `localhost/login.php`, aby otworzyć stronę do logowania.

# Najprostsza aplikacja w PHP

Jest to pomysł przynajmniej beznadziejny.

# Router systemów internetowych

Router jest częścią aplikacji, która odpowiada za przypisanie konkretnych akcji do wywoływanych adresów URL.

Najczęstsza konfiguracja to wskazanie serwerowi, aby zawsze przekierowywał domenę na jeden plik (`index.php`) lub serwis (WSGI). Wówczas już aplikacja zajmie się rozprowadzaniem zadań, a nie serwer HTTP.

# Router w Django

```
from django.urls import path
from app import views

urlpatterns = [
    path('/', views.get_index),
    path('/faq', views.get_faq),
    path('/about', views.get_about),
    path('/login', views.get_login),
    path('/user/<ind:id>', views.get_user_page),
]
```

# Router w Laravelu

```
<?php

Route::get('/', 'HomeController@getIndex');
Route::get('/faq', 'HomeController@geFaq');
Route::get('/about', 'HomeController@getAbout');
Route::get('/login', 'LoginController@get');
Route::post('/login', 'LoginController@post');
Route::get('/user/:id', 'UserController@getUserPage');
```

# Router w ASP.NET MVC

```
public class HomeController : Controller
{
        [Route('/')]
        public ActionResult View()
        {
                return View('Home');
        }

        [Route('about')]
        public ActionResult View()
        {
                return View('About');
        }
}
```

# Podsumowanie

# Bibliografia i ciekawe źródła


- https://w3techs.com/technologies/overview/web_server/all
- https://en.wikipedia.org/wiki/Hypertext_Transfer_Protocol
- https://en.wikipedia.org/wiki/List_of_HTTP_status_codes
- https://en.wikipedia.org/wiki/List_of_HTTP_header_fields

# 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`