



PROGRAMOWANIE APLIKACJI INTERNETOWYCH

Wykład 1, część 1

Architektura aplikacji internetowej współpracującej z bazą danych. Metody protokołu HTTP. Nagłówki żądania i odpowiedzi HTTP. PHP jako przykład języka skryptowego działającego po stronie serwera. Podstawowe elementy języka.

Beata Pańczyk



Cele przedmiotu i wymagania wstępne

- **Cele przedmiotu:**
 - Zapoznanie studentów ze specyfiką programowania aplikacji internetowej współpracującej z bazą danych z uwzględnieniem aspektów bezpieczeństwa na przykładzie PHP i MySQL
 - Nabycie przez studentów umiejętności projektowania i programowania aplikacji internetowych w języku PHP z bazą danych MySQL
- **Wymagania wstępne w zakresie wiedzy, umiejętności i innych kompetencji:**
 - Znajomość podstaw aplikacji internetowych
 - Znajomość algorytmów i struktur danych
 - Umiejętność programowania obiektowego
- **Warunki zaliczenia:** egzamin w postaci testu wielokrotnego wyboru z treści programowych (pytania dotyczą każdego z obszarów W1-W10 z taką samą wagą w ocenie końcowej)

Tematyka wykładów (1)

<u>W1</u>	Architektura aplikacji internetowej współpracującej z bazą danych. Metody protokołu HTTP. Nagłówki żądania i odpowiedzi HTTP. PHP jako przykład języka skryptowego działającego po stronie serwera. Podstawowe elementy języka. Metody pobierania danych z formularzy HTML.
<u>W2</u>	Typy danych w języku PHP. Tablice asocjacyjne. Operacje na plikach. Ciągi i wyrażenia regularne.
<u>W3</u>	Funkcje i szablony w PHP. Interakcja aplikacji internetowej z systemem plików na serwerze WWW.
<u>W4</u>	Programowanie obiektowe w PHP. Klasy i obiekty. Definiowanie składowych klasy. Metody i typy argumentów w metodach. Zarządzanie dostępem do klasy. Dziedziczenie. Metody i składowe statyczne. Klasy abstrakcyjne i interfejsy. Obsługa wyjątków.
<u>W5</u>	Współpraca aplikacji internetowej z bazami danych. Podstawowe zapytania SQL. Serwer MySQL i narzędzie administrowania bazami danych PHPMyAdmin. PHP i MySQL - interfejs mysqli i PDO.

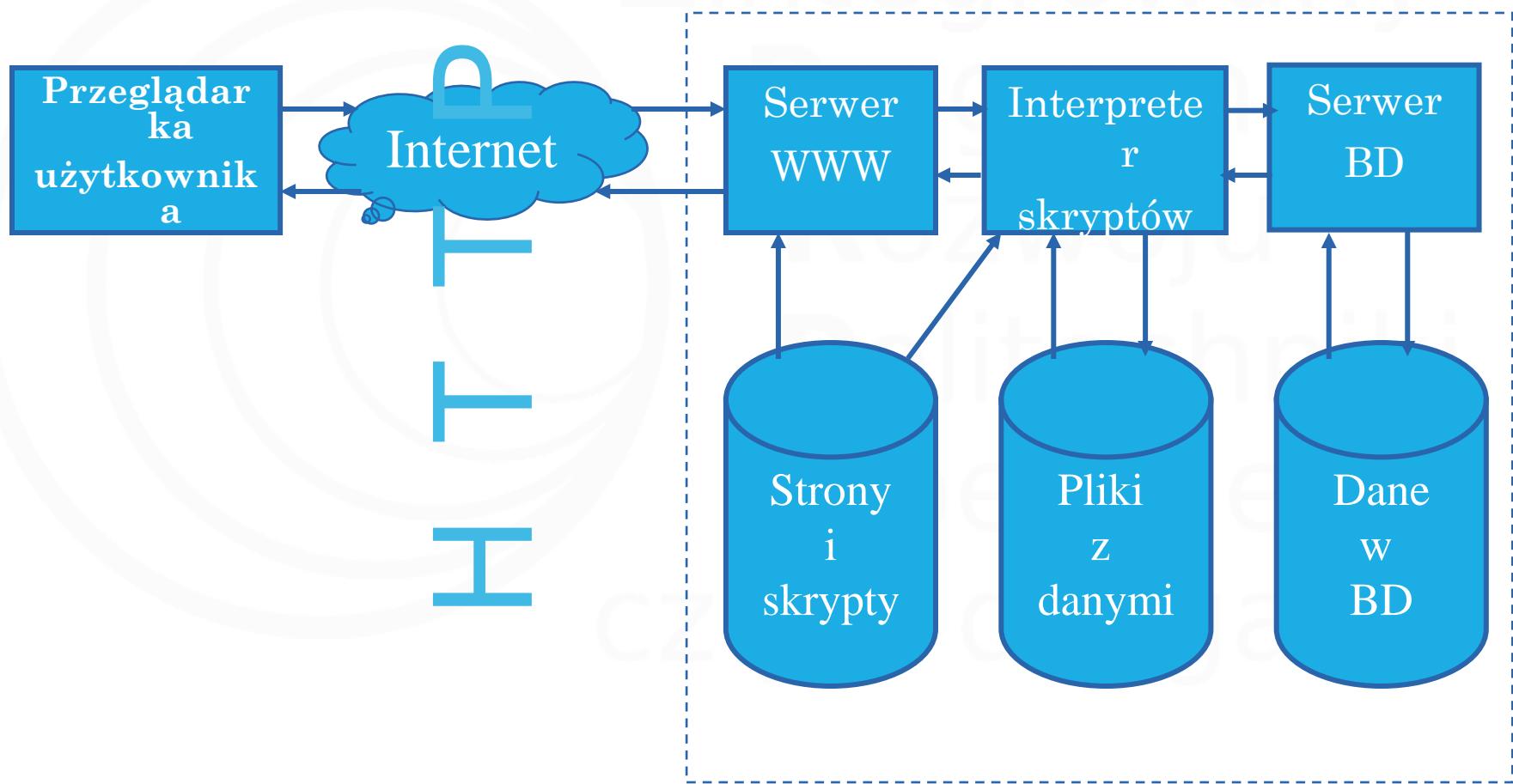
Tematyka wykładów (2)

<u>W6</u>	Zagrożenia transakcji internetowej. Szyfrowanie i certyfikaty cyfrowe. HTTP jako protokół bezstanowy. Pojęcie, zasada działania i funkcje sesji. Uwierzytelnianie w kontroli sesji.
<u>W7</u>	Strategia tworzenia bezpiecznych aplikacji internetowych. Rodzaje ataków sieciowych. Metody zabezpieczania aplikacji i ich implementacja w PHP.
<u>W8</u>	Model modułowej aplikacji internetowej na przykładzie strony fikcyjnej firmy.
<u>W9</u>	Projektowanie obiektowe. Wprowadzenie do wzorców projektowych. Elementy języka UML. Aplikacje wielowarstwowe i wzorzec projektowy MVC. Modele, widoki, kontrolery i ich implementacja w PHP.
<u>W10</u>	Transmisja danych z serwera w trybie asynchronicznym – dynamiczna interakcja z użytkownikiem za pomocą obiektu Ajax, jQuery i PHP.
<u>W11</u>	Przegląd aktualnych platform programistycznych do tworzenia aplikacji internetowych. Wstęp do mapowania obiektowo relacyjnego (ORM).

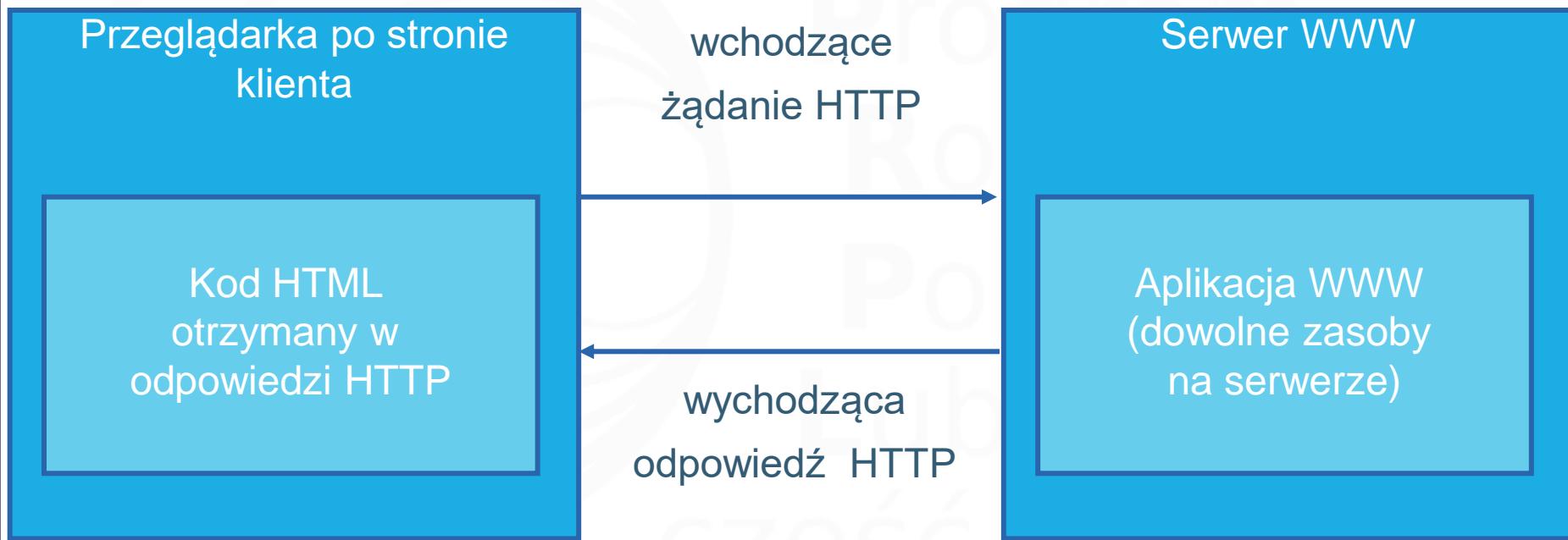
Bibliografia

- Nixon R., PHP, MySQL i JavaScript. Wprowadzenie. Helion, 2019
- <https://www.php.net/manual/en/index.php>, dokumentacja i podręcznik PHP
- <http://www.w3schools.com/php/default.asp>
- Jędrych S., Jędruszak B., Pańczyk B., Tworzenie aplikacji internetowych na platformie JEE i PHP – analiza porównawcza, Journal of Computer Sciences Institute, 11, 86-90, <https://ph.pollub.pl/index.php/jcsi/article/view/145>
- Pańczyk B., Ślawiński A., Technologie mapowania obiektowo-relacyjnego w aplikacjach PHP, Informatyka, Automatyka, Pomiary W Gospodarce I Ochronie Środowiska, 2015, nr 1, vol. 5, s. 29-32, [https://www.researchgate.net/publication/281429862 TECHNOLOGIE MAPOWANIA OBIEKTOWO-RELACYJNEGO W APLIKACJACH PHP](https://www.researchgate.net/publication/281429862)

Środowisko aplikacji internetowej współpracującej z bazą danych



Cykl żądania i odpowiedzi HTTP



Serwer WWW

- Serwer WWW (Web server) - program działający na serwerze internetowym, obsługujący żądania protokołu komunikacyjnego HTTP
- Serwer WWW może korzystać z usług innego, równolegle działającego oprogramowania, np. MySQL i PHP
- Najbardziej popularne serwery WWW:
 - Apache
 - Microsoft
 - nginx
 - serwery aplikacji Java EE: Apache Tomcat, GlassFish

Metody protokołu HTTP

- Metoda HTTP jest poleceniem lub żądaniem wysyłanym przez klienta (przeglądarkę) do serwera
- Metody protokołu HTTP:
 - GET - klient chce otrzymać zasób (plik, dane wyjściowe programu itp.) z serwera
 - HEAD - klient chce uzyskać tylko informacje o dokumencie (sam dokument nie jest mu potrzebny)
 - POST - klient udostępnia/wysyła jakieś informacje (np. z formularza) do serwera, a serwer może je wykorzystać np. zapisać w bazie danych
 - PUT - wykorzystuje się do przesyłania danych w celu modyfikacji zasobu już istniejącego na serwerze
 - DELETE – klient prosi o usunięcie zasobu z serwera

Metody GET i POST protokołu HTTP

- **GET** - w metodzie tej parametry żądania HTTP można przekazywać **jedynie poprzez dołączenie ich do adresu URL**, np. `http://localhost/index.php?par1=wart1&par2=wart2`
- **POST** - również reprezentuje żądanie klienta HTTP, lecz do żądania tego zwykle dołączone jest ciało (ang. **body**), które reprezentuje dane **wysyłane przez klienta** do serwera HTTP (np. parametry, pliki), a w związku z obecnością ciała żądania, komunikat sieciowy zawiera pola nagłówkowe:
 - **Content-Type**: `application/x-www-form-urlencoded` (domyślnie)
 - **Content-Length** (wielkość przesyłanych danych w bajtach)

Jezyki aplikacji webowej

- Aplikacje webowe mogą być napisane w wielu językach (np. Ruby, Python, PHP, Java, C#). Języków jest bardzo dużo, jednak **mechanizm działania jest zawsze ten sam**
- Aplikacja webowa napisana w języku X interpretuje żądanie wysłane przez przeglądarkę użytkownika do serwera i odpowiada na nie generując odpowiednią zawartość (plik generowany dynamicznie, który jest zrozumiały dla przeglądarki)
- Język, który rozumie przeglądarka: HTML, CSS, JavaScript

Formularze HTML – podstawa interfejsu użytkownika

```
<form action="index.php" method="GET">  
<table>  
    <tr> <td> Nazwisko: </td>  
        <td><input name="nazwisko" /></td> </tr>  
    <tr> <td> Średnia: </td>  
        <td> <input name="srednia" /></td> </tr>  
</table>  
<input type="reset" name="reset"  
       value="Wyczyść formularz" />  
<input type="submit" name="wyslij" value="Wyslij" /><br />  
</form>
```

Przesyłanie danych z formularzy

URL w GET:

http://localhost/index.php?nazwisko=Nowak&srednia=4.76&wyslij=Wyslij

URL w POST:

http://localhost/index.php

i dodatkowo nagłówki
oraz ciało żądania:

Nazwisko: Nowak

Średnia: 4.76

Wyczyść formularz

Wyslij

HTTP Headers

Content-Type: application/x-www-form-urlencoded

Content-Length: 41

body

nazwisko=Nowak&srednia=4.76&wyslij=Wyslij

PHP - przykładowy język programowania po stronie serwera

- PHP ("PHP: Hypertext Preprocesor") - twórca Rasmus Lerdorf (1994)
- Skrypt PHP - niezależny od systemu operacyjnego i od rodzaju przeglądarki, zależy jedynie od oprogramowania na serwerze

PHP oferuje:

- możliwość wyprowadzania na wyjście dowolnych danych tekstowych (HTML, XML, JSON), plików pdf, zasobów graficznych itp.
- możliwość pracy z plikami i katalogami w lokalnym systemie plików lub zdalnie
- obsługę wielu rodzajów baz danych poprzez interfejsy dedykowane konkretnym bazom (np. Oracle, MySQL) lub uniwersalny interfejs PDO
- udostępnia funkcje do haszowania i szyfrowanie danych
- i wiele innych możliwości użytkowych w aplikacjach internetowych

Co jest potrzebne?

- Do pisania skryptów po stronie serwera w PHP potrzebujemy: PHP, serwera WWW i przeglądarki (klienta)
- Kod źródłowy PHP do pobrania ze strony:
<http://www.php.net/downloads.php> (stabilna wersja na czerwiec 2021 to 8.0.7)
- Szczegóły instalacji dla poszczególnych systemów operacyjnych i serwerów można znaleźć w podręczniku:
<https://www.php.net/manual/en/install.php>
- Alternatywą jest skorzystanie z gotowego, skonfigurowanego już do pracy środowiska (Apache, PHP i MySQL) po instalacji pakietu XAMPP (na wybrany system operacyjny):
<https://www.apachefriends.org/pl/download.html>

Apache i PHP w XAMPP

W celu uruchomienia skryptu PHP należy:

- uruchomić serwer Apache
- umieścić folder z projektem w katalogu **htdocs** w instalacji serwera Apache (może to być inny folder wskazany w ustawieniach konfiguracyjnych serwera jako **DocumentRoot**)
- wskazać skrypt do uruchomienia przeglądarce wpisując adres np.:
localhost:80/projekt/skrypt.php
lub
127.0.0.1/projekt/skrypt.php

Ważne pliki konfiguracyjne:

- **httpd.conf** zawiera ustawienia konfiguracyjne serwera np.:
ServerRoot "C:/Program Files/xampp/apache"
DocumentRoot "C:/Program Files/xampp/htdocs"
Listen 80
- **php.ini** opisuje konfigurację PHP

PHP - zaczynamy

- Pierwsze skrypty PHP rozpatrujemy jako fragmenty kodu zagnieżdżone w HTML. Wszystkie części takiego skryptu PHP muszą być zamknięte znacznikami początku i końca PHP:
`<?php ... ?>`

- Kod źródłowy dokumentu ze skryptem **musi mieć** rozszerzenie **.php**
- Komentarze:

```
// komentarz jednoliniowy  
# komentarz jednoliniowy  
/* komentarz  
   wieloliniowy */
```

- wyświetlanie informacji:
`<?php echo "<p> PHP - zaczynamy</p>"; ?>`
`<?php echo '<p> "PHP" - zaczynamy</p>'; ?>`
`<?php print("<p> PHP - zaczynamy</p>"); ?>`
`<?php print '<p> "PHP" - zaczynamy</p>'; ?>`

PHP - zaczynamy

"PHP" - zaczynamy

PHP - zaczynamy

"PHP" - zaczynamy

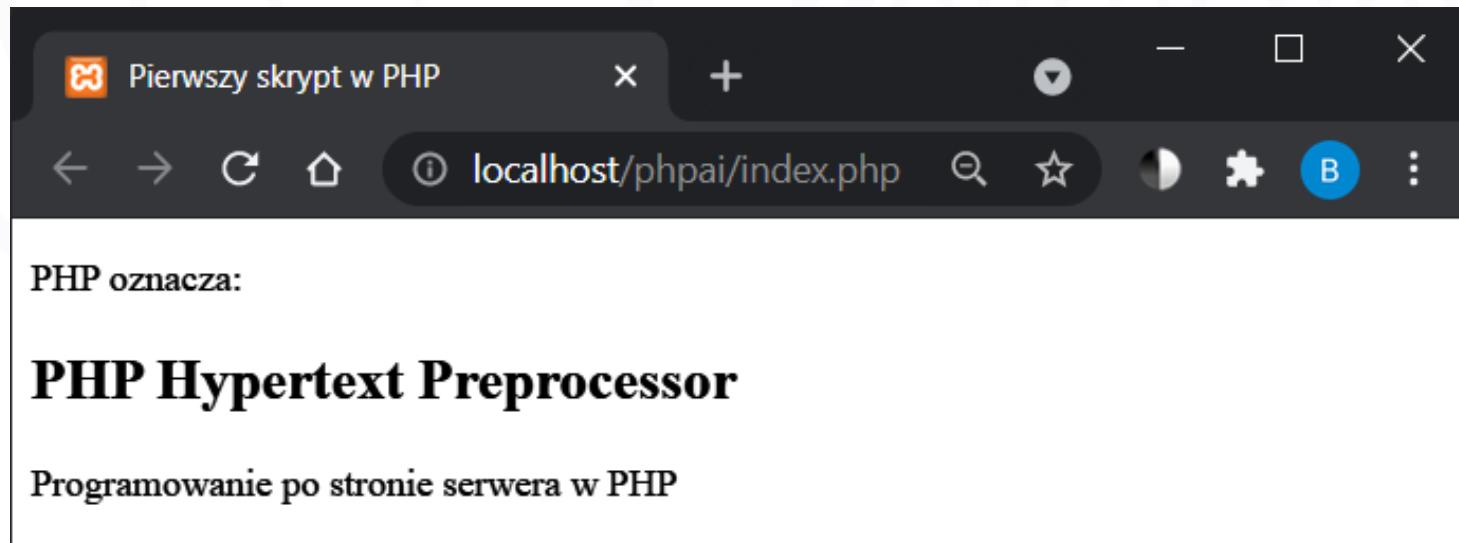
PHP w dokumencie HTML

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title> Pierwszy skrypt w PHP</title>
  </head>
  <body>
    <p>PHP oznacza:</p>
    <?php
      //początek kodu PHP (komentarz w skrypcie)
      print("<h2>PHP Hypertext Preprocessor</h2>");
    ?> <!-- koniec kodu PHP – komentarz HTML
          poza skryptem -->
    <p> Programowanie po stronie serwera w PHP </p>
  </body>
</html>
```

Widok w przeglądarce

Kod źródłowy widoczny w przeglądarce (Ctrl+U):

```
<!DOCTYPE html><html> <head> ... </head>
<body>
<p>PHP oznacza:</p>
<h2>PHP Hypertext Preprocessor</h2>
<!-- koniec kodu PHP – komentarz HTML poza skryptem -->
<p> Programowanie po stronie serwera w PHP </p></body>
</html>
```



Identyfikatory, typy zmiennych, stałe

- Identyfikatory zmiennych PHP rozpoczynają się zawsze od znaku \$
- PHP stosuje typowanie dynamiczne (inaczej słabe typowanie)
- Typy danych w PHP: **Integer, Flout, Double, String, Boolean , Array, Object, NULL** (typ specjalny) np.:
`$wartosc=1.3; $wartosc="PHP";`
- Rzutowanie typu, np.:
`$ilosc=0; $wartosc=(double)$ilosc;`
- Zmienne zmiennych - użycie wartości jednej zmiennej jako nazwy drugiej np.:
`$nazwa="ilosc"; $$nazwa=15; // $ilosc=15;`
- Stałe (oznaczane umownie dużymi literami), np.:
`define("CENA", 100); echo CENA;`

Instrukcje i operatory

- Podstawowe instrukcje i operatory - większość jak w C
- Operator łączenia ciągów - znak **kropki**, np.:
`$a="PHP"; $b="-operatory"; $c=$a.$b;`
//lub:
`$c=$a.-operatory;`
//lub:
`$c="$a-operatory";`
- Operator referencji: **&**
np. `$a=5; $b=$a; $a=10; //a=10, b=5`
 `$a=5; $b=&$a; $a=10; //a=10, b=10`
- **==** (operator identyczności) - zwraca true, jeśli operandy są równe co do wartości i typu
- **@** (operator tłumienia błędów - np. `$a=@(5/0);` bez operatora @ wygenerowane będzie ostrzeżenie o dzieleniu przez 0, z @ błąd zostanie stłumiony, ale należy go obsłużyć samodzielnie

Skrypt PHP do generowania tabeli

```
<table style="border:1px solid black">
<tr> <th> Odległość w km </th>
      <th> Koszt w PLN </th>
</tr>
<?php
$d = 50;
while ($d <= 350) {
    echo "<tr><td> $d </td>";
    echo "<td>".($d/10)."</td>";
    echo "</tr>";
    $d += 50;
}
?>
</table>
```

Odległość w km	Koszt w PLN
50	5
100	10
150	15
200	20
250	25
300	30
350	35

POLITECHNIKA LUBELSKA

WYDZIAŁ ELEKTROTECHNIKI I INFORMATYKI

INFORMATYKA



Zintegrowany
Program
Rozwoju
Politechniki
Lubelskiej -
część druga

PROGRAMOWANIE APLIKACJI INTERNETOWYCH

Wykład 1, część 2

Metody pobierania danych z formularzy HTML

Beata Pańczyk



Fundusze
Europejskie
Wiedza Edukacja Rozwój



Rzeczpospolita
Polska

Unia Europejska
Europejski Fundusz Społeczny



Odbieranie parametrów żądania

- Atrybuty elementu <form>:
`<form method="post/get" action="http://localhost/phpai/wyniki.php">` informują, że dane pobrane z formularza mają być wysłany do skryptu **wyniki.php**, który znajduje się na serwerze lokalnym w głównym folderze projektu o przykładowej nazwie **phpai**.
- **wyniki.php** - skrypt PHP odbierze parametry przesłane w żądaniu HTTP zebrane z formularza (domyślnie przesłane w postaci par name=value)
- dane przesłane w żądaniu są dostępne w skrypcie PHP za pośrednictwem specjalnych asocjacyjnych tablic globalnych (**\$_GET**, **\$_POST**, **\$_COOKIE**), które otrzymują dane przesłane z żądania (odpowiednią metodą protokołu HTTP) i przechowują je w postaci par klucz=wartość
- tablica **\$_REQUEST** zawiera dane parametry przekazane metodą GET i POST oraz ciasteczka (cookie), jest zbiorem wszystkich elementów z tablic **\$_GET**, **\$_POST** i **\$_COOKIE**

Przykładowy formularz

The screenshot shows a web browser window with the URL `localhost/phpai/form1.html` in the address bar. The page content is as follows:

Dane osobowe:

Nazwisko: Pańczyk

Imię: Beata

Adres e-mail: beata@gmail.com

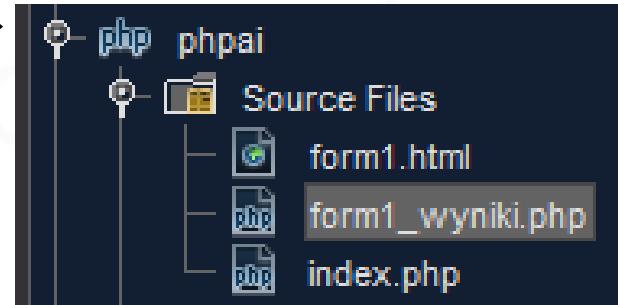
Proszę zaznaczyć zamawiane produkty:

PHP C++ Java

Wyslij **Anuluj**

Plik form1.html

```
<body>
<div><h4>Dane osobowe:</h4>
<form method="get" action="/phpai/form1_wyniki.php">
<p> Nazwisko: <input name="nazw" /><br/>
    Imię: <input name="imie" /><br/>
    Adres e-mail:<input name="email" /></p>
<h4>Proszę zaznaczyć zamawiane produkty:</h4>
<p>
    <input name="php" type="checkbox"/>PHP
    <input name="cpp" type="checkbox"/>C++
    <input name="java" type="checkbox"/>Java <br/>
    <input type="submit" value="Wyslij"/>
    <input type="reset" value="Anuluj"/>
</p>
</form></div>
</body>
```



Efekt działania skryptu odczytującego dane z formularza

← → C ⌂ ⓘ localhost/phpai/form1_wyniki.php?nazw=Pańczyk&i

Poniżej znajdują się dane odebrane z formularza:

Nazwisko: Pańczyk

Imię: Beata

Adres e-mail:b.panczyk@pollub.pl

Zamawiane produkty:

- PHP
- C++
- Java

Skrypt form1_wyniki.php

```
<body>
<h4>Poniżej znajdują się dane odebrane z formularza:</h4>
<?php
    $nazw = $_GET['nazw'];    print("Nazwisko: $nazw ");
    print("<br />Imię: " . $_GET['imie']);
    print("<br />Adres e-mail:" . $_GET['email']);
?
<h4>Zamawiane produkty:</h4>
<?php
    if (isset($_GET['php'])) print("- PHP<br />");
    if (isset($_GET['cpp'])) print("- C++<br />");
    if (isset($_GET['java']))print("- Java<br />");
?
</body>
```

Metoda GET i POST w formularzu

- Adres URL z poprzedniego slajdu ma postać:
localhost/form1_wyniki?nazw=Pańczyk&imie=Beata...&php=on&cpp=on&java=on
- Pozycje po znaku zapytania w adresie URL są nazwami i wartościami zmiennych przesyłanymi z formularza do serwera, oddzielone od siebie znakiem &
- Gdy dane są przesyłane metodą **GET**, skrypt PHP czytając adres URL, automatycznie dodaje te pary do tablicy asocjacyjnej **\$_GET** (oraz **\$_REQUEST**)
- Używając metody **POST** dane są dołączane **do ciała żądania** i nie są widoczne w adresie URL, a skrypty PHP mają do nich dostęp za pomocą tablicy asocjacyjnej **\$_POST** (oraz **\$_REQUEST**)

Przekazywanie danych przez formularze i łącza

Pliki do realizacji przykładu:

- **form2.html** – formularz HTML, za pomocą którego przesyłane są informacje wprowadzone przez użytkownika
- skrypt **form2_linki.php** - odbiera dane z formularza HTML i tworzy trzy hiperłącza, które prowadzą do kolejnych skryptów z przekazaniem odpowiednich porcji danych:
 - **form2_kontakt.php** - dostaje i wyświetla tylko informacje kontaktowe
 - **form2_firm.php** - dostaje i wyświetla tylko informacje dotyczące firmy
 - **form2_hobby.php** - dostaje i wyświetla wyłącznie informacje o hobby

Widok formularza form2.html

← → C ⌂ ⓘ localhost/phpai/form2.html

Informacje kontaktowe:

Nazwisko: Brzęczyszczykiewicz

Imię: Grzegorz

Telefon: 567567567

Informacje zawodowe:

Nazwa firmy: Film Co.

Telefon służbowy: 123123123

Zainteresowania:

sport muzyka film

Wyslij **Anuluj**

← → C ⌂ ⓘ localhost/phpai/form2_linki.php

Łącza utworzone na podstawie danych użytkownika:

[Informacje kontaktowe](#)

[Informacje służbowe](#)

[Ulubione hobby](#)

Plik z formularzem - form2.html

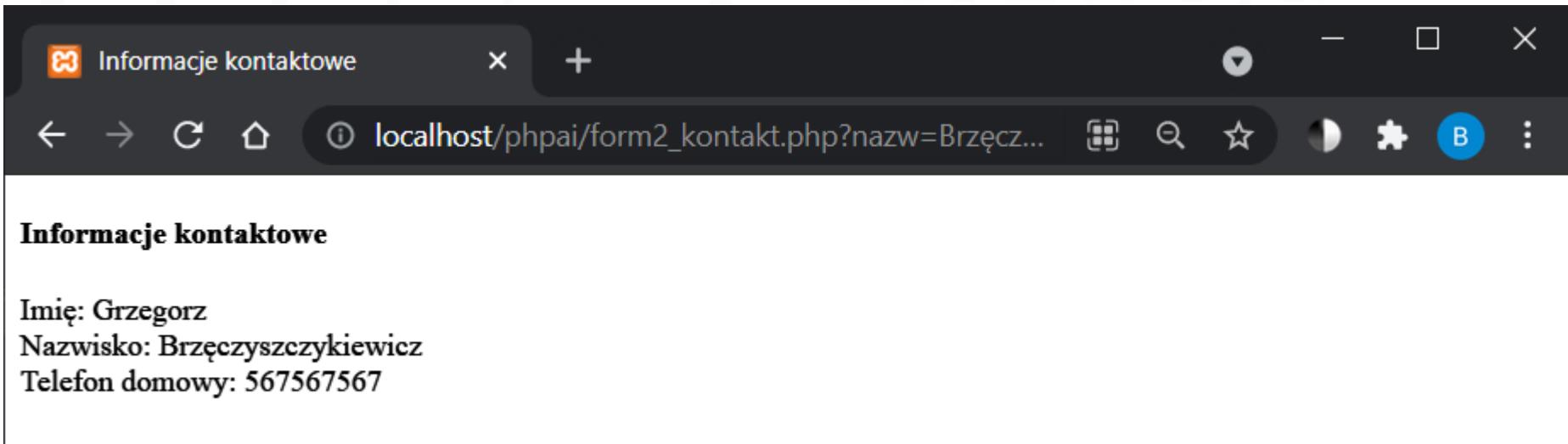
```
<body>
  <h4>Informacje kontaktowe:</h4>
  <form method="post" action="form2_linki.php">
    <p> Nazwisko: <input name="nazw" /><br/>
        Imię: <input name="imie" /><br/>
        Telefon:<input name="telp" /></p>
    <h4>Informacje zawodowe:</h4>
    <p> Nazwa firmy: <input name="firma" /><br/>
        Telefon służbowy:<input name="tels" /></p>
    <h4>Zainteresowania:</h4>
    <p> sport<input type="radio" name="hobby" value="sport"/>
        muzyka<input type="radio" name="hobby" value="muzyka"/>
        film<input type="radio" name="hobby" value="film"/>
        <br/>
        <input type="submit" value="Wyslij"/>
        <input type="reset" value="Anuluj"/></p>
  </form>
</body>
```

Skrypt form2_linki.php

```
<h4>Łącza utworzone na podstawie danych użytkownika:</h4>
<?php
    $nazw = $_POST['nazw']; $imie = $_POST['imie'];
    $telp = $_POST['telp']; $firma = $_POST['firma'];
    $tels = $_POST['tels']; $hobby = $_POST['hobby'];
    /* utworzenie łącza do form2_kon.php z równoczesnym
     * przekazaniem wartości zmiennych nazw, imie, telp
     * pobranych z formularza form2.html i dołączonych
     * jako parametry do adresu URL w hiperłączu */
    print("<p><a href=
        'form2_kontakt.php?nazw=$nazw&imie=$imie&telp=$telp'>");
    print("Informacje kontaktowe</a>"); 
    print("<p><a href=
        'form2_firma.php?firma=$firma&tels=$tels'>");
    print("Informacje służbowe</a>"); 
    print("<p><a href='form2_hobby.php?hobby=$hobby'>"); 
    print("Ulubione hobby</a>"); 
?>
```

Skrypt form2_kontakt.php

```
<h4>Informacje kontaktowe </h4>
<p> Imię: <?php echo $_GET['imie']; ?> <br/>
    Nazwisko: <?php echo $_GET['nazw']; ?> <br/>
    Telefon prywatny: <?php echo $_GET['telp']; ?>
</p>
```



Skrypt form3.php generujący i obsługujący prosty formularz

```
<body>
<?php
    //skrypt generuje formularz i jednocześnie
    //odbiera dane z niego wysłane
    if (isset($_POST['tekst'])) //przesłano żądanie z parametrem 'tekst'
    {
        $tekst=$_POST['tekst'];
        print "Wpisano: $tekst <br/>";
        print "<a href='form3.php'> Powrót do formularza</a>";
    }
    else //nie przesłano danych z formularza - w żądaniu nie
        //ma parametru o kluczu 'tekst' - wyświetl formularz
    {
        print "Podaj tekst :<form method='post'
                action='form3.php'>";
        print "<input type='tekst' name='tekst' size='30' />";
        print "<input type='submit' value='Wyślij' />";
        print "</form>";
    }
?> </body>
```

Przykładowe wyniki działania

← → ⌛ ⌂ ⓘ localhost/phpai/form3.php

Podaj tekst :

Zwykły tekst

← → ⌛ ⌂ ⓘ localhost/phpai/form3.php

Wpisano: Zwykły tekst
[Powrót do formularza](#)

← → ⌛ ⌂ ⓘ localhost/phpai/form3.php

Podaj tekst :

<h2>Tekst z tagiem HTML

← → ⌛ ⌂ ⓘ localhost/phpai/form3.php

Wpisano:
Tekst z tagiem HTML
[Powrót do formularza](#)

Wynik działania z tekstem JavaScript

The screenshot shows a web browser window with the URL `localhost/phpai/form3...`. In the input field, the user has typed the following JavaScript code:

```
<script>alert("ATAK CSS")</script>
```

Next to the input field is a button labeled "Wyślij". Below the input field, a modal dialog box is displayed with the title "Komunikat ze strony localhost". Inside the dialog, the text "ATAK CSS" is shown, which was output by the executed JavaScript. A blue "OK" button is at the bottom right of the dialog. At the bottom of the page, there is some server-side code output:7 <body>
8 Wpisano: <script>alert("ATAK CSS")</script>

Powrót do formularza</body>

Bezpieczne wyświetlanie danych

- Podstawowe zabezpieczenie – kontrola znaków specjalnych HTML w przychodzących danych
- W PHP najprostsze zabezpieczenie realizuje funkcja **htmlspecialchars()**, która zamienia znaki specjalne HTML na kody „bezpieczne” do wyświetlania na stronach WWW
- W przykładzie należy wprowadzić jedną modyfikację:
\$tekst = **htmlspecialchars(\$_POST['tekst']);**

Wpisano: <script>alert("ATAK CSS")</script>
[Powrót do formularza](#)

```
<body>
    Wpisano: &lt;script&gt;alert(&quot;ATAK
CSS&quot;)&lt;/script&gt; <br/><a href='form3.php'> Powrót do
formularza</a></body>
```

Użyteczne funkcje do operacji na ciągach

- **trim()** - usuwa pustą przestrzeń (\n, \r, \t, \v, \0, \s), z początku i końca ciągu (lub podane w drugim parametrze znaki) i zwraca串 wynikowy, np.: \$nazwa= " a\nb\tc "; \$nazwa = trim(\$nazwa);
- **Itrim(), chop()** - j.w. ale tylko z początku/końca ciągu
- **nl2br()** - zamienia w ciągu wszystkie znaki końca linii na

- **strtoupper(), strtolower()** - zamiana liter w ciągu na wielkie/małe
- **ucfirst()** - zamiana na wielką pierwszej litery ciągu, jeżeli jest literą
- **ucwords()** - zamiana na wielką pierwszej litery każdego wyrazu ciągu, jeżeli jest ona literą
np.: \$tekst="Język PHP"; strtoupper(\$tekst);
- **echo** - polecenie do wyświetlania tekstu w przeglądarce
- **print()** - funkcja, która działa jak echo, ale zwraca wartość 0 lub 1
- **printf()** - wyświetla sformatowany串 w przeglądarce (odpowiednik funkcji z języka C), np.:
\$x=1234.56789; printf("Wartość zamówienia wynosi %.2f", \$x);



PROGRAMOWANIE APLIKACJI INTERNETOWYCH

Wykład 2

Typy danych w języku PHP. Tablice asocjacyjne. Operacje na plikach. Ciągi i wyrażenia regularne.

Beata Pańczyk



Podstawowe typy danych w PHP

- Typy proste: INTEGER, DOUBLE, STRING i BOOLEAN
- Tablice indeksowane numerycznie np.:
`$towar=array("Zeszyt","Blok","Kredki");`
`$towar=["Zeszyt","Blok","Kredki"];`
- Tablice asocjacyjne, w których dostęp do elementów jest możliwy za pomocą klucza. W tablicach asocjacyjnych przechowywane są elementy w postaci pary klucza i wartości, np.:
`$ceny=["Zeszyt"=>2, "Blok"=>8, "Kredki"=>4];`
 - dostęp do elementów tablicy:
`$ceny["Blok"]=6;`
 - indeksy tablicy nie są liczbami, ale kluczami typu String, a dostęp do elementów tablicy możliwy jest za pomocą specjalnej pętli foreach

Konstrukcja foreach

- Iteracyjne przetwarzanie tablic asocjacyjnych:
**foreach (\$tab as \$klucz=>\$wartosc) { echo "\$klucz = \$wartosc
"; }**
- Pętla foreach operuje na kopii zmiennej \$tab, więc w celu modyfikacji wartości elementów tablicy w pętli trzeba skorzystać z operatora referencji &, np.:
**foreach (\$tab as \$klucz=>&\$wartosc) {
 //modyfikacja wartości
}**
- Wyświetlanie danych z tablicy asocjacyjnej, np.:
**\$ceny=["Zeszyt"=>2, "Blok"=>8, "Kredki"=>4];
foreach (\$ceny as \$klucz=>\$wartosc)
{
 print (' \$ceny['.\$klucz.']='.\$wartosc. '
');
}**

```
$ceny[Zeszyt]=2  
$ceny[Blok]=8  
$ceny[Kredki]=4
```

Sortowanie tablic

- Tablice numeryczne:

sort() - sortowanie rosnące

rsort() - sortowanie malejące

np.:

```
$ciag=array(20,10,18); sort($ciag);
```

- Tablice asocjacyjne:

asort() - sortowanie rosnące wg. wartości elementu

ksort() - sortowanie rosnące wg. klucza

arsort(), **krsort()** - sortowanie malejące

np.:

```
$ceny=array("Zeszyt"=>2, "Blok"=>8, "Kredki"=>4);
```

```
asort($ceny); // $ceny=array("Zeszyt"=>2, "Kredki"=>4, "Blok"=>8 );
```

```
ksort($ceny); // $ceny=array("Blok"=>8, "Kredki"=>4, "Zeszyt"=>2 );
```

Operacje na plikach - fopen

```
$wp=fopen("ściezka/nazwa_pliku","tryb",1); // $wp - wskaźnik pliku
```

- Tryby otwarcia pliku:
 - **r** - odczyt (od początku pliku)
 - **r+** - odczyt i zapis (od początku pliku)
 - **w** - zapis (od początku pliku) - jeśli plik nie istnieje następuje próba jego utworzenia, jeśli istnieje to bieżąca zawartość zostanie skasowana
 - **w+** - zapis i odczyt (od początku pliku); (j.w.)
 - **a** - dodawanie zawartości (od końca istniejącej zawartości pliku) a jeśli plik nie istnieje następuje próba jego utworzenia
 - **a+** - dodawanie zawartości i odczyt (od końca istniejącej zawartości pliku); jeśli plik nie istnieje - próba jego utworzenia
 - **b** - tryb binarny (w połączeniu z powyższymi jeśli system rozróżnia pliki tekstowe i binarne)
- Trzeci parametr (opcjonalny) o wartości 1 nakazuje poszukiwanie pliku w lokalizacjach podanych w opcji **include_path (php.ini)** - nie trzeba wtedy podawać ścieżki dostępu do pliku

Poprawność otwarcia pliku

- Funkcja **fopen()**
 - zwraca wartość wskaźnika pliku jeśli operacja otwarcia pliku zostanie zakończona sukcesem
 - zwraca wartość **false** w przeciwnym razie
- Wskazane jest sprawdzenie czy plik udało się otworzyć we wskazanym trybie, np.:

```
@ $wp = fopen("dane.txt", "a", 1);
if (!$wp)
{
    echo "<p>Zamówienie nie może zostać przyjęte.
    Spróbuj później</p>";
    exit;
}
//wykonaj operacje na pliku i zamknij plik
fclose($wp);
```

Przykładowe funkcje odczytu/zapisu

- **Zapis do pliku:**

- **int fwrite (\$wp , \$ciag [, \$ile])** //lub jej alias **fputs**
funkcja zapisuje **\$ciag** do pliku, dopóki nie osiągnie końca ciągu lub zapisze **\$ile** bajtów (zależnie od tego co wystąpi wcześniej, np.:
\$ciag=\$imie."\t".\$nazwisko."\t".\$data."\n"; fwrite(\$wp,\$ciag);

- **Odczyt z pliku:**

- **\$z = fgets(\$wp, 100);** /*czyta linię z pliku dopóki nie natrafi na \n, EOF, lub przeczyta 99 bajtów */
- **string fgetss(int wk, int dlugosc, string [dozw_znaczniki]);** /*podobnie jak fgets, ale z czytanego ciągu usuwa wszystkie znaczniki PHP i HTML poza wyszczególnionymi*/
- **array fgetcsv(int wk, int dlugosc, string [znak_podziału]);**
/* podobnie do fgets() tylko, że przetwarza odczytaną linię na pola i zwraca tablicę zawierającą odczytane pola. */

Odczyt całego pliku

- **int readfile(string nazwa_pliku, int [include_path]);**
 - otwiera plik, wyświetla zawartość w okienku przeglądarki i zamyka plik
 - drugi parametr opcjonalny określa, czy PHP powinien szukać pliku przez opcję include_path (tak jak w fopen)
- **fpasssthru()**
 - po otwarciu pliku funkcja wyświetla jego zawartość w okienku przeglądarki i po zakończeniu działania zamyka plik
 - funkcja zwraca wartość **true**, jeżeli odczyt powiedzie się lub **false** gdy się nie uda, np.:
`$wp = fopen("dane.txt","r"); fpasssthru($wp);`
- **\$tablicazpliku = file(\$wp);**
 - działa podobnie jak jak readfile(), ale zamiast wyświetlać zawartość pliku w przeglądarce, zamienia ją na tablicę tak, że każda linia pliku staje się osobnym elementem tablicy wynikowej
 - liczba elementów tablicy to liczba wierszy pliku

Inne funkcje plikowe

Ustawianie wskaźnika w pliku:

- **rewind()** - ustawienie wskaźnika na początek pliku
- **ftell()** - podaje jak daleko (w bajtach) został przesunięty wskaźnik
- **fseek(int wp,int b)** - ustawia wskaźnik pliku w punkcie b bajtów, licząc od początku pliku

Inne użyteczne funkcje:

- **file_exists()** – sprawdza istnienia pliku bez jego otwierania np.:
`if (file_exists("plik")) echo "komunikat 1";
else echo "komunikat 2";`
- **filesize()** - określa wielkości pliku w bajtach, np.:
`$wp = fopen("plik.txt","r");
echo fread($wp, filesize("plik.txt"));
fclose($wp);`

Synchronizacja dostępu do plików - flock

Blokowanie dostępu do pliku:

- **flock(int wp, int blokada)** - zwraca true, jeśli blokada jest prawidłowa
- flock() należy dodać do wszystkich skryptów korzystających z pliku

Rodzaje blokady:

- **LOCK_SH** (1) - pozwala na dzielenie pliku z innymi czytającymi (shared lock)
- **LOCK_EX** (2) - wyłącza plik z użytku - nie może być dzielony (exclusive lock)
- **LOCK_UN** (3) - zwolnienie istniejącej blokady (release a lock)

Np.:

```
$wp = fopen("plik.txt", "a");
//blokada zapisu:
flock( $wp, LOCK_EX);
fwrite($wp, "ciag");
//zwolnienie blokady zapisu:
flock( $wp, LOCK_UN);
fclose( $wp );
```

Przykład - pobieranie tablicy z pliku

```
<?php
    //załadowanie pliku do tablicy, gdzie
    //każda linia pliku to jeden element tablicy
    $osoby = file("osoby.txt");
    //określenie liczby elementów tablicy (wierszy z pliku):
    $ile = count($osoby);
    for ( $i=0; $i<$ile; $i++)
    {
        echo $osoby[$i]."<br /> ";
    }
?>
```

1 Ania Kowalska
2 Magdalena Nowak
3 Jan Abacki
4 Adam Babacki

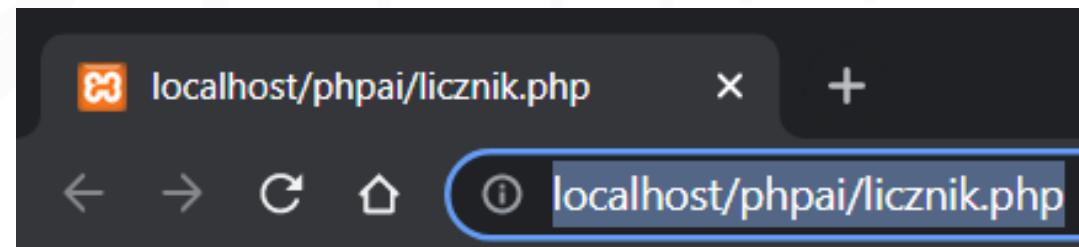
Przykład - pobieranie tablicy z pliku i dostęp do poszczególnych danych

```
<?php  
$osoby = file("osoby.txt");  
$ile = count($osoby);  
echo "<table border='1'><tbody>";  
echo "<tr><th> ID</th><th>Imię</th>";  
echo "<th> Nazwisko</th></tr>";  
for ( $i=0; $i<$ile; $i++)  
{  
    $dane = explode(" ",$osoby[$i]); //podział ciągu  
                                //względem separatora  
    $id = intval($dane[0]); //konwersja tekstu na wartość  
                            //całkowitą  
    echo "<tr><td>$id</td><td>$dane[1]</td>  
          <td>$dane[2]</td></tr>";  
}  
echo "</tbody></table>"; ?>
```

ID	Imię	Nazwisko
1	Ania	Kowalska
2	Magdalena	Nowak
3	Jan	Abacki
4	Adam	Babacki

Przykład - licznik odwiedzin

```
<?php // W pliku httpd.conf: DocumentRoot "C:/xampp/htdocs"
      // Plik umieścimy poza katalogiem htdocs, np. w C:/xampp/Mojepliki
      $d_root = $_SERVER['DOCUMENT_ROOT'];
if (!(file_exists("$d_root/..../Mojepliki/liczba.txt")))
{ $plik=fopen("$d_root/..../Mojepliki/liczba.txt", "w+");
  fputs($plik, "0");
  fclose($plik); }
$plik=fopen("$d_root/..../Mojepliki/liczba.txt", "r+");
if (!$plik) { echo "Nie da się otworzyć pliku."; }
else {
  flock($plik, LOCK_EX);
  $ile=fgets($plik,255);
  $ile++;
  print "Licznik wskazuje: $ile";
  fseek($plik,0);
  fputs($plik,$ile);
  flock($plik,LOCK_UN);
  fclose($plik);
}
?>
```



Licznik wskazuje: 5

Przykład - formularz ankiety sonda.html

```
<form method="post" action="dopisz.php">  
    Czy podoba ci się nasz serwis? <br />  
    <input type="radio" name="odp" value="tak"/> Tak<br />  
    <input type="radio" name="odp" value="nie"/> Nie<br />  
    <input type="radio" name="odp" value="nw"/> Nie wiem<br />  
    <input type="submit" value="Wyślij"/>  
</form>
```

Czy podoba ci się nasz serwis?

Tak

Nie

Nie wiem

Wyślij

Przykład - plik dopisz.php

```
<?php  
if (!(file_exists("sonda.txt"))) {  
    $plik=fopen("sonda.txt","w+");  
    fputs($plik,"0#0#0"); fclose($plik); }  
$plik=fopen("sonda.txt", "r+")  
        or exit ("Nie da się otworzyć pliku.");  
$linia = fgets($plik, 255);  
$tab = explode("#",$linia); //rozbicie ciągu $linia na el. tablicy  
$tbl["tak"] = $tab[0]; //tworzenie pomocniczej tablicy asocj.  
$tbl["nie"] = $tab[1]; $tbl["nw"] = $tab[2];  
++$tbl[$_POST['odp']]; //zwiększenie liczby głosów na daną odpowiedź  
fseek($plik, 0); //przesunięcie wskaźnika pliku do początku  
//zapis do pliku:  
fwrite($plik, $tbl["tak"]."#".$tbl["nie"]."#".$tbl["nw"]);  
header("Location: sonda.php"); //ustawienie nagłówka HTTP  
?>
```

3#2#2|

Przykład - plik sonda.php

```
<table border="1"><tbody>
<?php
    $file = fopen("sonda.txt","r+");
    $linia = fgets($file, 255);
    $tab = explode("#",$linia);
    $razem=$tab[0]+$tab[1]+$tab[2];
    echo "<tr><td>Tak</td><td>". $tab[0]. "</td></tr>";
    echo "<tr><td>Nie</td><td>". $tab[1]. "</td></tr>";
    echo "<tr><td>Nie wiem</td><td>". $tab[2]. "</td></tr>";
    echo "<tr><td>Łącznie głosów:</td><td>". $razem. "</td></tr>";
?
</tbody></table>
<br /><a href="sonda.html"> Powrót do formularza sondy</a>
```

Tak	3
Nie	2
Nie wiem	2
Łącznie głosów:	7

[Powrót do formularza sondy](#)

Przykładowe funkcje do operacji na ciągach

- **array explode(string separator, string tekst)** - rozbicie ciągu na części według separatora
- **implode(), join()** - efekt odwrotny do działania explode()
- **string substr(string ciag, int start, int [dlugosc])** - zwraca podciąg skopiowany z ciągu
- **int strcmp(string ciag1, string ciag2)** - jeżeli ciągi są równe funkcja zwraca 0, jeżeli ciag1>=ciag 2 (w porządku leksykograficznym) zwracana jest liczba >0, w przeciwnym razie liczba < 0, przy czym uwzględniana jest wielkość liter
- **strcasecmp()** - j.w. tylko nie uwzględniane są wielkości liter
- **strstr(), strchr ()** - odnajdywanie tekstu lub znaku w ciągu z uwzględnieniem wielkości liter
- **strpos()** - działa podobnie jak strstr(), lecz zwraca numeryczną pozycję podciągu
- **strspos()** - j.w. lecz zwraca pozycję ostatniego wystąpienia tekstu w ciągu
- **string str_replace(string tekst, string nowy_tekst, string ciag)** - zamienia wszystkie znalezione fragmenty tekstu na nowy_tekst w ciągu

Wyrażenia regularne i funkcje PCRE

- Funkcje PCRE wymagają aby wyrażenie regularne było ujęte w ograniczniki (delimiters - znaki / # ~), np. **#[a-z]#i**
- **Przykładowe modyfikatory** dodawane po końcowym ograniczniku:
 - *i* (*PCRE_CASELESS*) - brak rozróżniania dużych i małych liter
 - *m* (*PCRE_MULTILINE*) - przy ustawionym modyfikatorze, jeśli w łańcuchu nie występuje znak "\n" w rozważanym ciągu, ustawienie nie ma znaczenia
 - *s* (*PCRE_DOTALL*) - kropka we wzorcu pasuje do wszystkich znaków, łącznie ze znakiem nowej linii (normalnie jest on wykluczony)
 - *x* (*PCRE_EXTENDED*) - białe znaki we wzorcu są ignorowane
- Przykładowe funkcje PCRE:
 - `preg_filter` - wyszukuje wyrażenie i zastępuje podanym
 - `preg_grep` - zwraca tablicę ciągów pasujących do wzorca
 - `preg_last_error` - zwraca kod błędu ostatniego wykonania funkcji PCRE
 - `preg_match_all` - globalne dopasowanie wyrażenia regularnego
 - `preg_match` - dopasowanie wyrażenia regularnego
 - `preg_replace` - wyszukuje RegExp i zamienia podanym
 - `preg_split` - dzieli串 względem RegExp

Funkcja preg_replace

preg_replace(string|array \$pattern, string|array \$replacement, string|array \$subject, int \$limit = -1, int &\$count = null): string|array|null

- \$pattern - łańcuch/tablica łańcuchów
- \$replacement - łańcuch/tablica łańcuchów do zamiany
- \$subject - łańcuch/tablica łańcuchów do wyszukania i zamiany
- \$limit - maksymalna liczba zamian (domyślnie -1 b.o.)
- \$count - jeśli podana to wskazuje liczbę zamian

Np.:

```
<?php $ciag = 'Ala ma kota i Ala ma psa';
```

```
$wzor1 = '/A/';  
$wzor2 = '/A/i';  
$zamiana = 'U';
```

```
Ula ma kota i Ula ma psa  
UlU mU kotU i UlU mU psU
```

```
echo preg_replace($wzor1, $zamiana, $ciag);  
echo '<br />';  
echo preg_replace($wzor2, $zamiana, $ciag); ?>
```

Przykład - preg_replace dla tablic

```
<?php  
    $ciag = 'Szybki, rudy lis.';  
    echo $ciag.'<br />';  
    $wzor = [];  
    $wzor[0] = '/szybki/i';  
    $wzor[1] = '/rudy/';  
    $wzor[2] = '/lis/';  
    $zamiana = [];  
    $zamiana[0] = 'Powolny';  
    $zamiana[1] = 'bury';  
    $zamiana[2] = 'niedźwiedź';  
    echo preg_replace($wzor, $zamiana, $ciag);  
?>
```

Szybki, rudy lis.
Powolny, bury niedźwiedź.

Funkcja preg_match

**preg_match(string \$pattern, string \$subject,
array &\$matches = null, int \$flags = 0, int \$offset = 0): int|false**

- \$pattern, \$subject - jak poprzednio
- \$matches - tablica z wynikami wyszukiwania: *\$matches[0]* zawiera tekst wzorca, *\$matches[1]* - pierwszy pasujący tekst itp.
- \$offset -opcjonalnie inna niż początek ciągu pozycja startu (w bajtach)
- zwraca albo 0 albo 1 (kończy szukanie po pierwszym
znalezieniu ciągu pasującego do wzorca - w przeciwieństwie
do funkcji preg_match_all(), która przeszukuje ciąg do końca
i zwraca FALSE w przypadku wystąpienia błędu

Przykład - funkcja preg_match

```
<?php  
if (preg_match("/^(W3C)$/i", "Skrót W3C"))  
{ echo "<br />1. Wzór został znaleziony."; }  
else { echo "<br />1. Wzór nie został znaleziony."; }  
if (preg_match("/(W3C)$/i", "Skrót W3C"))  
{ echo "<br />2. Wzór został znaleziony."; }  
else { echo "<br />2. Wzór nie został znaleziony."; }  
if (preg_match("/^(W3C)$/i", "W3C"))  
{ echo "<br />3. Wzór został znaleziony."; }  
else { echo "<br />3. Wzór nie został znaleziony."; }  
}  
?>
```

1. Wzór nie został znaleziony.
2. Wzór został znaleziony.
3. Wzór został znaleziony.

Przykład - walidacja adresu e-mail

```
<?php
if (isset($_POST['submit']))
{ if (!preg_match("/^[_a-zA-Z0-9_-]+@[a-zA-Z0-9\-.]
+\.[a-zA-Z0-9\-\.\.]+\$/", $_POST['mail']))
{ echo "<h2> Niepoprawny adres e-mail </h2><br/>";
echo "Spróbuj jeszcze raz:
                  <a href='reg1.php'>Wstecz</a>" ;
} else echo "<h2> Dziękujemy za e-mail!</h2>";
}
Else { ?>
<h4>Podaj e-mail :</h4>
<form method="post" action="reg1.php">
<input name="mail" width="25"/><br /><br />
<input type="submit" value="Wyślij" name="submit"/>
<input type="reset" value="Anuluj"/>
<?php }
?>
```

Filtrowanie danych skalarnych i tablic

- **filter_input(typ, nazwa, filtr, opcje)** - filtruje dane pochodzące ze wskazanego wejścia i zwraca je przefiltrowane lub **false**
 - **typ** - typ wejścia np. INPUT_GET, INPUT_POST, INPUT_COOKIE
 - **nazwa** - nazwa zmiennej do sprawdzenia
 - **filtr** - opcjonalnie rodzaj filtra (domyślnie FILTER_DEFAULT - brak)
 - **opcje** - opcjonalnie, **flagi/opcje** zależne od rodzaju filtra
- **filter_var** - filtuje pojedynczą **zmienną** zgodnie ze wskazanym filtrem ¶
- **filter_input_array** - pobiera dane zewnętrzne i opcjonalnie je filtuje
- **filter_var_array** - pobiera tablicę zmiennych i opcjonalnie je filtuje ¶
- **Typy filtrów:**
 - **FILTER_VALIDATE** (filtry do sprawdzania poprawności danych - walidujące),
http://www.w3schools.com/php/php_ref_filter.asp
 - **FILTER_SANITIZE** (filtry czyszczące - np. usuwające wybrane znaki specjalne lub zastępujące je bezpiecznymi encjami),
<http://php.net/manual/en/filter.filters.sanitize.php>

Przykładowe filtry

- **Filtr do walidacji:**
 - FILTER_VALIDATE_BOOLEAN, FILTER_VALIDATE_EMAIL, FILTER_VALIDATE_FLOAT, FILTER_VALIDATE_INT, FILTER_VALIDATE_REGEXP, FILTER_VALIDATE_URL
- **Filtr czyszczące:**
 - FILTER_SANITIZE_EMAIL - usuwa wszystkie znaki za wyjątkiem liter, cyfr oraz znaków !#\$%&'*+-=?^`{|}~@.[]
 - FILTER_SANITIZE_ENCODED - opcjonalnie usuwaj lub koduje znaki specjalne (zgodnie z zadaną flagą)
 - FILTER_SANITIZE_FULL_SPECIAL_CHARS - usuwa lub koduje znaki specjalne (zgodnie z zadaną flagą) - odpowiednik **htmlspecialchars()**
 - FILTER_SANITIZE_ADD_SLASHES - odpowiednik **addslashes()**
 - FILTER_SANITIZE_NUMBER_INT - usuwa wszystkie znaki poza cyframi oraz znakami + -

Przykład - walidacja adresu e-mail za pomocą funkcji filter_input

```
<?php
if (filter_input(INPUT_POST, "submit"))
{ //Naciśnięto przycisk z name="submit"
    if (!filter_input(INPUT_POST, "mail",
                    FILTER_VALIDATE_EMAIL))
    { echo "<h2> Niepoprawny adres e-mail </h2><br/>";
        echo "Spróbuj jeszcze raz:
              <a href='reg1.php'>Wstecz</a>";
    } else echo "<h2> Dziękujemy za e-mail!</h2>";
}
else
// ... dalej jak na slajdzie 62
```

Przykład - walidacja za pomocą funkcji filter_var

```
$str = "Ala ma kota i Ala ma psa";
$opt = [ "options" => [ "regexp" => "/Ale/" ] ];
if (filter_var($str, FILTER_VALIDATE_REGEXP, $opt ))
    echo "Znaleziono pasujący ciąg ";
else echo "Nie znaleziono pasującego ciągu";
```

```
$email = filter_input(INPUT_POST, "mail");
if (!filter_var($email, FILTER_VALIDATE_EMAIL)) {
    echo("$email poprawny");
} else
{   echo("$email nie poprawny");
}
```

Przykład - walidacja za pomocą funkcji filter_input_array (1)

```
<?php
error_reporting(E_ALL | E_STRICT);

/* Przykładowe dane przesłane metodą POST:
$_POST = array(
    'product_id' => 'libgd<script>',
    'component'   => '10',
    'versions'    => '2.0.33',
    'testscalar'  => array('2', '23', '10', '12'),
    'testarray'   => '2',
);
*/
```

Przykład – walidacja za pomocą funkcji filter_input_array (2)

//przygotowanie tablicy argumentów do walidacji:

```
$args = [
    'product_id'  => FILTER_SANITIZE_ENCODED,
    'component'   => [ 'filter' => FILTER_VALIDATE_INT,
                        'flags'  => FILTER_REQUIRE_ARRAY,
                        'options'=> [ 'min_range' => 1,
                                      'max_range'  => 10 ] ],
    'versions'    => FILTER_SANITIZE_ENCODED,
    'doesnotexist'=> FILTER_VALIDATE_INT,
    'testscalar'   => [ 'filter' => FILTER_VALIDATE_FLOAT,
                        'flags'  => FILTER_REQUIRE_SCALAR ],
    'testarray'    => [ 'filter'  => FILTER_VALIDATE_INT,
                        'flags'   => FILTER_REQUIRE_ARRAY ]
];
```

Przykład - kontynuacja skryptu i wyświetlenie wyniku walidacji

```
$myinputs = filter_input_array(INPUT_POST, $args);
var_dump( $myinputs );
?>
//Przykładowy wynik:
array(6) {
    ["product_id"] => string(17) "libgd%3Cscript%3E"
    ["component"] => array(1) { [0] => int(10) }
    ["versions"] => string(6) "2.0.33"
    ["doesnotexist"] => NULL
    ["testscalar"] => bool(false)
    ["testarray"] => array(1) { [0]=> int(2) }
}
```



PROGRAMOWANIE APLIKACJI INTERNETOWYCH

Wykład 3

Funkcje i szablony w PHP. Interakcja aplikacji internetowej
z systemem plików na serwerze WWW.

Beata Pańczyk



Funkcje w PHP

- Definicja funkcji:

```
function nazwa_funkcji($arg1, $arg2, ...)  
{ instrukcje;  
    return $wynik; //nie musi występować return  
}
```
- Dopuszczalna jest dowolna liczba argumentów funkcji i wartości domniemane parametrów
- Domyślnie parametry przekazywane **przez wartość**
- Możliwość przekazywania parametrów przez referencje
- Możliwość przekazywania zmiennej liczby parametrów
- Możliwe są wywołania rekurencyjne funkcji
- Funkcja może zawierać definicje kolejnych funkcji i definicje klas
- Wszystkie funkcje (i klasy) w PHP mają zasięg globalny - mogą być wołane z zewnątrz, nawet jeśli są zdefiniowane wewnątrz innej funkcji
- Brak przeciążania nazw funkcji

Zasięg zmiennych

- **Lokalny** - zmienne zadeklarowane wewnątrz funkcji to **zmienne lokalne** (widoczne od miejsca deklaracji do końca funkcji)
- **Globalny** - zmienne zadeklarowane na zewnątrz funkcji to **zmienne globalne** (widoczne od miejsca deklaracji do końca pliku, ale nie wewnątrz funkcji!)
- słowo kluczowe **global** można zastosować do ustalenia globalnego zasięgu zmiennej zdefiniowanej lub stosowanej wewnątrz funkcji
- Funkcja **unset(\$nazwa_zmiennej);** - usuwa zmienną

Przykład - zasięg zmiennych globalnych

```
<?php
    function fun()
    {   echo "W funkcji: n = $n <br />";
        $n=5;
        echo "W funkcji: n=$n <br />";
    }
    $n=10;
    fun();
    echo "Na zewnątrz funkcji: n=$n <br />";
?>
```

Warning: Undefined variable \$n in
C:\xampp\htdocs\phpai\przyklady\funkcje.php on line **16**
W funkcji: n=
W funkcji: n=5
Na zewnątrz funkcji: n=10

Przykład - zasięg zmiennych globalnych

```
<?php  
function fun()  
{ global $n;  
    $n=5;  
    echo "W funkcji: n=$n <br />";  
}  
$n=10;  
echo "Na zewnątrz funkcji: n=$n <br />";  
fun();  
echo "Na zewnątrz funkcji:n=$n <br />";  
?>
```

Na zewnątrz funkcji: n=10
W funkcji: n=5
Na zewnątrz funkcji:n=5

UWAGA!

Miejsce deklaracji funkcji
nie ma znaczenia - ważne
jest miejsce wywołania
funkcji

Przekazywanie parametrów

- **Przez wartość** - w bloku funkcji tworzona jest kopia oryginału, można ją dowolnie modyfikować, ale wartość oryginalnej zmiennej poza funkcją pozostaje niezmieniona, np.:

```
<?php
    function dodaj($a,$b) { $a=$a+$b; }
    $a=10; dodaj($a,5); echo $a;
?>
```
- **Przez referencję** - funkcja otrzymuje referencję (odniesienie) do oryginału, każda modyfikacja referencji dotyczy również oryginału, np.:

```
<?php
    function dodaj(&$a,$b=5) { $a=$a+$b; }
    $a=10; dodaj($a); echo $a;
?>
```

Funkcje require, include, require_once, include_once

- Dołączają i wykonują wskazany plik
- **require()** działa analogicznie do **include()**, tylko w razie błędu generuje błąd E_COMPILE_ERROR (fatal error) i zatrzymuje działanie skryptu, podczas gdy include() zgłasza jedynie komunikat E_WARNING (warning), co pozwala skryptowi na dalsze działanie
- Jeśli w pliku php.ini jest włączona opcja **allow_url_include = On** to można dołączać pliki wskazując ich URL
- **require_once()** - identyczne działanie jak require(), poza tym, że PHP sprawdzi czy żądany plik nie został już dołączony i jeśli tak to nie dołączy go ponownie
- **include_once()** - j.w.
- Wykorzystywane w przypadku kiedy te same pliki mogą być dołączane i wykonywane więcej niż jeden raz (pozwala to uniknąć problemów związanych z redefiniowaniem funkcji, nadpisywaniem zmiennych itp.)

Dołączanie plików

- Dołączane pliki powinny mieć rozszerzenie **.php** i ze względów bezpieczeństwa powinny się znajdować w odrębnym katalogu
- Dowolny kod w pliku dołączanym, który powinien podlegać parsowaniu PHP musi być umieszczony w znacznikach `<?php ... ?>`
- Jeśli dołączany plik nie jest ujęty w znaczniki `<? php ... ?>` jest traktowany jako zwykły tekst lub HTML
- Zastosowanie - tworzenie szablonów stron, np.: plik **glowny.php**:
`<?php`
 `include_once("naglowek.php");`
 `include_once("tresc.php");`
 `include_once("stopka.php");`
`?>`

Przykład - naglowek.php

```
<!doctype ...>
<html>
    <head>
        <meta ... />
        <title> Szablon strony</title>
        <link rel="stylesheet" href="styl.css"
              type="text/css" />
    </head>
    <body>
        <div id = "kontener" >
            <div id = "nag" >
                <!-- TYTUŁ STRONY -->
                <!-- Nawigacja -->
            </div>
        </div>
```

Przykład - pozostałe pliki

- plik **tresc.php**:

```
<div id = "tresc" >  
    <! – Główna zawartość strony -->  
</div>
```

- plik **stopka.php**:

```
<div id = "stopka" >  
    <! – Zawartość stopki -->  
</div><! – koniec kontenera głównego -->  
</body>  
</html>
```

Pliki zdalne

- Pliki zdalne mogą być przetwarzane na zdalnym serwerze (w zależności od rozszerzenia pliku i tego czy serwer zdalny uruchamia pliki PHP czy nie) - niezależnie od tego pliki te muszą zawierać poprawny skrypt PHP ponieważ są one przetwarzane na serwerze lokalnym (jako dołączone pliki), np.:

```
include 'http://www.example.com/file.php?x=1&y=2';
```

- Jeśli plik z serwera zdalnego ma być wykonany i tylko jego **wynik** ma być pobrany, lepszym rozwiązaniem jest funkcja **readfile()**, w przeciwnym razie należy zwrócić szczególną uwagę na to aby zdalny skrypt zawierał poprawny kod

Operacje plikowe

- **rename()**

```
$wynik= rename($stara_nazwa,$nowa_nazwa);  
if ($wynik) echo "Nazwa pliku została zmieniona.";
```

- **copy()**

```
$wynik= copy($nazwa_pliku1,$nazwa_pliku2);  
if ($wynik) echo "Plik został skopiowany.";
```

- **unlink()**

```
if (unlink($nazwa_pliku)) echo "Plik został usunięty.";
```

- **if (file_exists(\$nazwa_pliku))**

```
echo "Plik $nazwa_pliku istnieje.;"
```

```
else
```

```
echo "Plik $nazwa_pliku nie istnieje.;"
```

Informacje o pliku: data utworzenia, ostatni dostęp, modyfikacja

- **fileatime()** – czas ostatniego dostępu
- **filemtime()** – czas ostatniej modyfikacji
- powyższe funkcje zwracają czas w formacie UNIXa, który należy skonwertować funkcją **date()**

Ostatni dostęp do pliku dane.txt: July 23 2021 19:03:10.
Ostatnia modyfikacja pliku dane.txt: 06 07 2021 20:56:21.

```
$plik = 'dane.txt';
if (file_exists($plik))
{ echo "Ostatni dostęp do pliku $plik: ".
      date("F d Y H:i:s.", fileatime($plik));
  echo "<br />";
  echo "Ostatnia modyfikacja pliku $plik: ".
      date("d m Y H:i:s.", filemtime($plik));
}
```

Szczegółowe informacje o nazwie pliku

- **pathinfo("nazwa_pliku")** - pobiera informację o nazwie pliku i zwraca ją w postaci tablicy asocjacyjnej czterech elementów o kluczach: **dirname**, **basename**, **extension**, **filename**
- Np.:

```
$info_pliku = pathinfo("osoby.txt");
var_dump($info_pliku);
```

```
array(4) { ["dirname"]=> string(1) "."
["basename"]=> string(9) "osoby.txt"
["extension"]=> string(3) "txt"
["filename"]=> string(5) "osoby" }
```

Formularz wysyłania plików na serwer

- Ustawienie metody POST w elemencie form i parametru **enctype="multipart/form-data"** powoduje, że plik będzie wysłany razem ze zwykłymi danymi formularza
- **<input type="file" name="plik" >** - dla pola typu **file** przeglądarka wyświetla okno dialogowe do wyboru pliku
- Wielkość przesyłanych plików można ustawiać w pliku **php.ini**:
upload_max_filesize = 16M
- Informacje o przesłanych plikach znajdują się w tablicy **\$_FILES**
- Funkcja **var_dump(\$_FILES)** wyświetli np.:
**array(1) { ["plik"]=>array(5)
 { ["name"]=>string(10) "jesien.jpg"
 ["type"]=>string(10) "image/jpg"
 ["tmp_name"]=>string(24) "C:\xampp\tmp\phpD9FB.tmp"
 ["error"]=>int(0)
 ["size"]=>int(754344)
 } }**

Informacje o pliku

- **name** - oryginalna nazwa pliku
- **type** - typ pliku MIME
- **tmp_name** - nazwa przekazanego pliku na serwerze
- **error** - informacja o błędzie
- **size** - wielkość pliku w bajtach
- plik po wysłaniu trafia do domyślnego katalogu tymczasowego na serwerze i jeżeli nie nastąpi przeniesienie lub zmiana nazwy pliku zanim skrypt się zakończy - plik zostanie **skasowany**
- **move_uploaded_file("plik_uzytkownika", "docelowe_miejsce")**
funkcja realizuje przeniesienie pliku z tymczasowej lokalizacji na miejsce docelowe. Funkcja sprawdza bezpieczeństwo i wykonuje kopiowanie, zwraca false lub true

Formularz wysyłania plików na serwer

```
<h3> Wysyłanie plików na serwer </h3>
<form method="post" action="odbierzplik.php"
      enctype="multipart/form-data" >
  <p>Wybierz plik: <input name="plik" type="file" size="50">
  </p>
  <p><input type="submit" name="submit" value="Wyślij"></p>
</form>
```

Wysyłanie plików na serwer

Wybierz plik: IMG_2014..._183135.jpg

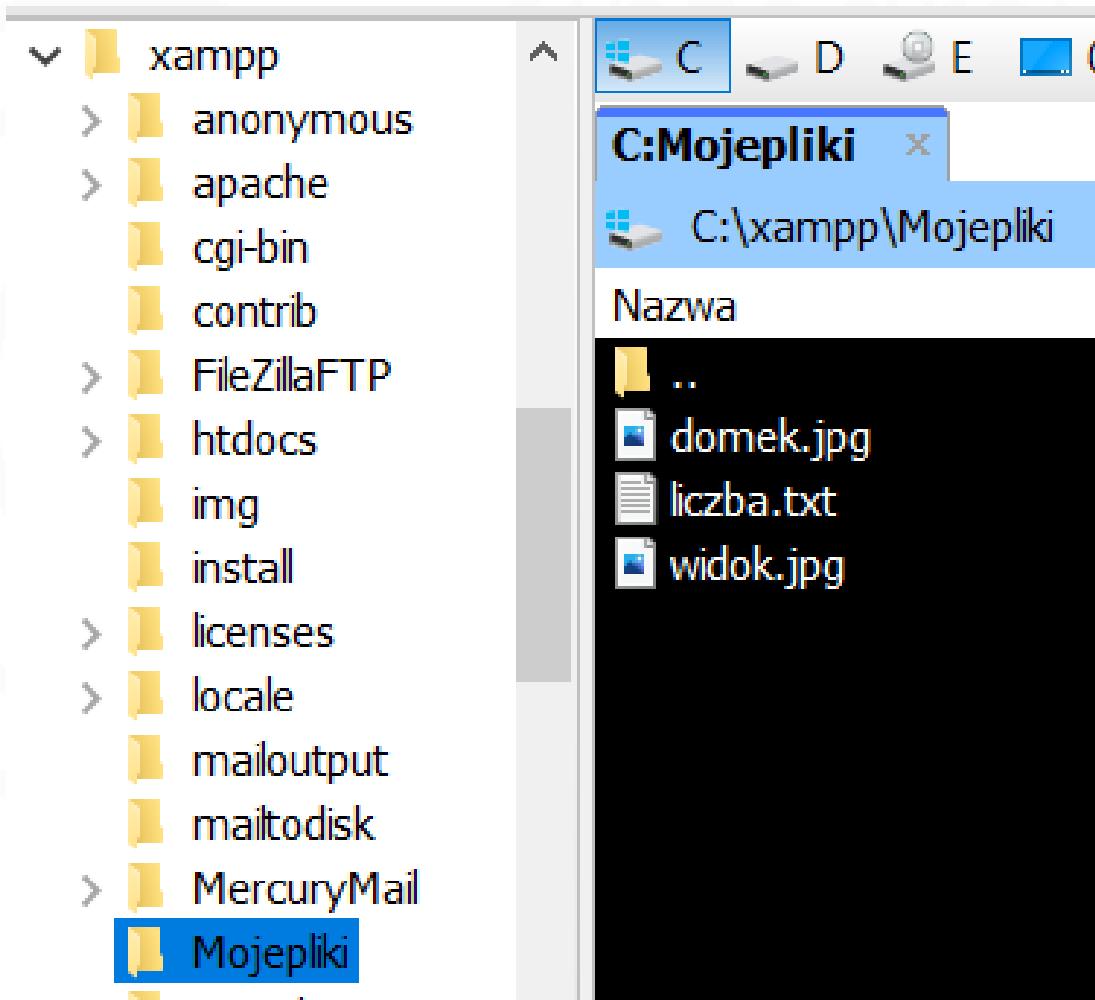
Skrypt odbierzplik.php

```
<h3> Wysyłanie pliku ... </h3>
<?php
    $tmp_name=$_FILES['plik']['tmp_name'];
    $name=$_FILES['plik']['name'];
    $size=$_FILES['plik']['size'];
    $path=$_SERVER['DOCUMENT_ROOT']."/..../Mojepliki/".$name;
    if (move_uploaded_file($tmp_name,$path))
        echo "Wysłano plik: $name, o rozmiarze: $size bajtów.";
    else "Plik nie został wysłany";
    echo "<br />";
?>
```

Wysyłanie pliku ...

Wysłano plik: widok.jpg, o rozmiarze: 2991735 bajtów.

Lokalizacja folderu Mojepliki na serwerze Apache



Restrykcje na wysyłane pliki

```
<?php
if ((($_FILES["plik"]["type"]=="image/gif")||($_FILES["plik"]["type"]=="image/jpeg")||($_FILES["plik"]["type"]=="image/jpg"))&&
    ($_FILES["plik"]["size"] < 1000000))
{ if ($_FILES["plik"]["error"] > 0)
    { echo "Error: " . $_FILES["plik"]["error"] . "<br />"; }
    else
    { echo "Przesyłany plik: " . $_FILES["plik"]["name"] . "<br />";
        echo "Typ: " . $_FILES["plik"]["type"] . "<br />";
        echo "Rozmiar: " . ($_FILES["plik"]["size"]/1024)." kB<br />";
        echo "Folder tymczasowy: " . $_FILES["plik"]["tmp_name"];
        //wywołanie funkcji move_uploaded_file - jak na poprzednim slajdzie
    }
}
else
{
    echo "Błędny plik";
}
?>
```

Wysyłanie pliku ...

Przesyłany plik: obraz4.JPG
Typ: image/jpeg
Rozmiar: 14.0087890625 kB
Folder tymczasowy: C:\xampp\tmp\phpDB0D.tmp
Wysłano plik: obraz4.JPG, o rozmiarze: 14345 bajtów.

Odczyt właściwości pliku

- Jednoargumentowe funkcje **is_readable()**, **is_writeable()**, **is_executable()**, **is_file()**, **is_dir()** działają na plikach serwera (nie na plikach zdalnych) i zwracają wartość boolean
- Przy kilkakrotnym wywołaniu funkcji **is_file()** należy wyczyścić buforowane dane funkcją **clearstatecache()**
- **fileowner()** - odczytanie właściciela pliku
- Sprawdzenie prawa do odczytu np.:

```
$nazwa_pliku="d:\beatap\wyslij.html"; //Windows  
if (is_readable($nazwa_pliku))  
    print file_get_contents($nazwa_pliku);  
else print "Nie masz prawa do odczytu pliku: $nazwa_pliku";
```
- Sprawdzenie właściciela pliku np.:

```
$wlasiciel=fileowner("etc/passwd");  
if ($wlasiciel!=0) print "Ostrzeżenie: właścicielem jest root!";
```

Praca z katalogami

- **opendir()** - otwarcie katalogu do odczytu, funkcja zwraca uchwyt (wskaźnik) katalogu podobnie jak funkcja fopen() zwraca uchwyt pliku, np.:
`$kat = opendir($aktualny_katalog);`
- **readdir(\$kat)** - odczyt plików z otwartego katalogu, funkcja zwraca false, kiedy wszystkie pliki zostaną już odczytane, pliki nie są uporządkowane w żaden określony sposób
- **rewinddir(\$kat)** - powrót do odczytu nazw plików od początku katalogu
- **mkdir(\$kat, prawa_dostępu)** – zwraca true lub false
- **rmdir(\$kat)** - usuwa pusty katalog
- **closedir(\$kat)**

Przeglądanie katalogu

```
<?php  
$katalog=  
    filter_input(INPUT_SERVER, 'DOCUMENT_ROOT')."/../Mojepliki/";  
$kat=@opendir($katalog) or die("Nie można otworzyć katalogu");  
echo "<h3> Zawartość katalogu:<br />$katalog</h3>";  
while ($plik = readdir($kat))  
    echo "$plik<br />";  
closedir($kat);  
?>
```

Zawartość katalogu:
C:/xampp/htdocs/..//Mojepliki/

.

..

domek.jpg

liczba.txt

widok.jpg

Wyświetlanie informacji o pliku

Modyfikacja pętli while z poprzedniego slajdu:

```
while ($plik=readdir($kat))  
echo "<a href="  
'wlasciwosci_pliku.php?plik=$plik&katalog='.$katalog' >  
$plik</a><br />";
```

Zawartość katalogu:

C:/xampp/htdocs/../**Mojepliki/**

..

..

[domek.jpg](#)

[liczba.txt](#)

[widok.jpg](#)

WŁAŚCIWOŚCI PLIKU: domek.jpg

Katalog: C:/xampp/htdocs/../**Mojepliki/**

Utworzony: 11 July 2021 20:21

Zmodyfikowany: 11 July 2021 20:21

Typ pliku: file

Rozmiar pliku: 3456740

Skrypt wlasciwosci_pliku.php

```
<?php  
    $katalog=$_GET['katalog'];  
    $name=$_GET['plik']; $plik="$katalog/".$name;  
    $utworz=fileatime($plik); $modyf=filemtime($plik);  
    $data_u=date("j F Y H:i", $utworz);  
    $data_m=date("j F Y H:i", $modyf);  
    echo "<h4>WŁAŚCIWOŚCI PLIKU: $name </h4>";  
    echo "Katalog: $katalog <br >";  
    echo "Utworzony: $data_u <br />";  
    echo "Zmodyfikowany: $data_m <br />";  
    echo "Typ pliku: ". filetype($plik). "<br />";  
    echo "Rozmiar pliku: ". filesize($plik). "<br />";  
?>
```

Pliki zdalne

- Ustawienie w php.ini: **allow_url_fopen=On**
- Praca z plikiem zdalnym:

```
echo "<h3>Informacje z pliku zdalnego</h3>";  
//Połączenie z URL i odczytanie informacji:  
$url = "https://www.php.net/";  
$wp = @fopen($url, "r") or die("Otwarcie url niemożliwe");  
$zawartosc = strip_tags(fread($wp, 10000));  
echo $zawartosc;  
fclose($wp);
```

Informacje z pliku zdalnego

php



- [Downloads](#)
- [Documentation](#)
- [Get Involved](#)

Informacje z pliku zdalnego

PHP: Hypertext Preprocessor Downloads Documentation
Get Involved Help Getting Started Introduction A simple
tutorial Language Reference Basic syntax Types
Variables Constants Expressions Operators Control
Structures Functions Classes and Objects Namespaces
Errors Exceptions Generators Attributes References
Explained Predefined Variables Predefined Exceptions



PROGRAMOWANIE APLIKACJI INTERNETOWYCH

Wykład 4

Programowanie obiektowe w PHP. Klasy i obiekty. Definiowanie składowych klasy. Metody i typy argumentów w metodach.

Zarządzanie dostępem do klasy. Dziedziczenie. Metody i składowe statyczne. Klasy abstrakcyjne i interfejsy. Obsługa wyjątków.

Beata Pańczyk



Przykład - klasa i obiekt w PHP

```
<?php  
    class A  
    { //deklaracje pól  
        public $a = 'wartość domyślna';  
        //deklaracje metod  
        public function wyswietl_a()  
        { echo $this->a; }  
    }  
  
    $obiekt = new A(); //utworzenie obiektu klasy A  
?>
```

Automatyczne dołączanie obiektów

- W praktyce programowania obiektowego zwykle tworzy się osobne skrypty z definicją klasy, które następnie dołącza do skryptu głównego za pomocą funkcji `include` (dla każdej klasy osobno). W wyniku uzyskuje się długą listę z wywołaniem **`include`**
- Od PHP 5 istnieje możliwość wykorzystania funkcji `_autoload`, która automatycznie wykona dołączanie odpowiednich klas do kodu skryptu głównego. Funkcja taka będzie automatycznie wywoływana w momencie próby tworzenia obiektu klasy nie znanej jeszcze w skrypcie (jest to ostatnia próba dołączenia klasy przed wysłaniem komunikatu o błędzie)

Funkcja autoload

- Zakładamy, że skrypty **Klasa1.php** i **Klasa2.php** zawierają definicje odpowiednich klas
- Dołączenie klas z tych plików:

```
<?php  
    function __autoload($kласа)  
    { require_once $kласа.'.php'; }  
  
    $об1 = new Klasa1();  
    $об2 = new Klasa2();  
?>
```

Konstruktory, destruktory i specyfikatory dostępu

- Konstruktor: `__construct ([argumenty [, ...]])`
- Destruktor: `__destruct ()`
- Konstruktor (i destruktor) klasy bazowej musi być jawnie wywoływany w klasie pochodnej
- **Specyfikatory dostępu:**
 - **public** - nieograniczony dostęp do składników klasy
 - **protected** - dostęp do składników klasy mają tylko klasy pochodne
 - **private** - niemożliwy dostęp spoza klasy)
- Brak specyfikatora dostępu oznacza dostęp **public**

Konstruktor klasy bazowej i pochodnej

```
<?php  
class KlasaA {  
    function __construct()  
    { print "Konstruktor klasy A <br />";}  
}  
class PodklasaA extends klasaA {  
    function __construct()  
    { parent::__construct();  
        print "Konstruktor podklasyA<br />"; }  
}  
$obj1 = new KlasaA();  
$obj2 = new PodklasaA();  
?>
```

Konstruktor klasy A
Konstruktor klasy A
Konstruktor podklasyA

Składnik static i const

- Deklaracja składników klasy jako statyczne umożliwia odwołanie się do tych składników za pomocą nazwy klasy w przypadku gdy nie istnieje jeszcze żaden obiekt tej klasy
- Składniki statyczne nie mogą być modyfikowane na zewnątrz klasy ani w jej klasach pochodnych
- Deklaracja **static** umieszczana jest po specyfikatorze dostępu
- Metody statyczne są wywoływane na rzecz samej klasy - autoreferencja **\$this** nie może być w nich stosowana
- Do pól statycznych nie można odwoływać się z obiektu za pomocą operatorów: -> i ::
- Nazwa składnika stałego **const** nie zawiera symbolu \$ i tak jak w przypadku składników statycznych, składniki stałe są dostępne poprzez nazwę klasy i nie są dostępne z obiektu poprzez operator ->

Składniki statyczne

```
<?php  
class A {  
    public static $pole_stat='A';  
    public static function  
        funkcja_stat() {  
            echo "Funkcja statyczna ";}  
    public function fa()  
    { return self::$pole_stat; }  
}  
  
class B extends A {  
    public function fb()  
    { return parent::$pole_stat; }  
}  
  
print A::$pole_stat."<br />";  
$a = new A();  
print $a->fa()."<br />";
```

```
print $a->pole_stat."-błąd (warning)  
    <br />";  
//print $a::pole_stat."- błąd (fatal  
error)! <br />";  
print B::$pole_stat."<br />";  
$b = new B();  
print $b->fa()."<br />";  
A::funkcja_stat();    ?>
```

A
A

Notice: Accessing static property A::\$pole_stat as non static in
C:\xampp\htdocs\phpai\przykłady\klasy.php on line **40**

Warning: Undefined property: A::\$pole_stat in
C:\xampp\htdocs\phpai\przykłady\klasy.php on line **40**
- błąd (warning)

A
A
Funkcja statyczna

Składnik stały

```
<?php  
class A {  
    const N=10;  
    function  
    wyswietl_N()  
    { echo self::N."<br  
    />"; }  
}
```

```
echo A::N."<br />";  
$a = new A();  
$a->wyswietl_N();  
echo $a::N; //poprawne od v.5.3  
echo $a->N; //warning  
?>
```

```
10  
10  
10
```

Warning: Undefined property: A::\$N in
C:\xampp\htdocs\phpai\przyklady\klasy.php on line **28**

Klasy abstrakcyjne i interfejsy w PHP

- Słowo kluczowe **abstract** definiuje metodę lub klasę jako abstrakcyjną. Każda klasa zawierająca nawet jedną metodę abstrakcyjną (**abstract**) musi być zdefiniowana jako abstrakcyjna
- Jeśli klasa abstrakcyjna deklaruje metodę jako **protected** to jej konkretna implementacja powinna ją definiować jako **protected** lub **public**
- Interfejs obiektowy (definiowany słowem kluczowym **interface**) umożliwia grupowanie metod abstrakcyjnych, które następnie muszą być zaimplementowane w klasach korzystających z interfejsu
- Klasa, która implementuje dany interfejs (**implements**) musi zawierać definicję wszystkich metod tego interfejsu
- Klasa może **implementować jeden lub więcej** interfejsów

Interfejsy - uwagi

- Wszystkie metody interfejsu muszą być **publiczne** (co wynika z natury samego interfejsu)
- Brak implementacji metody z interfejsu powoduje pojawienie się błędu (fatal error)
- Klasa nie może implementować dwóch interfejsów, które posiadają tak samo nazwane metody
- Interfejsy mogą po sobie dziedziczyć tak jak zwykłe klasy (**extends**)
- Klasa implementująca interfejs musi definiować dokładnie takie same metody jak zadeklarowano w interfejsie - w przeciwnym razie pojawi się błąd (fatal error)
- Interfejs może zawierać składniki stałe, które zachowują własności stałych zwykłych klas - nie mogą być jednak przesłaniane przez klasy/interfejsy po nich dziedziczące

Przykładowe interfejsy

```
interface a
{
    public function fa();
}

interface b
{
    public function fb();
}

interface c extends a, b
{
    public function fc();
}

class d implements c
{
    public function fa()
    {
    }

    public function fb()
    {
    }

    public function fc()
    {
    }
}
```

Przykład - obiektowy licznik

Licznik.php (1)

```
<?php
class Licznik{
private $ile;
private $plik;
function __construct()
{ if (!(file_exists("licznik.txt")))
  { $this->plik=fopen("licznik.txt","w+");
    fputs($this->plik,"0");
    $this->ile=0;
  }
else {
  $this->plik=fopen("licznik.txt","r+");
  if (!$this->plik)
  { echo "Nie da się otworzyć pliku."; exit;
    }
  }
}
function dodaj($ile)
{ $this->ile+=$ile;
  fputs($this->plik,$this->ile);
}
function usun()
{ $this->ile=0;
  fputs($this->plik,"0");
}
function wynik()
{ return $this->ile;
}
}
```

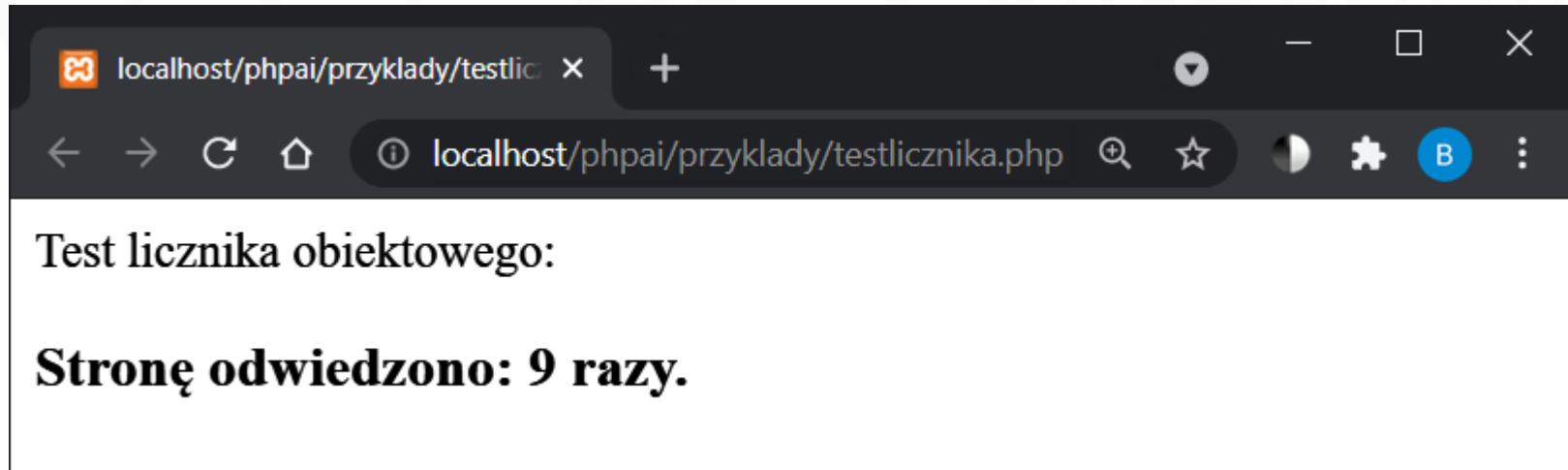
Przykład - obiektowy licznik

Licznik.php (2)

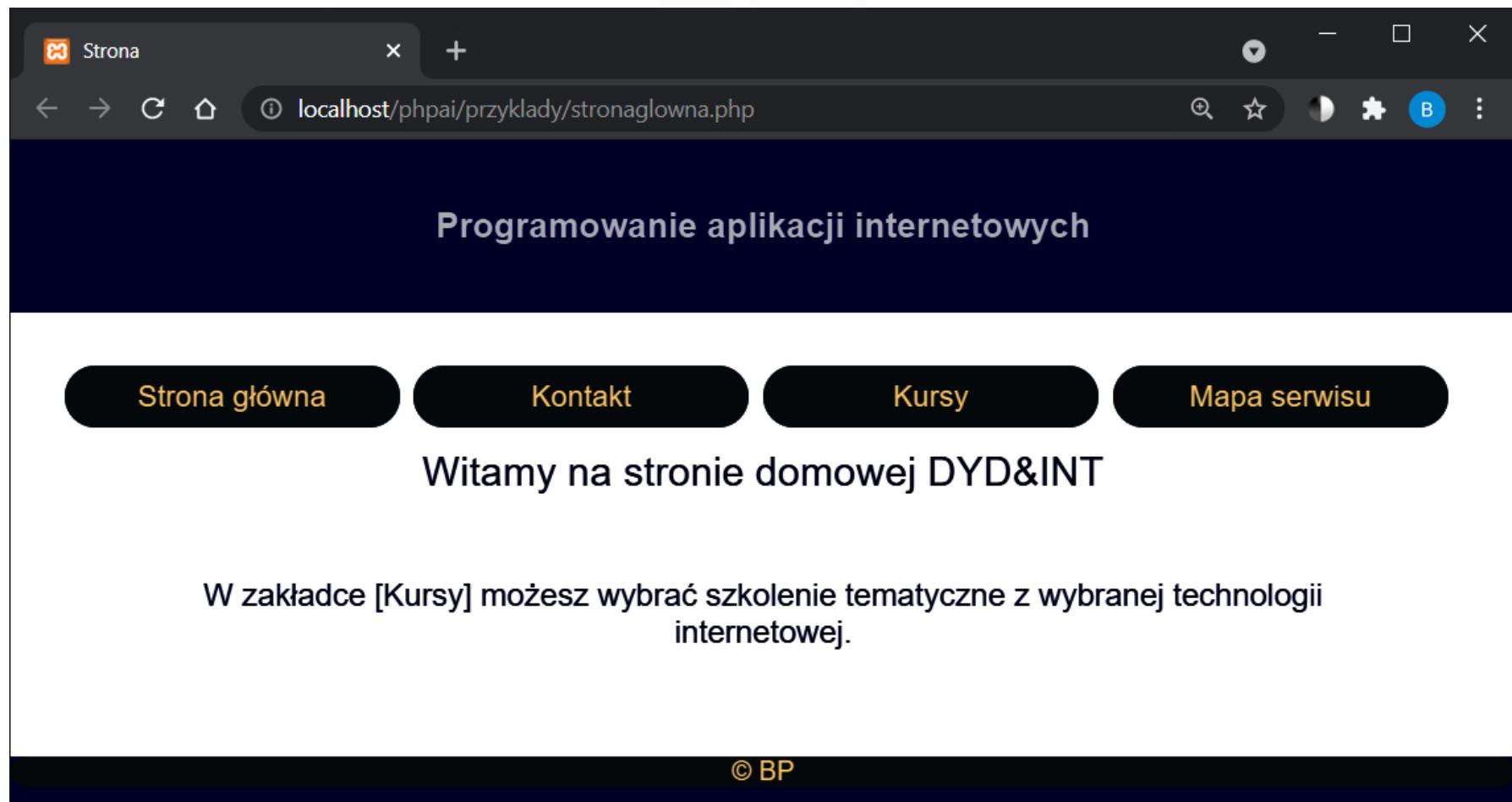
```
flock($this->plik, 2);
$this->ile=fgets($this->plik,255);
$this->ile++;
fseek($this->plik,0);
fputs($this->plik,$this->ile);
flock($this->plik,3);
}
public function __destruct() {fclose($this->plik);}
public function ile_odwiedzin() {return $this->ile;}
} //koniec klasy Licznik
?>
```

Testlicznika.php

```
<?php
    include_once("Licznik.php");
$licznik=new Licznik();
echo "Test licznika obiektowego:";
echo "<h3>Stronę odwiedzono: ".
    $licznik->ile_owiedzin()." razy.</h3>";
?>
```



Przykład - obiektowy szablon strony



Przykład - klasa Strona.php (1)

```
<?php  
class Strona //początek definicji klasy  
{  
    //pola klasy:  
    protected $zawartosc;  
    protected $tytul="Programowanie aplikacji internetowych";  
    protected $przyciski = [  
        "Strona główna"=>"glowna.php",  
        "Kontakt"=>"kontakt.php",  
        "Kursy"=>"kursy.php",  
        "Mapa serwisu"=>"mapa.php"  
    ];
```

Przykład - klasa Strona.php (2)

```
//metody klasy:  
function ustaw_zawartosc($nowa_zawartosc) {  
    $this->zawartosc=$nowa_zawartosc; }  
function ustaw_tytul($nowy_tytul) {  
    $this->tytul=$nowy_tytul; }  
function ustaw_przyciski($nowe_przyciski) {  
    $this->przyciski=$nowe_przyciski; }  
function ustaw_style($url) {  
    echo "<link rel='stylesheet' href='$url' type='text/css' />"; }  
function wyswietl()  
{    $this->wyswietl_naglowek();  
    $this->wyswietl_zawartosc();  
    $this->wyswietl_stopke();  
}  
function wyswietl_menu()  
{ echo "<div id='nav'>";  
foreach ($this->przyciski as $nazwa => $url) {  
    echo "<a href='$url'>$nazwa</a>"; }  
echo "</div>"; }
```

Przykład - klasa Strona.php (3)

```
function wyswietl_naglowek()
{ ?>
<!DOCTYPE html> <html> <head> <title> Strona główna </title>
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width,
    initial-scale=1.0">
<?php $this->ustaw_style('css/bp.css');
    echo "</head><body><div id='nag'><h1>";
    echo $this->tytul."</h1></div>";
}
function wyswietl_zawartosc()
{ echo "<div id='tresc'>";
    $this->wyswietl_menu();
    echo $this->zawartosc."</div>";
}
function wyswietl_stopke()
{ echo "<div id='stopka'> © BP</div></body></html>"; }
} //koniec definicji klasy strona
?>
```

Przykład - stronaglowna.php

```
<?php
require_once("Strona.php");
$strona_glowna = new Strona();
$tresc=<h2> Witamy na stronie domowej DYD&INT</h2>
<h3> W zakładce [Kursy] możesz wybrać szkolenie
tematyczne z wybranej technologii internetowej.
</h3>";
//treść może być też pobrana np. ze wskazanego pliku

$strona_glowna->ustaw_zawartosc($tresc);
$strona_glowna->wyswietl();
?>
```

Przykład - fragment pliku bp.css

```
body {  
    text-align:center; margin: 0;  
    padding: 0; background:#000088;  
    font-size: small; color: #000040;  
    font-family:"Lucida Grande", sans-serif;  
}  
  
h1, h2, h3, h4 {  
    margin: 1em; font-weight: normal;  
    padding: 10px;  
}  
  
h1{  
    font-size:130%; font-weight:bold;  
    color:#fdfdf; padding:15px;  
}
```

Obsługa wyjątków

```
<?php  
function inverse($x) {  
    if (!$x) {throw new Exception('Dzielenie przez zero.')}  
    else return 1/$x; }  
try { echo inverse(5) . "<br>"; }  
catch (Exception $e) {  
echo 'Wyjątek: ', $e->getMessage(), "<br>"; }  
finally { echo "Pierwsza klauzula finally.<br>"; }  
try { echo inverse(0)."<br> "; }  
catch (Exception $e) { echo 'Wyjątek: ',  
$e->getMessage(), "<br>"; }  
finally {  
echo "Druga klauzula  
        finally. <br>"; }  
//... c.d. skryptu  
?>
```

0.2

Pierwsza klauzula finally.
Wyjątek: Dzielenie przez zero.
Druga klauzula finally.



PROGRAMOWANIE APLIKACJI INTERNETOWYCH

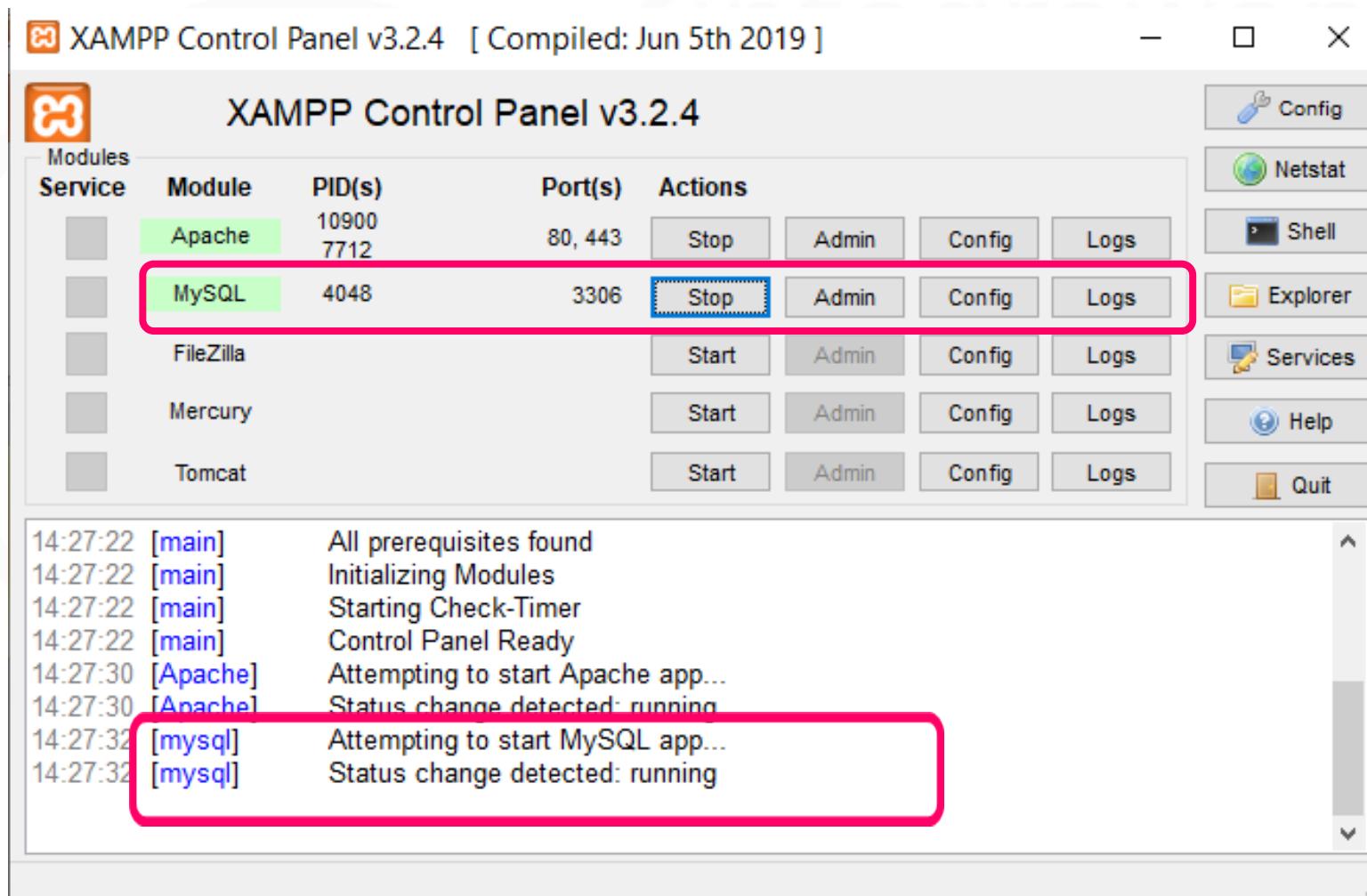
Wykład 5

Współpraca aplikacji internetowej z bazami danych.
Podstawowe zapytania SQL. Serwer MySQL i narzędzie
administrowania bazami danych phpMyAdmin.
PHP i MySQL - interfejs mysqli i PDO.

Beata Pańczyk



Uruchomienie serwera MySQL w XAMPP Control Pane



Narzędzie phpMyAdmin

The screenshot shows the phpMyAdmin interface running on a local server at `localhost / 127.0.0.1 | phpMyAdmin`. The main navigation bar includes links for **Bazy danych**, **SQL**, **Status**, and **Więcej**.

Ustawienia ogólne (General Settings):

- Sortowanie połączenia z serwerem: `utf8mb4_unicode_ci`
- [Więcej ustawień](#)

Ustawienia wyglądu (Appearance Settings):

- Język - Language: `Polski - Polish`
- Konsola: `pmahomme`

Serwer bazy danych (Database Server):

- Serwer: 127.0.0.1 via TCP/IP
- Typ serwera: MariaDB
- Połączenie z serwerem: SSL nie jest używany
- Wersja serwera: 10.4.17-MariaDB - mariadb.org binary distribution
- Wersja protokołu: 10
- Użytkownik: `root@localhost`

Tabela *user* w bazie *mysql*

The screenshot shows the phpMyAdmin interface for the mysql database. The left sidebar lists various tables and objects, with the 'user' table highlighted by a red oval. The main area displays the 'user' table data:

Host	User	Password	Select_priv	Insert_priv
localhost	root		Y	Y
127.0.0.1	root		Y	Y
::1	root		Y	Y
localhost	pma		N	N

The 'User' column is highlighted with a red box. The URL at the bottom of the browser window is `localhost/phpmyadmin/sql.php?server=1&db=mysql&table=user&pos=0`.

Tworzenie nowej bazy dane w phpMyAdmin

The screenshot shows the phpMyAdmin interface on a web browser. The title bar indicates the URL is `localhost / 127.0.0.1 | phpMyAdmin`. The main window is titled "Bazy danych" (Databases) under the heading "Serwer: 127.0.0.1". A red oval highlights the "Utwórz bazę danych" (Create database) button. Below it, a form has "dane" entered in the database name field and "utf8_general_ci" selected in the character set dropdown. The "Utwórz" (Create) button is also highlighted with a red oval. The sidebar on the left lists existing databases: time_zone_name, time_zone_transition, transaction_registry, Widoki, Nowy, user, performance_schema, phpmyadmin, test, and world. The "Bazy danych" tab is active, and the "Metoda porównywania napisów" (Collation) tab is selected in the bottom navigation bar.

Baza danych	Metoda porównywania napisów	Działanie
information_schema	utf8_general_ci	Sprawdź uprawnienia
mysql	utf8mb4_general_ci	Sprawdź uprawnienia
performance_schema	utf8_general_ci	Sprawdź uprawnienia
phpmyadmin	utf8_bin	Sprawdź uprawnienia
Konsola	latin1_swedish_ci	Sprawdź uprawnienia

Dodanie tabeli news do bazy dane

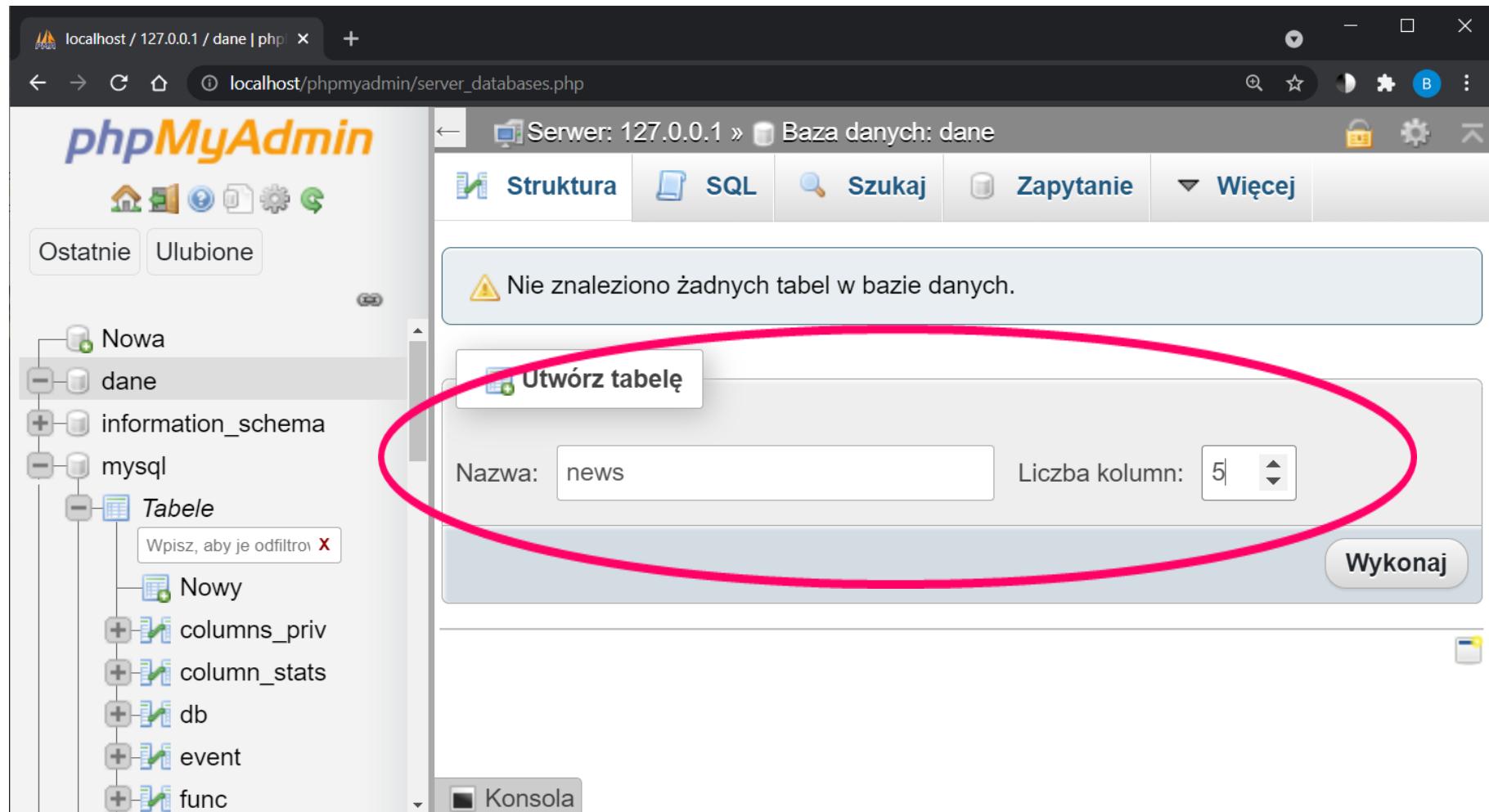


Tabela *news* w bazie *dane*

The screenshot shows the phpMyAdmin interface for a MySQL database. On the left, the database tree is visible with the 'dane' database selected. Inside 'dane', the 'news' table is selected and highlighted with a red circle. The main panel displays the structure of the 'news' table. A red box highlights the first four columns of the table structure: '#', 'Nazwa', 'Typ', and 'Metoda porównywania napisów'. The table has five rows:

#	Nazwa	Typ	Metoda porównywania napisów	Atrybuty	Null	Ustawienia domyślne	Komentarz
1	Lp	int(11)			Nie	Brak	
2	Nagłówek	varchar(100)	utf8_general_ci		Nie	Brak	
3	Treść	text	utf8_general_ci		Nie	Brak	
4	Data	date			Nie	Brak	
5	Autor	varchar(100)	utf8_general_ci		Nie	Brak	

At the bottom of the table, there are buttons for 'Konsola', 'Zaznacz wszystko', 'Z zaznaczonymi:', 'Przeglądaj', and 'Zmień'.

MySQL - zapisywanie danych do bazy

INSERT INTO nazwa_tabeli [(kol1,kol2,...)] VALUES (wart1, wart2,...)

- Wstawienia rekordu do przykładowej tabeli news:

insert into news values

(NULL,'Artykuł1', 'Treść artykułu 1', '2021-07-10','Autor1');

- Wstawienie wartości tylko do wybranych kolumn:

insert into news ('naglowek', 'tresc','autor') values

('Artykuł2', 'Treść artykułu 2','Autor2');

- Wstawianie kilku rekordów:

insert into news values

(NULL,'Artykuł1', 'Treść artykułu 1', '2020-12-10 ','Autor1'),

(NULL,'Artykuł2', 'Treść artykułu 2', '2019-12-10 ','Autor2');

MySQL - wyszukiwanie danych

SELECT pozycje FROM nazwy_tabel

[**WHERE** warunek]

[**GROUP BY** rodzaj_grupowania]

[**HAVING** wartość_funkcji]

[**ORDER BY** porządek_sortowania]

[**LIMIT** limit];

Np.:

- select autor, naglowek from news; //wyświetlenie kolumn autor i nagłówek
- select * from news; //wyświetlenie wszystkich pól, wszystkich rekordów
- select * from news where autor='Autor1';
- select * from news where autor='Autor1' or autor='Autor2';
- select autor from news limit 2,3; //wyszukaj wszystkich autorów i wyświetl
// trzech z nich począwszy od drugiego

MySQL - modyfikacja i usuwanie danych

Modyfikacje tabeli:

- **ALTER TABLE** klienci **MODIFY** nazwisko char(40) not null;
- **ALTER TABLE** zamówienia **ADD** vat float(6.2) after wartosc;
- **ALTER TABLE** zamówienia **DROP** vat

Usuwanie tabeli:

- **DELETE FROM** tabela [**WHERE** warunek] [**LIMIT** ile]
gdzie LIMIT określa max liczbę wierszy do usunięcia
- Np.:

DELETE FROM tabela WHERE id=3;

DELETE FROM ksiazki WHERE id = \$id;

API PHP dla serwera MySQL

- **mysqli Extension** - rozszerzenie dołączane do PHP od wersji 5
<http://si.php.net/manual/en/book.mysql.php>
 - obiektowo zorientowany interfejs
 - wsparcie dla zapytań SQL
 - wsparcie dla złożonych zapytań SQL
 - obsługa transakcji
 - wbudowana obsługa serwera
- **PHP Data Objects (PDO)** - abstrakcyjna warstwa do obsługi baz danych w PHP niezależna od rodzaju serwera BD
<http://si.php.net/manual/en/book pdo.php>
 - używając PDO API można przełączać się pomiędzy serwerami BD wprowadzając jedynie drobne modyfikacje w kodzie PHP
 - pomimo swoich zalet (proste, przenośne, lekkie API) nie pozwala korzystać z niektórych możliwości serwera MySQL
 - driver PDO MYSQL jest jednym z wielu dostępnych driverów PDO (np. drivery dla bazodanowych serwerów Firebird czy PostgreSQL)

Algorytm pracy z BD

- Połączenie z serwerem BD
- Utworzenie/wybranie bazy danych
- Wysłanie zapytania SQL do serwera (dodanie, usunięcie lub modyfikacja danych)
- Odczytanie rezultatów zapytań
- Zamknięcie połączenia z serwerem BD

Współpraca PHP z MySQL - połączenie z serwerem BD i utworzenie nowej BD

```
$servername = "localhost"; $username = "username";
$password = "password";
// Utwórz połączenie z serwerem BD:
$conn = new mysqli($servername, $username, $password);
//Sprawdź połączenie:
if ($conn->connect_error) {
    die("Błąd połączenia: " . $conn->connect_error);
}
// Utwórz BD:
$sql = "CREATE DATABASE nowaBD";
if ($conn->query($sql) === TRUE) { echo "Baza danych utworzona";
}
else { echo "Błąd tworzenia BD: " . $conn->error; }
//Wykonaj operacje na BD . . .
//Zamknij połączenie:
$conn->close();
```

Utworzenie tabeli w BD i dodanie nowego rekordu

```
$sql = "CREATE TABLE  
        MyGuests ( id INT(6) UNSIGNED AUTO_INCREMENT PRIMARY KEY,  
        firstname VARCHAR(30) NOT NULL,  
        lastname VARCHAR(30) NOT NULL,  
        email VARCHAR(50),  
        reg_date TIMESTAMP )";  
if ($conn->query($sql) === TRUE) { echo "Tabela utworzona"; }  
else { echo "Błąd tworzenia tabeli: " . $conn->error; }  
$sql = "INSERT INTO MyGuests (firstname, lastname, email)  
        VALUES ('Ania', 'Annowska', 'ania@gmail.com')";  
if ($conn->query($sql) === TRUE) { echo "Dodano nowy rekord"; }  
else { echo "Błąd dodawania rekordu: $sql <br>" . $conn->error; }
```

Zapytania SELECT, DELETE i UPDATE

```
$sql = "SELECT id, firstname, lastname FROM MyGuests";
$result = $conn->query($sql);
if ($result->num_rows > 0) {
    // wyświetl wyniki wierszami:
    while($row = $result->fetch_assoc()) {
        echo "id: ". $row["id"]." - Name: " .
            $row["firstname"]." ". $row["lastname"]."<br>";
    }
} else { echo "Nie znaleziono rekordów"; }

$sql = "DELETE FROM MyGuests WHERE id=3";
if ($conn->query($sql) === TRUE) { echo "Rekord usunięty"; }
else { echo "Błąd usunięcia rekordu: " . $conn->error; }

$sql = "UPDATE MyGuests SET lastname='Kowalski' WHERE id=2";
if ($conn->query($sql) === TRUE) { echo "Rekord zmodyfikowany"; }
else { echo "Błąd modyfikacji danych: " . $conn->error; }
```

Przykład - praca z mysqli (1)

```
//BD:dane z tabeli news: Lp, Nagłówek, Treść, Data, Autor
$mysqli = new mysqli('localhost', 'root', '', 'dane');
/* sprawdz połączenie */
if ($mysqli->connect_error) {
    printf("Nie udało się połączenie z serwerem: %s\n",
        $mysqli->connect_error);    exit();
}
/* zmien kodowanie na utf8 */
if (!$mysqli->set_charset("utf8")) {
    printf("Error - nie można zmienić kodowania na utf8: %s<br />",
        $mysqli->error);
} else {
    printf("Bieżące kodowanie: %s<br />",
        $mysqli->character_set_name());
}
```

Przykład - praca z mysqli (2)

```
/* wykonaj zapytanie typu select */
if ($result = $mysqli->query("SELECT * FROM news"))
{ printf("W wyniku zapytania otrzymano %d wiersze. <br />",
$result->num_rows);
// pętla po wyniku zapytania $results
printf("<table><tbody>");
while ($row = $result->fetch_object())
{   print("<tr><td class='blue'>");
    printf("%s</td></tr>", $row->Nagłówek);
    printf("<td>Autor: %s", $row->Autor);
    printf("<br> Nadesłano: %s <hr />", $row->Data);
    printf("%s</td></tr>", $row->Treść);
}
printf("</table></tbody>");
$result->close(); /* zwolnij pamięć */
}
$mysqli->close();
```

Przykład - wynik

A screenshot of a web browser window titled "Praca z BD". The address bar shows "localhost/phpai/bd/testbd.php". The page content displays the results of a database query:

Bieżące kodowanie: utf8
W wyniku zapytania otrzymano 3 wiersze.

Artykuł o PHP
Autor: Beata Pańczyk
Nadesłano: 2021-07-14

Treść artykułu na temat podstaw PHP.

Artykuł o systemach operacyjnych
Autor: Maciej Pańczyk
Nadesłano: 2021-07-06

Trochę więcej rzeczy na temat systemów operacyjnych i procesach systemowych.

Szkielet programistyczny Laravel
Autor: Beata Pańczyk
Nadesłano: 2021-07-15

Jak rozpocząć pracę z wykorzystaniem wzorca MVC na przykładzie framework'a Laravel w wersji 8.

Przykład - klasa Baza.php (1)

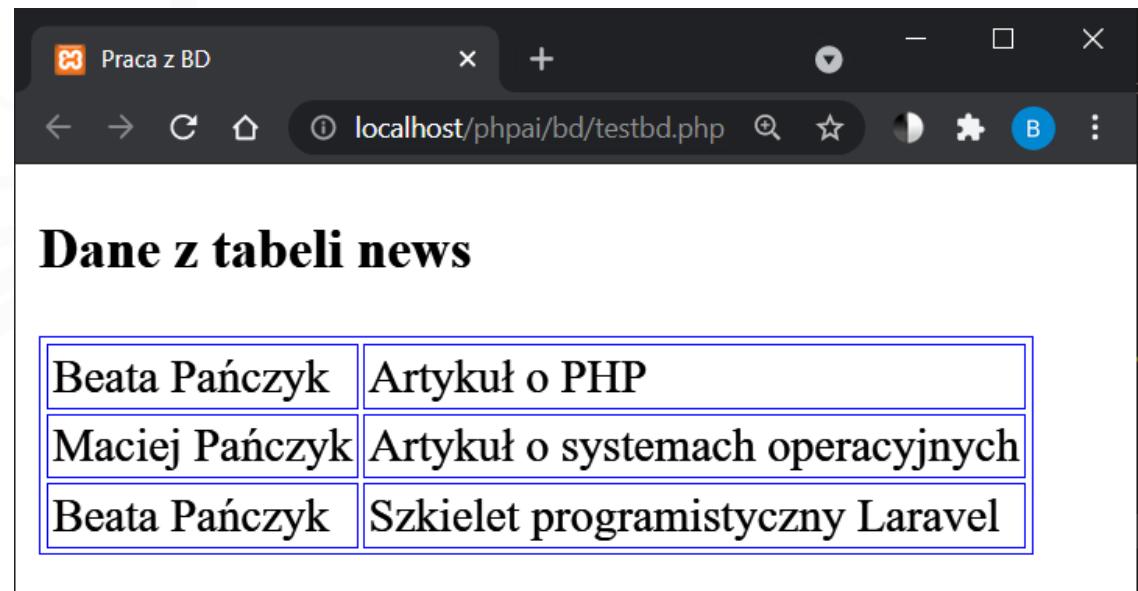
```
class Baza {  
    private $mysqli; //uchwyt do BD  
    public function __construct($serwer, $user, $pass, $baza) {  
        $this->mysqli = new mysqli($serwer, $user, $pass, $baza);  
        if ($this->mysqli->connect_errno) {  
            printf("Nie udało się połączenie z serwerem: %s",  
                $mysqli->connect_error); exit();  
        }  
        if ($this->mysqli->set_charset("utf8")) {  
            //udalo sie zmienic kodowanie  
        }  
    } //koniec funkcji konstruktora  
    function __destruct() { $this->mysqli->close(); }  
    public function insert($sql) {  
        if( $this->mysqli->query($sql)) return true;  
        else return false;  
    }  
    public function getMysqli() { return $this->mysqli; }  
}
```

Przykład - klasa Baza.php (2)

```
public function select($sql, $pola) {  
    //parametr $sql - łańcuch zapytania select,  
    //parametr $pola - tablica z nazwami kolumn w bd  
    //wynik funkcji - gotowy do wyświetlenia kod HTML tabeli z danymi z bd  
    $trecs = "";  
    if ($result = $this->mysqli->query($sql)) {  
        $ilepol = count($pola); //ile kolumn do pokazania  
        $ile = $result->num_rows; //ile rekordów spełnia warunek select  
        $trecs.= "<table>";  
        while ($row = $result->fetch_object()) { $trecs.= "<tr>";  
            for ($i = 0; $i < $ilepol; $i++) {  
                $p = $pola[$i]; $trecs.= "<td>" . $row->$p . "</td>"; }  
            $trecs.= "</tr>";  
        }  
        $trecs.= "</table>"; $result->close(); } //koniec if  
    return $trecs; //gotowa tabela HTML  
} /*koniec metody select */ } /*koniec klasy Baza */ ?>
```

Przykład - zastosowanie klasy Baza

```
<?php  
include_once "klasy/Baza.php";  
$baza = new Baza('localhost', 'root', '', 'dane');  
$sql = "SELECT * FROM news";  
$pola = ["Autor", "Nagłówek"];  
echo "<h3>Dane z tabeli news</h3>";  
echo $baza->select($sql, $pola);  
?>
```



Przykład - formularz dodawania danych do tabeli news w bazie dane

Formularz dodawania x +

localhost/phpai/bd/data_in.php

Dodaj wiadomość:

Autor:

Tytuł:

Wiadomość:

Porównanie aplikacji typu Single Page Application (SPA) i Multi Page Application (MPA).

Przykład - skrypt data_in.php (1)

```
<?php
if (filter_input(INPUT_POST, 'submit')) {
    include_once "klasy/Baza.php";
    $db = new Baza('localhost', 'root', '', 'dane');
    $nagl=filter_input(INPUT_POST, 'nagl', FILTER_SANITIZE_ADD_SLASHES);
    $trecs=filter_input(INPUT_POST, 'trecs', FILTER_SANITIZE_ADD_SLASHES);
    $autor=filter_input(INPUT_POST, 'autor', FILTER_SANITIZE_ADD_SLASHES);
    if (!$autor && $nagl && $trecs)
        echo "Nie wypełniono niektórych pól";
    else {
        $data = (new DateTime())->format("Y-m-d");
        $sql = "INSERT INTO news VALUES
                (NULL, '$nagl', '$trecs', '$data', '$autor')";
        if ($db->insert($sql)) print("<h4> Dane zostały wpisane</h4>");
        else print("<h5> Poniżej możesz dodać kolejną wiadomość </h5>");
    }
}
?>
```

Przykład - skrypt data_in.php (2)

```
<h3>Dodaj wiadomość:</h3>
<p>
<form method="post" action="data_in.php">
    Autor:<br/><input name="autor" size="20"/> <br/>
    Tytuł:<br/><input name="nagl" size="20"/><br/>
    Wiadomość:<br/>
    <textarea cols="25" rows="4" name="tresc" > </textarea>
    <br/>
    <input type="submit" name="submit" value="Dodaj"/>
    <input type="reset" value="Anuluj"/></p>
</form><br />
<a href="data_out.php" > Pokaż dane z bazy News</a>
</p>
```

Przykład - skrypt data_out.php

Dane z bazy

localhost/phpai/bd/data_out.php

Wiadomości z tabeli News

Uporządkuj wiadomości według [dat](#), [nagłówków](#) lub [autorów](#)

lub tylko [zobacz](#) artykuły napisane przez:

[redacted].

2021-07-14 Beata Pańczyk Artykuł o PHP

2021-07-06 Maciej Pańczyk Artykuł o systemach operacyjnych

2021-07-15 Beata Pańczyk Szkielet programistyczny Laravel

2021-07-14 Ola Olewska Aplikacje typu MPA

Przykład - skrypt data_out.php (1)

```
<div>
<h2> Wiadomości z tabeli News</h2>
Uporządkuj wiadomości według
<a href="data_out.php?orderby=data">dat</a>,
<a href="data_out.php?orderby=tytuł">nagłówków</a> lub
<a href="data_out.php?orderby=autor">autorów</a>
<p>
<form method="post" action="data_out.php"> lub tylko
    <input type="submit" name="submit" value="zobacz"/>
    artykuły napisane przez:<input name="autor"/>.
</form>
</p>
```

Przykład - skrypt data_out.php (2)

```
<?php
//ten skrypt pobiera wiadomości z tabeli news bazy dane
include_once "klasy/Baza.php";
$db = new Baza('localhost', 'root', '', 'dane');
$autor = filter_input(INPUT_POST,
                      'autor', FILTER_SANITIZE_ADD_SLASHES);
$orderby = filter_input(INPUT_GET, 'orderby');
if ($orderby == 'data')
    $sql = "select * from news order by Data ";
elseif ($orderby == 'autor')
    $sql = "select * from news order by Autor ";
elseif ($orderby == 'tytuł')
    $sql = "select * from news order by Nagłówek ";
elseif (filter_input(INPUT_POST, 'submit'))
    $sql = "select * from news where Autor='". $autor . "' ";
else $sql = "select * from news";
echo $db->select($sql, ["Data", "Autor", "Nagłówek"]);      ?>
```

Przykład - skrypt data_out.php (modyfikacje)

Fragment kodu:

```
$orderby = filter_input(INPUT_GET, 'orderby');
if ($orderby == 'data')
    $sql = "select * from news order by Data ";
elseif ($orderby == 'autor')
    $sql = "select * from news order by Autor ";
elseif ($orderby == 'tytuł')
    $sql = "select * from news order by Nagłówek ";
```

Można zamienić na (mniej bezpiecznie):

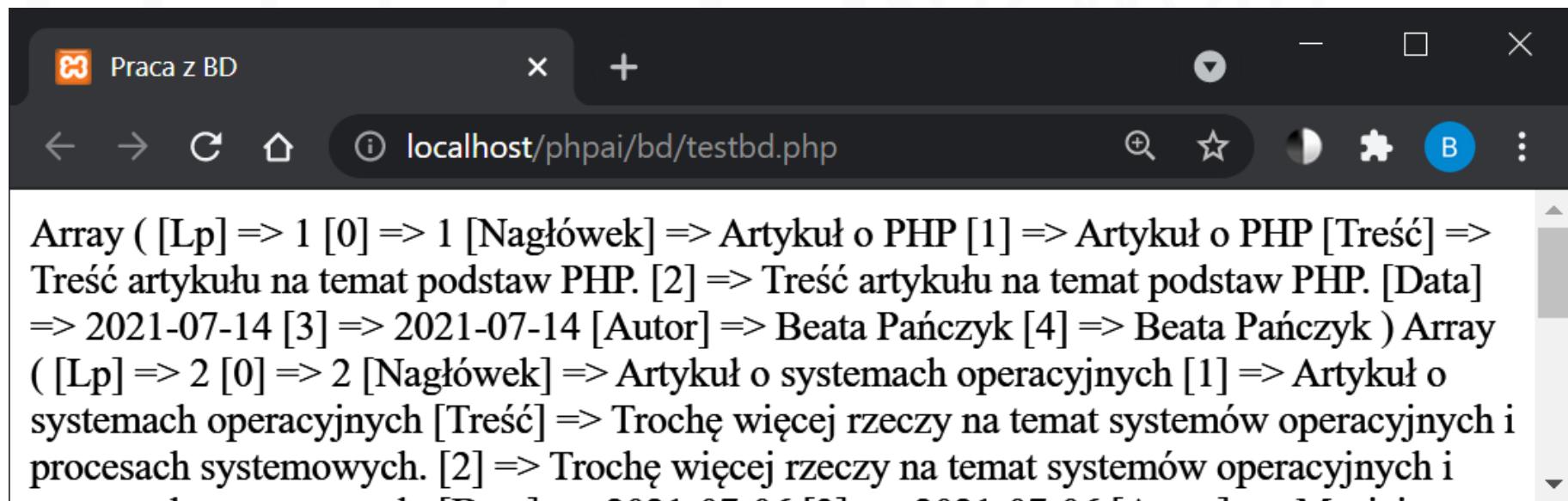
```
$orderby = filter_input(INPUT_GET, 'orderby',
                        FILTER_SANITIZE_ADD_SLASHES);
if ($orderby)
    $sql = "select * from news order by $orderby ";
```

Przykład - klasa Baza_PDO

```
class Baza_PDO {  
    private $dbh; //uchwyt do BD  
    public function __construct($serwer, $user, $pass) {  
        try { $this->dbh = new PDO($serwer, $user, $pass,  
            [PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES utf8"]); }  
        catch (PDOException $e) {  
            print "Error!: " . $e->getMessage() . "<br/>"; die(); }  
    } //koniec funkcji konstruktora  
    function __destruct() { $this->dbh=null; }  
    public function select($sql) {  
        //parametr $sql - łańcuch zapytania select  
        foreach ($this->dbh->query($sql) as $row) {  
            print_r($row); }  
    }  
} //koniec klasy Baza_PDO  
?>
```

Przykład - zastosowanie klasy Baza_PDO

```
include_once "Klasy/Baza_PDO.php";
$baza = new Baza_PDO( 'mysql:host=localhost;dbname=dane',
                      'root', '');
$baza->select('Select * from news');
```



Praca z BD

localhost/phpai/bd/testbd.php

```
Array ( [0] => Array ( [Lp] => 1 [0] => 1 [Nagłówek] => Artykuł o PHP [1] => Artykuł o PHP [Treść] => Treść artykułu na temat podstaw PHP. [2] => Treść artykułu na temat podstaw PHP. [Data] => 2021-07-14 [3] => 2021-07-14 [Autor] => Beata Pańczyk [4] => Beata Pańczyk ) Array ( [0] => Array ( [Lp] => 2 [0] => 2 [Nagłówek] => Artykuł o systemach operacyjnych [1] => Artykuł o systemach operacyjnych [Treść] => Trochę więcej rzeczy na temat systemów operacyjnych i procesach systemowych. [2] => Trochę więcej rzeczy na temat systemów operacyjnych i ) )
```



PROGRAMOWANIE APLIKACJI INTERNETOWYCH

Wykład 6, część 1

Zagrożenia transakcji internetowej. Szyfrowanie i certyfikaty cyfrowe. HTTP jako protokół bezstanowy. Pojęcie, zasada działania i funkcje sesji.

Beata Pańczyk



**Fundusze
Europejskie**
Wiedza Edukacja Rozwój



**Rzeczpospolita
Polska**

Unia Europejska
Europejski Fundusz Społeczny



Zagrożenia transakcji internetowej

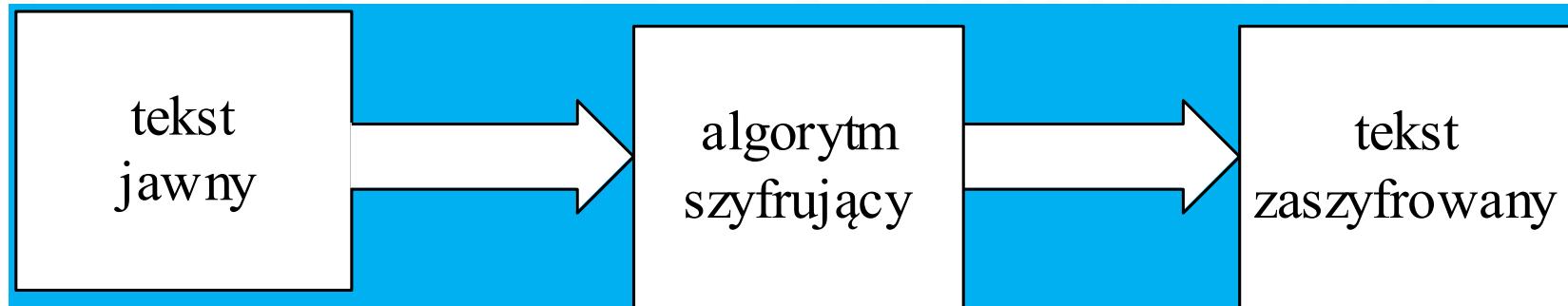
- Zagrożenia:
 - włamanie do komputera klienta
 - podsłuchanie danych na dowolnym odcinku sieci
 - włamanie do serwera dostawcy Internetu
 - włamanie do serwera sklepu internetowego
 - przechwycenie danych w Internecie
 - włamanie do komputera banku lub sklepu internetowego
- Zapewnienie bezpieczeństwa:
 - zapewnienie bezpieczeństwa serwera i zgromadzonych tam danych
 - zabezpieczenie komputera klienta - poziom bezpieczeństwa zależy tylko i wyłącznie od użytkownika i zainstalowanego oprogramowania
 - zapewnienie bezpieczeństwa danych podczas przesyłania informacji przez medium transmisyjne

Bezpieczeństwo aplikacji sieciowych

- **uwierzytelnianie** - weryfikacja użytkowników (zwykle *login* i *password*)
- **autoryzacja** - udostępnianie zasobów systemu wybranej grupie użytkowników
- **poufność** - pewność, że przesyłane dane mogą odczytać określeni (tylko przez nas) użytkownicy
- **integralność** - sprawdzenie czy informacja nie została zmieniona podczas transmisji
- **audyt** - monitorowanie środowiska sieciowego pod kątem ataków, włamań i nadużyć do systemu komputerowego
- **administrowanie** - sprawowanie kontroli nad wymienionymi wyżej zadaniami

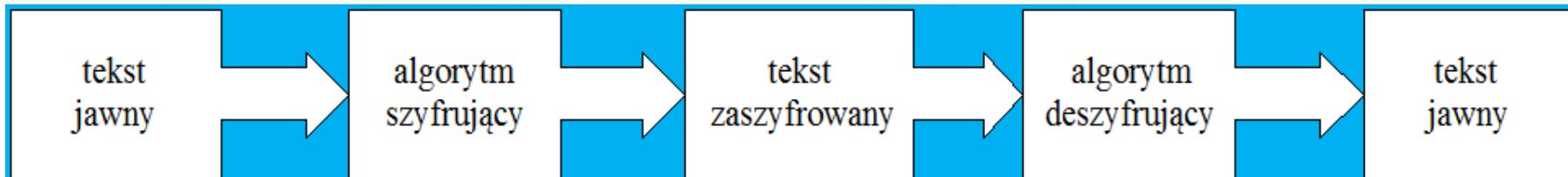
Haszowanie danych

- **Haszowanie** - zamiana dowolnych danych (dane tekstowe, liczbowe, grafika) na pozornie losowy ciąg danych
- Realizowane za pomocą algorytmów haszujących
- Zapewnia zaszyfrowanie danych, nie zapewnia odwrotnego procesu deszyfracji
- Na serwerze odbywa się porównanie zahasowanej postaci hasła wpisanego przez użytkownika z jego zahasowaną postacią znajdującą się w bazie danych



Szyfrowanie danych

- Zapewnione szyfrowanie i deszyfrowanie
- Szyfrowanie kluczem publicznym (symetryczne):
 - szyfrowanie i odczytywanie wiadomości za pomocą tego samego klucza (nadawca i odbiorca wiadomości posiadają ten sam klucz)
 - wada - w przypadku dostania się klucza w niepowołane ręce nie ma gwarancji, że wiadomość będzie przesłana bezpiecznie
- Szyfrowanie asymetryczne:
 - zaszyfrowanie wiadomości następuje za pomocą klucza publicznego dostępnego dla dowolnej osoby, ale jego odszyfrowanie jest możliwe tylko za pomocą klucza prywatnego, przypisanego indywidualnie



Certyfikaty cyfrowe

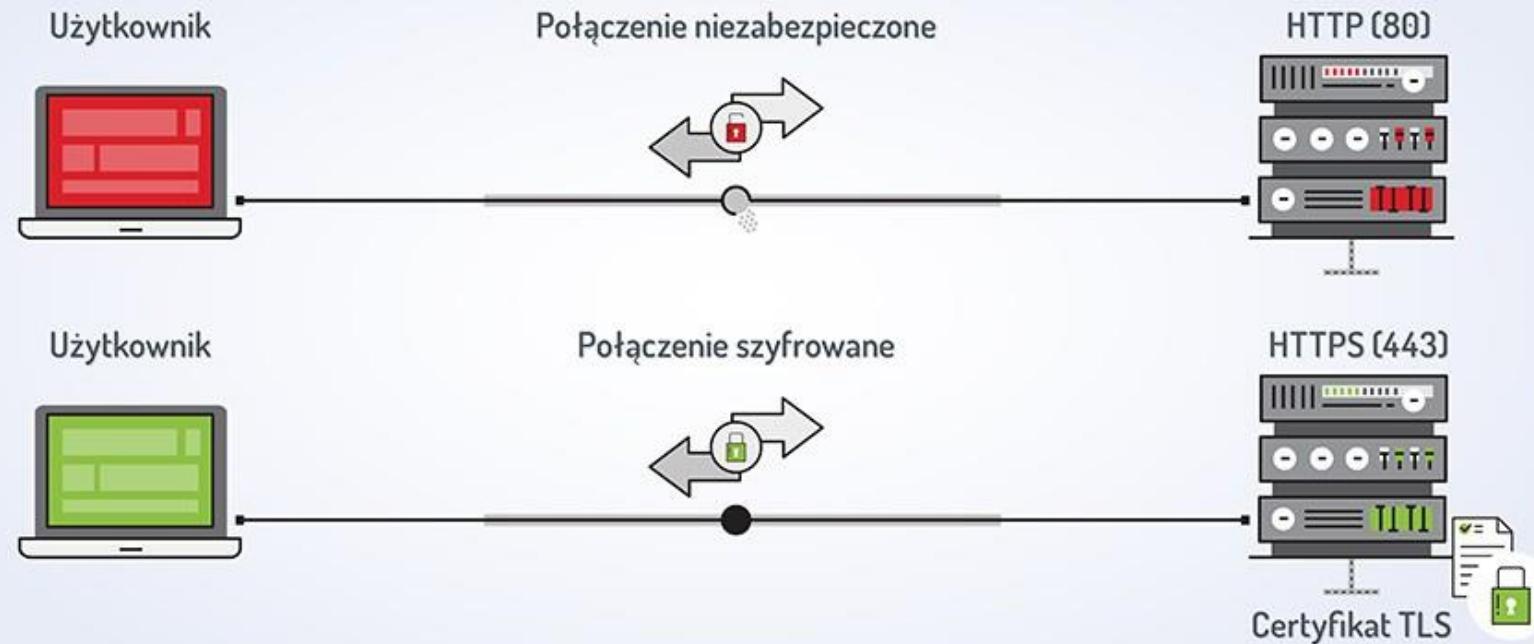
- Zapewniają wiarygodność, autoryzację, poufność i integralność
- Zapewnienie dla klienta, że dana firma spełnia określone wymogi co do szyfrowania danych i ich bezpiecznego przesyłania
- Każdy uczestnik wymiany informacji posiada unikatową parę kluczy (ogólnie dostępny klucz publiczny i tajny klucz prywatny)
- Wystawiony przez samą firmę lub przez niezależne ośrodki zajmujące się tego rodzaju usługami. Certyfikaty przyznawane przez inne firmy zwiększą wiarygodność i poczucie bezpieczeństwa
- Certyfikat nie jest wystarczającym dowodem, że firma prowadzi działalność zgodną z prawem lub, że jest w stanie wypłacić należność za nie dostarczony towar
- Certyfikat pozwala na stwierdzenie, że dana firma zapewnia bezpieczeństwo operacji na pewnym określonym poziomie opisany w certyfikacie

Bezpieczne technologie i protokoły transmisji

- **Protokół SSL** (ang. *Secure Sockets Layer*) i jego rozwinięcie **TLS** (ang. *Transport Layer Security*) przyjęte jako standard w Internecie
- **TLS** zapewnia poufność i integralność transmisji danych, uwierzytelnienie serwera, a czasem również klienta. Opiera się na szyfrowaniu asymetrycznym oraz certyfikatach X.509, działa w warstwie prezentacji, dzięki czemu może zabezpieczać protokoły warstwy najwyższej (warstwy aplikacji, np. HTTP, POP3, IMAP)
- **Protokół HTTPS** (ang. *Hypertext Transfer Protocol Secure*)
 - szyfrowana wersja protokołu HTTP, szyfruje dane za pomocą TLS, co zapobiega ich przechwytywaniu i zmienianiu w trakcie przesyłania
 - działa domyślnie na porcie nr 443
 - wywołania tego protokołu zaczynają się od **https://**

[https://www.eactive.pl/blog-o-seo/protokol-
http-i-protokol-https-czym-sie-roznia/](https://www.eactive.pl/blog-o-seo/protokol-http-i-protokol-https-czym-sie-roznia/)

HTTP CZY HTTPS?



Haszowanie w PHP

- **string md5 (string \$str)**
haszuje ciąg \$str algorytmem MD5 i zwraca串 32 znakowy
- **string sha1 (string \$str)**
haszuje串 \$str algorytmem Secure Hash i zwraca串 40 znakowy
- **string hash(string \$algo, string \$data [, bool \$raw_output = false])**
 - \$algo - algorytm haszujący ("md5", "sha256", "haval160,4", itp.)
 - \$data - dane do haszowania
 - \$row_output - jeśli TRUE, wynikiem jest串 binarny
- Np.:

```
<?php
$str = 'beata'; $pass1= md5($str); $pass2= sha1($str);
echo "$pass1<br />$pass2";
?>
```

```
6aa6a8e3ea1a71a8cce5be0285d3baf8
656b4978460270b2e5bc18cc9c71a3d8ad9d759d
```

Haszowanie haseł i suma kontrolna w PHP

string password_hash(string \$password , int \$algo [, \$options])

- funkcja tworzy nowy skrót hasła przy użyciu silnego jednokierunkowego algorytmu haszującego
- do sprawdzania zahasowanego ciągu stosuje się funkcję **password_verify**

int crc32(string \$str)

- funkcja generuje 32-bitowy ciąg (sumę kontrolną ciągu) na podstawie łańcucha \$str, wykorzystuje się do sprawdzania integralności przesyłanych danych
- w PHP całkowity typ jest definiowany ze znakiem (signed) i aby uzyskać łańcuch reprezentujący nieujemną wartość sumy kontrolnej crc32 należy zastosować format "%u" (dla sprintf() lub printf()) jak pokazano na kolejnym slajdzie

Sprawdzenie sumy kontrolnej

```
<?php  
    //ciag przesyłany:  
    $ciag1="Tworzenie Aplikacji Internetowych w PHP.;"  
    $suma_kontr = crc32($ciag1);  
    //ciag otrzymany:  
    $ciag2="Tworzenie Aplikacji Internetowych w ASP.";  
    if ($suma_kontr==crc32($ciag2))  
        printf("Suma kontrolna %u, OK <br />", $suma_kontr);  
    else  
        printf("Dane w ciągu zostały zmienione!!!");  
?>
```

Dane w ciągu zostały zmienione!!!

Suma kontrolna 1257593352, OK

Przykład - tworzenie tabeli w BD z poziomu PHP - skrypt user.php

```
include_once "klasy/Baza.php";
$db = new Baza('localhost', 'root', '', 'dane');
$sql = "create table user ( name varchar(30) not null,
                           pass varchar(255) not null, primary key (name) );";
if ($db->getMysqli()->query($sql)) {
    echo "Tabela user została utworzona <br>";
    $pass=hash('sha256', 'beata');
    $sql = "insert into user values ('beata', '$pass' );";
    $db->insert($sql);
    echo "Dodano nowego użytkownika <br>";
}
else echo "Nie udało się utworzyć tabeli user <br>";
```

name	pass
-------------	-------------

beata	f691e61ba9a69e1f65984391128656d88749eae94016b451d5...
-------	---

Przykład - proste uwierzytelnianie użytkownika - skrypt haslo.php (1)

```
<?php function drukuj_form() { ?>
<form method="post"
      action="haslo.php">


#### Proszę się zalogować:


```

Proszę się zalogować:

Użytkownik:

Hasło:

Zaloguj się

Przykład - haslo.php (2)

```
/* poczatek skryptu głównego */
if (filter_input(INPUT_POST, 'submit')) {
    include_once "klasy/Baza.php";
    $db = new Baza('localhost', 'root', '', 'dane');
    $login = filter_input(INPUT_POST, 'login',
        FILTER_SANITIZE_ADD_SLASHES);
    $pass = filter_input(INPUT_POST, 'pass',
        FILTER_SANITIZE_ADD_SLASHES);
    $hashpass=hash('sha256', $pass);
    $sql = "select * from user where name='$login' and
            pass='$hashpass'";
    $wynik = $db->getMysqli()->query($sql);
    $ile_rek = $wynik->num_rows;
    if ($ile_rek <> 1) {
        ?> <h4> Niepoprawna nazwa użytkownika lub hasło!
            Spróbuj jeszcze raz</h4>
        <?php    drukuj_form();
    } else { // $ile_rek=1
        echo "<h4> Hasło zaakceptowane!</h4>";    }
    } else drukuj_form();
?>
```

Emulacja sesji HTTP

- HTTP jest protokołem **bezstanowym** i nie może określić, które żądania pochodzą od jednego użytkownika
- Emulowanie sesji dla HTTP - umożliwia obsługę logowania i śledzenia użytkowników
- **Emulacja sesji** HTTP opiera się na założeniu, że serwer aplikacji generuje dla każdego klienta HTTP niepowtarzalny identyfikator sesji
- **Identyfikator sesji** po stronie klienta, jest zwykle zapisywany w zmiennej Cookie (lub dołączany do adresu URL, dzięki czemu powraca do serwera aplikacji przy każdym kolejnym żądaniu HTTP zgłaszanym przez **tego samego klienta HTTP**)
- Po stronie serwera aplikacji znajdują się **pliki** przechowujące tzw. **tablicę sesji**, w której z każdą wartością identyfikatora sesji związany jest zbiór programowych obiektów reprezentujących stan sesji (pliki znajdują się na serwerze np. w folderze xampp/tmp)
- Serwer aplikacji rozpoznaje użytkownika końcowego i udostępnia mu stan sesji wyłącznie **w oparciu o wartość identyfikatora sesji**

Cookie i sesja HTTP

- **Cookie** - niewielki fragment informacji, który skrypty mogą przechować na komputerze klienta
- przeglądarka łącząc się z URL-em poszukuje ciasteczek przechowywanych lokalnie - jeżeli któryś z nich jest odpowiednie dla danego adresu URL to zostaje przekazane serwerowi
- Mechanizm sesji wykorzystuje ciasteczko lub metodę GET do przesłania **id sesji** na serwer
- Jeśli użytkownik akceptuje cookie to są one wykorzystywane do pamiętania identyfikatora sesji, a jeżeli nie akceptuje to jest on dołączany do URL, co zwiększa bezpieczeństwo przechwycenia id sesji przez nieuprawnionego użytkownika
- Ciasteczko jest dostępne dla wszystkich stron **z tej samej witryny** (także po przeładowaniu strony)

Konfiguracja ciasteczek w PHP

Tablica \$_COOKIE

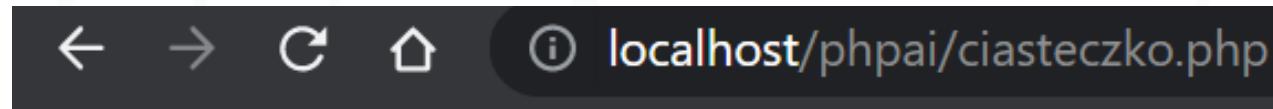
```
bool setcookie(string $name [, string $value = "" [, int $expire = 0 [, string $path = "" [, string $domain = "" [, bool $secure = false [, bool $httponly = false ]]]]]] )
```

- \$name - nazwa ciasteczka, \$value - wartość
- \$expire - czas ważności (np. time() + 60 * 60 * 24 * 30 na 30 dni)
- \$path - ścieżka dostępu do cookie na serwerze
- \$domain - domena, z której cookie jest dostępne
- \$secure - jeśli TRUE to cookie będzie tworzone jeśli istnieje bezpieczne połączenie HTTPS (np. \$httponly - gdy TRUE cookie będzie dostępne tylko przez protokół HTTP a nie będzie dostępne z poziomu JavaScript)
- Np.:

```
setcookie("mojecookie","wartosc"); //ustawienie cookie  
$wartosc = $_COOKIE["mojecookie"]; //odczytanie wartości cookie o nazwie  
setcookie("mojecookie"); //usunięcie cookie  
// jeśli podano parametry URL lub datę wygaśnięcia, należy wysłać te  
// parametry ponownie lub usunięcie nie będzie możliwe).
```

Przykład - praca z ciasteczkami

```
<?php
if (!filter_input(INPUT_COOKIE, 'ile'))
    setcookie('ile', 0);
$ile = filter_input(INPUT_COOKIE, 'ile') + 1;
setcookie('ile', $ile);
echo "Stronę odwiedziłeś $ile razy w tej przeglądarce";
?>
```

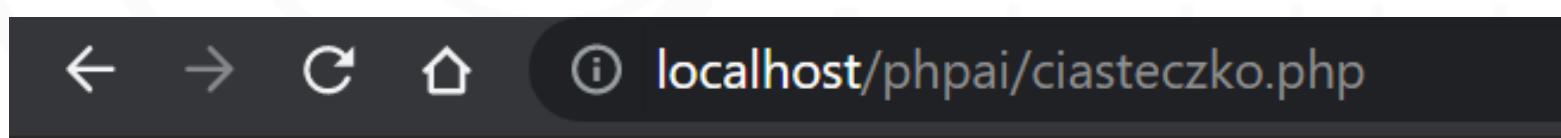


Stronę odwiedziłeś 4 razy w tej przeglądarce

A screenshot of the Chrome DevTools Application tab. The left sidebar shows "Session Storage" with sections for "IndexedDB", "Web SQL", and "Cookies". The "Cookies" section is expanded, showing a table with one row for the domain "http://localhost". The table has columns for "Name" and "Value". The row shows "ile" in the Name column and "4" in the Value column. The "Value" column is highlighted with a blue background.

Przykład - ciasteczko z terminem ważności

```
<?php  
    if (!filter_input(INPUT_COOKIE, 'ile'))  
        setcookie('ile', 0, time() + 24*60*60 );  
    $ile = filter_input(INPUT_COOKIE, 'ile') + 1;  
    setcookie('ile', $ile);  
    echo "Stronę odwiedziłeś dzisiaj: ".$ile." razy.";  
?>
```



Stronę odwiedziłeś dzisiaj: 5 razy.

Cookie W Request i Response

The screenshot shows the Network tab in the Chrome DevTools. A request for 'ciasteczko.php' is selected. The Headers section is active, displaying the following response headers:

- Connection:** Keep-Alive
- Content-Length:** 370
- Content-Type:** text/html; charset=UTF-8
- Date:** Thu, 15 Jul 2021 14:11:05 GMT
- Keep-Alive:** timeout=5, max=100
- Server:** Apache/2.4.46 (Win64) OpenSSL/1.1.1h PHP/8.0.2
- Set-Cookie:** ile=5
- X-Powered-By:** PHP/8.0.2

Below the response headers, the request headers are listed:

- Accept:** text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
- Accept-Encoding:** gzip, deflate, br
- Accept-Language:** pl-PL,pl;q=0.9,en-US;q=0.8,en;q=0.7
- Cache-Control:** max-age=0
- Connection:** keep-alive
- Cookie:** ile=4
- Host:** localhost

Tablica \$_SESSION w PHP

Funkcje do obsługi sesji HTTP

- Sesję HTTP w PHP można rozpoczęć na 3 sposoby:
 - poprzez funkcję **session_start()**
 - przy próbie zgłoszenia zmiennej sesji
 - poprzez skonfigurowanie PHP do automatycznego rozpoczynania sesji **sesion.auto_start** w pliku *php.ini*
- Ustawianie zmiennych sesji w specjalnej tablicy danych sesji:
np. **\$_SESSION['zmienna1'] = 5;**
- **session_id()** - zwraca identyfikator sesji użytkownika
- **session_name()** - zwraca nazwę sesji (i ciasteczko z id sesji jeśli istnieje)
- Korzystanie ze zmiennych sesji po uprzednim sprawdzeniu czy istnieją:
if (isset(\$_SESSION["zmienna"])) { }
- Usuwanie zmiennych sesji:
unset(\$_SESSION['zmienna']);
- Zakończenie sesji (domyślnie w momencie zamknięcia przeglądarki użytkownika) lub po wyczyszczeniu tablicy danych sesji:
\$_SESSION = []; session_destroy();

Przykład - zmienne sesji strona1.php

```
<?php  
session_start();  
$_SESSION["s1"]="Sesja 1";  $_SESSION["s2"]=[10,20,30];  
$_SESSION["s3"]=['nazwisko'=>'Olewski', 'srednia'=>4.51];  
echo "Zawartość zmiennej_sesji: ".$_SESSION["s1"]."<br/>"  
Id=".session_id()."<br/>";  
echo "Zmienna sesji s2[0]=".$_SESSION["s2"][0]."<br/>";  
echo "Zmienna sesji  
s3['srednia']=".$_SESSION["s3"]["srednia"]."<br/>";  
?>  
<a href="strona2.php">  
Zobacz stronę 2</a>
```

Zawartość zmiennej_sesji: Sesja 1
Id=qujakoobe77ht5n4ftcofmdnp7
Zmienna sesji s2[0]=10
Zmienna sesji s3['srednia']=4.51
Zobacz stronę 2

Przykład - zmienne sesji strona2.php

```
<?php session_start();
echo "Zawartość zmiennej_sesji: ".$_SESSION["s1"] .
      "<br/>Id=".session_id()."<br/>";
echo "Zmienna sesji s2[0]=".$_SESSION["s2"][0]."<br/>";
echo "Zmienna sesji s3['srednia']= "
      .$_SESSION["s3"]["srednia"]."<br/>";
unset($_SESSION["s2"]);
?>
<a href="strona3.php">
Zobacz stronę 3</a>
```

UWAGA! Na końcu skryptu 1, zmieniona sesji zostaje zamrożona do czasu jej ponownego pobrania przez wywołanie session_start()

Zawartość zmiennej_sesji: Sesja 1
Id=qujakoobe77ht5n4ftcofmdnp7
Zmienna sesji s2[0]=10
Zmienna sesji s3['srednia']=4.51
[Zobacz stronę 3](#)

Przykład - zmienne sesji strona3.php

```
<?php session_start();
echo "Zawartość zmiennej_sesji: ";
echo $_SESSION["s1"]."<br />Id=".session_id()."<br />";
echo "s2[0]=".$_SESSION["s2"][0]."<br />";
echo "s3['srednia']=".$_SESSION["s3"]["srednia"]."<br/>";

$_SESSION = [];
session_destroy();
echo "Id=".session_id();
?>
<a href="strona1.php">
    Powrót na stronę 1
</a>
```

Zawartość zmiennej_sesji: Sesja 1
Id=qujakoobe77ht5n4ftcofmdnp7

Warning: Undefined array key "s2" in
C:\xampp\htdocs\phpai\sesja\strona3.php on line 17

Warning: Trying to access array offset on value of type null in
C:\xampp\htdocs\phpai\sesja\strona3.php on line 17
s2[0]=
s3['srednia']=4.51
Id= Powrót na stronę 1

Ciasteczko sesyjne PHPSESSID i jego likwidacja

Name	Value	D...	P...	E...	Size	H...
PHPSESSID	qujakoobe77ht5n4ftcof...	lo...	/	S...	35	

```
//Jeśli pożądane jest zabicie sesji, należy usunąć także  
//ciasteczko sesyjne  
//Uwaga: poniższy kod usunie sesję, nie tylko dane sesji:  
if ( isset($_COOKIE[session_name()]) ) {  
    setcookie(session_name(), '', time() - 42000, '/');  
}  
session_destroy();
```

POLITECHNIKA LUBELSKA

WYDZIAŁ ELEKTROTECHNIKI I INFORMATYKI

INFORMATYKA



Zintegrowany
Program
Rozwoju
Politechniki
Lubelskiej -
część druga

PROGRAMOWANIE APLIKACJI INTERNETOWYCH

Wykład 6, część 2 Uwierzytelnianie w kontroli sesji

Beata Pańczyk



**Fundusze
Europejskie**
Wiedza Edukacja Rozwój



**Rzeczpospolita
Polska**

Unia Europejska
Europejski Fundusz Społeczny



Umieszczanie obiektu w sesji - klasa Produkt

```
class Produkt {  
    private $id;  
    private $name;  
    private $cena;  
    public function __construct($id=1,$name="Xxx",$cena=1) {  
        $this->id = $id; $this->name=$name; $this->cena=$cena;  
    }  
    function getId() { return $this->id; }  
    function setId($id) { $this->id = $id; }  
    //pozostałe funkcje set/get  
    function show(){  
        echo $this->id." ".$this->name.", cena:".$this->cena;  
    }  
}
```

Serializacja i deserializacja obiektów

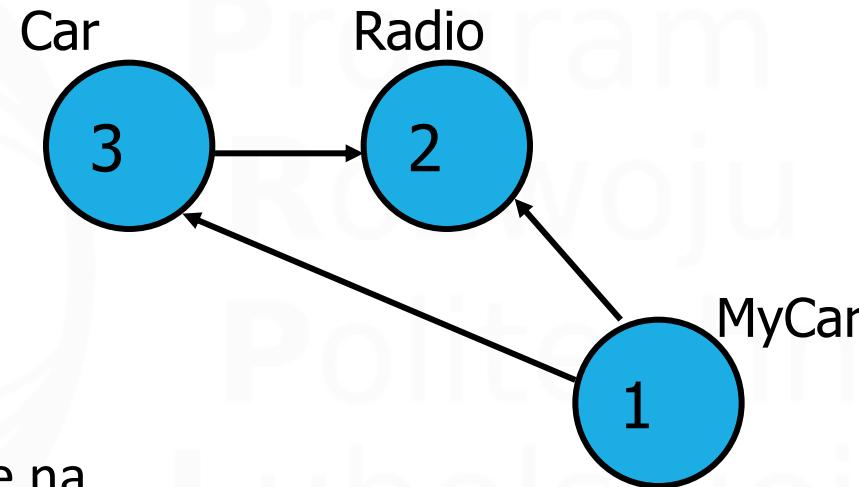
- **Serializacja obiektów** to proces przekształcania obiektów (instancji określonych klas), do postaci szeregowej (liniowej), czyli w strumień bajtów, z zachowaniem aktualnego stanu obiektu. Serializowany obiekt może zostać utrwalony w postaci pliku na dysku, przesłany do innego procesu lub innego komputera poprzez sieć
- **Postać szeregowa** zawiera wszystkie informacje niezbędne do zrekonstruowania (deserializacji) stanu obiektu i późniejszego wykorzystania tych informacji
- **Deserializacja** to proces odwrotny do serializacji, polega na wczytaniu utrwalonych danych z określonej lokalizacji i odtworzeniu obiektu wraz z jego stanem bezpośrednio przed serializacją
- Mechanizm serializacji jest używany między innymi na platformach .NET, Java, PHP, Python, Ruby

Graf obiektów

- Graf obiektów (ang. object graph) - łańcuch powiązanych ze sobą obiektów serializowanych do strumienia
- To prosty sposób dokumentowania wzajemnych związków między obiektami, ale nie w rozumieniu klasycznych wzajemności obiektowych (czyli relacji 'jest':is-a, 'zawiera':has-a)
- Każdemu obiekowi przyporządkowuje się unikatową wartość liczbową, a po niej graf wszystkich powiązanych z tym obiektem elementów (numery przypisane elementom grafu obiektów są przypadkowe i nie mają żadnego znaczenia na zewnątrz)

Przykład - graf obiektów

- Zbiór klas reprezentuje modele samochodów - klasa bazowa Car jest związana relacją ‘zawiera’ z klasą Radio. Klasa o nazwie MyCar dziedziczy po Car
- Graf obiektów modelujący te zależności ma postać:



- Zależności przedstawione na diagramie można wyrazić wzorem:
[Car 3, ref 2], [Radio 2], [MyCar 1, ref 3, ref 2]
- Serializując instancje typu MyCar, dzięki grafowi obiektów typy Radio i Car zostaną w tym procesie uwzględnione (graf jest tworzony automatycznie bez udziału programisty!)

Serializacja obiektów w PHP

serialize(mixed \$value) : string

- generuje ciąg reprezentujący dane z zachowaniem informacji o ich typie i strukturze, co jest bardzo przydatne do przechowywania lub przekazywania takich danych

unserialize(string \$str [, array \$options]) : mixed

- przyjmuje pojedynczą zmienną serializowaną (pierwszy parametr w postaci łańcucha) i konwertuje ją z powrotem na wartość PHP

Przykład - serialize.php

Serializacja obiektu

```
<?php
session_start();
include 'Produkt.php';
$p1 = new Produkt();
echo "<p> Właściwości obiektu p1: <br />";
$p1->show();
echo "</p>";
echo "<p>Łańcuch po serializacji obiektu:
<br />".serialize($p1)." </p>";
//serializowany obiekt dodajemy do sesji:
$_SESSION['p1'] = serialize($p1);
?>
<p><a href="unserialize.php" >Strona2</a> </p>
```

Przykład - widok w przeglądarce (serialize.php)

Właściwości obiektu p1:
1 Xxx, cena:1

Łańcuch po serializacji obiektu:
O:7:"Produkt":3:
{s:11:"Produktid";i:1;s:13:"Produktnname";s:3:"Xxx";s:13:"Produktcena";i:1;}

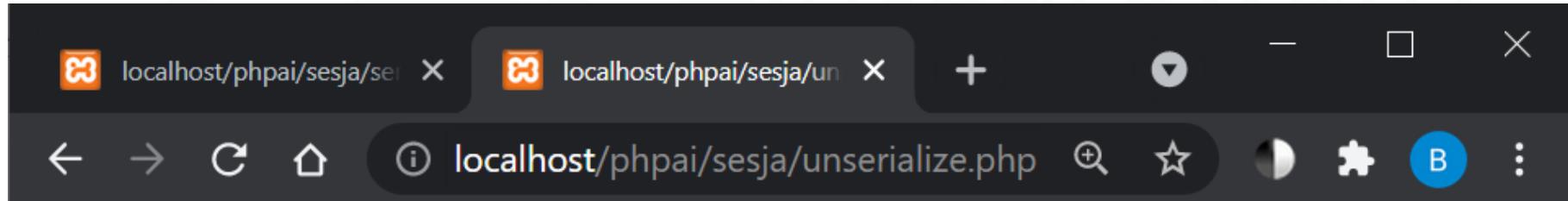
[Strona2](#)

Przykład - unserialize.php

Deserializacja obiektu

```
<?php
session_start();
include 'Produkt.php';
//sprawdź czy w sesji istnieje element o kluczu "p1":
if (isset($_SESSION['p1'])) {
    $p1_z_sesji=$_SESSION['p1'];
    echo "<p>Wartość elementu o kluczu 'p1' z sesji: <br />".
                      $p1_z_sesji. " </p>";
    $p1 = unserialize($_SESSION['p1']);
    echo "<p>Obiekt po odtworzeniu (deserializacji): <br />";
    $p1->show();
    echo "</p>";
}
else { echo "Brak obiektu w sesji";  }
?>
```

Przykład - widok w przeglądarce (unserialize.php)



The screenshot shows a web browser window with two tabs open. The tabs are both labeled with the URL `localhost/phpai/sesja/unserialize.php`. The browser interface includes standard controls like back, forward, and search, along with a blue button labeled 'B'.

Wartość elementu o kluczu 'p1' z sesji:
O:7:"Produkt":3:
{s:11:"Produktid";i:1;s:13:"Produktnname";s:3:"Xxx";s:13:"Produktcena";i:1;}

Obiekt po odtworzeniu (deserializacji):
1 Xxx, cena:1

Konfiguracja kontroli sesji w php.ini

Opcja	Znaczenie
session.auto_start	Automatycznie rozpoczyna sesję (0)
session.cache_expire	Czas przechowania sesji w min (180)
session.use_cookie	Stosowanie cookies po stronie klienta (1)
session.use_only_cookie	Korzystanie tylko z cookies w sesji (0)
session.cookie_path	Ścieżka ustawiana w cookie sesji
session.name	Nazwa sesji (cookie) w komputerze użytkownika ("PHPSESSID")
session.save_handler	Typ miejsca przech. danych (files)
session.save_path	Miejsce przechowywania danych

Przykład - przygotowanie bazy danych i tabeli users

- Przykład pokaże prosty mechanizm uwierzytelniania użytkownika na podstawie danych loginu i hasła przechowywanych w bazie danych
- Zakładamy, że na serwerze MySQL istnieje baza danych o nazwie **test** (jeśli nie to należy ją utworzyć)
- Korzystając z zakładki SQL w phpMyAdmin do bazy tej dodano tabelę użytkowników, wpisując w okienku SQL polecenie:
`create table users (
 name varchar(20) not null,
 pass varchar(255) not null,
 primary key (name)
);`

Przykład - dodanie przykładowych danych do tabeli users (add.php)

```
<?php  
include_once "Baza.php";  
$db = new Baza('localhost', 'root', '', 'test');  
$pass=hash('sha256', 'madzia');  
$sql = "insert into users values ('madzia', '$pass' );";  
$db->insert($sql);  
$pass=hash('sha256', 'ania');  
$sql = "insert into users values ('ania', '$pass' );";  
$db->insert($sql);
```

name	pass
ania	15ca7c08d604be43364be5a39513195d4862cd16852c5d0361...
madzia	29cfe16428a2dd1094d9b7a66578c445021ea8b1906e930bf7...

Przykład - uwierzytelnianie w kontroli sesji

Strona główna

Zaloguj się:

Name:

Password:

Login

Anuluj

[Tylko uprawnieni](#)

Informacje tylko dla uprawnionych

Użytkownik niezalogowany

Tylko zalogowani użytkownicy

[Powrót do strony głównej](#)

Strona główna

Użytkownik zalogowany jako:madzia

[Wyloguj](#)

[Tylko uprawnieni](#)

Informacje tylko dla uprawnionych

Użytkownik zalogowany jako:madzia

Informacje tylko dla zalogowanych użytkowników ...

[Powrót do strony głównej](#)

Przykład - klasa UserLog.php (1)

```
class UserLog {  
    private $name;  
    private $pass;  
    private $log = false;  
    function __construct($name, $pass) {  
        $this->name=$name; $this->pass=$pass; }  
    public function getName() { return $this->name; }  
    public function logout() {  
        $this->log = false;  
        $_SESSION = [];  
        if (filter_input(INPUT_COOKIE, session_name()) ) {  
            setcookie(session_name(), '', time() - 42000, '/'); }  
        session_destroy();  
    }  
}
```

Przykład - klasa UserLog.php (2)

```
static function loginForm($link)
// $link – wartość atrybutu action dla formularza logowania
{
    echo '<div><h3>Zaloguj się:</h3>';
    echo '<form action="'. $link .'" method="post" />';
    ?> Name: <br/><input type="text" name="name" /><br />
        Password: <br/><input type="password" name="pass" />
        <br />
        <input type="submit" name="login" value="Login" />
        <input type="reset" value="Anuluj" />
    </form>
<?php }
```

Przykład - klasa UserLog.php (3)

```
static function login($name, $pass, $bd, $table)
{ // $bd - uchwyt do BD, $table - nazwa tabeli z użytkownikami w bd
    $user = null;
    if (($name!=="") && ($pass!="")) {
        $sql="select * from $table where name='$name' and pass='$pass'";
        if ($result = $bd->getMysqli()->query($sql)) {
            $ile = $result->num_rows; // ile wierszy
            if ($ile == 1) { // użytkownik zalogowany
                $user = new UserLog($name, $pass);
                $user->log = true;
                $_SESSION["userOK"] = serialize($user);
            }
            $result->close(); /* zwolnij pamięć */
        }
    }
    return $user;
}
```

Przykład - skrypt loginForm.php

```
<?php
require_once "Baza.php";  require_once "UserLog.php";
session_start();
if (filter_input(INPUT_POST, "login")) {
    $bd = new Baza("localhost", "root", "", "test");
    $name = filter_input(INPUT_POST, "name",
                         FILTER_SANITIZE_ADD_SLASHES);
    $pass = hash("sha256", filter_input(INPUT_POST, "pass"));
    $user = UserLog::login($name, $pass, $bd, 'users');
    if ($user == null) echo "Błąd logowania";
}
?>
<h2>Strona główna</h2>
<?php
if (isset($_SESSION["userOK"])) {
    $user=unserialize($_SESSION['userOK']);
    echo "Użytkownik zalogowany jako: ".$user->getName()."<br />";
    echo "<a href='wyloguj.php' >Wyloguj</a><br />";
}
else UserLog::loginForm("#");
?>
<p> <a href="uprawnieni.php">Tylko uprawnieni</a></p>
```

Przykład - skrypt uprawnieni.php

```
<?php
    require_once "UserLog.php";
    session_start();
    echo "<h2>Informacje tylko dla uprawnionych</h2>";
    //sprawdzenie zmiennej sesji
    if (isset($_SESSION["userOK"])) {
        $user = unserialize($_SESSION["userOK"]);
        echo "Użytkownik zalogowany jako:" . $user->getName(). "<br/>";
        echo "Informacje tylko dla zalogowanych użytkowników ...";
    } else {
        echo "<h2>Użytkownik niezalogowany</h2>";
        echo "Tylko zalogowani użytkownicy mogą oglądać tę stronę";
    }
    echo "<br /><a href=\"loginForm.php\">
          Powrót do strony głównej</a>";
?
}
```

Przykład - wyloguj.php

```
<?php
    require_once "UserLog.php";
    session_start();
    if (isset($_SESSION["userOK"])) {
        $old_user = unserialize($_SESSION["userOK"]); //czy było logowanie
        if ($old_user != null) {
            echo "<p>Wylogowano użytkownika:". $old_user->getName(). "</p>";
            unset($_SESSION["userOK"]);
            $old_user->logout();
        }
    }
    else    echo "Użytkownik niezalogowany.<br />";
?>
<p><a href="loginForm.php"> Powrót do strony logowania </a></p>
```



PROGRAMOWANIE APLIKACJI INTERNETOWYCH

Wykład 7

Strategia tworzenia bezpiecznych aplikacji internetowych.
Rodzaje ataków sieciowych. Metody zabezpieczania aplikacji
i ich implementacja w PHP.

Beata Pańczyk



**Fundusze
Europejskie**
Wiedza Edukacja Rozwój



**Rzeczpospolita
Polska**

Unia Europejska
Europejski Fundusz Społeczny



Strategie tworzenia bezpiecznych aplikacji

- W celu stworzenia bezpiecznego połączenia pomiędzy komputerami stosuje się protokół HTTPS (korzystający z TLS), który zapewnia szyfrowanie na zadowalającym poziomie
- Dodatkowe zabezpieczenia (ang. Defense in Depth) - duplikowanie się różnych zabezpieczeń (np. ponowne uwierzytelnianie przed wykonaniem istotnych akcji)
- Minimum uprawnień dla użytkowników
- Minimalizowanie ekspozycji na zagrożenia
- Najlepsze rozwiązania to najprostsze rozwiązania
- Dbanie o równowagę pomiędzy użytecznością a minimalizacją ryzyka (rozwiązania przyjazne dla użytkownika i bezpieczne nie muszą się wzajemnie wykluczać)
- Śledzenie przepływu danych - sprawdzanie ich wartości i ustalanie gdzie się aktualnie znajdują i skąd pochodzą
- Filtrowanie i walidacja przychodzących danych
- Dodawanie znaków \ w danych wyjściowych (np. zapisywanych do bd)

Ukrywanie kodu

- Jeśli potencjalny włamywacz nie wie co skrypt robi, jaka jest architektura bazy danych - może tylko zgadywać, co jest czasochłonne i bezcelowe
- Jeśli serwery WWW i PHP pracują prawidłowo to **nie ma możliwości** podejrzenia kodu PHP strony poprzez stronę internetową
- Jeśli serwer WWW nieprawidłowo zainstaluje PHP to może zamiast wykonania skryptu wyświetlić jego kod - należy zadbać by pliki zawierające login i hasło do bazy danych nie były dostępne z poziomu przeglądarki a do skryptu dołączać je poprzez **include**
- Np.:
`<?php include_once("../..../hasla.php"); ?>`
Tak dołączony plik nie będzie dostępny z poziomu adresu URL w przypadku nieprawidłowego działania serwera WWW lub PHP

Kluczowe pliki - poza katalogiem głównym

- Kontrola przechowywanych plików i wysyłanych informacji
- Użytkownicy **nie mają dostępu** do plików spoza katalogu głównego serwera WWW
- Folderem bieżącym jest folder główny serwera np. **c:\xampp\htdocs** (i jest to najwyższy poziom dostępny dla odwiedzających)
- W pliku konfiguracyjnym ***php.ini*** można sprawdzić ustawienia dla zmiennej **doc_root** (np. **doc_root = "c:\xampp\htdocs"**) i w odwołaniach do plików podawać ścieżkę względną:
\$sciezka = \$doc_root."..\\Wyslane\\".\$plik;
(folder **Wyslane** znajduje się na tym samym poziomie co folder **htdocs**, ale jest poza zasięgiem użytkownika)

Uwaga na pliki dostępne publicznie

- Ogólnie dostępne są pliki w katalogu np. **htdocs**
- Niebezpieczne rozszerzenia np. **.txt**:
 - w pliku db.txt - zapisano dane do połączenia z bazą danych
 - include_once "db.txt" - wykorzystane w każdym skrypcie pracującym z BD
 - serwer posiada domenę np. www.serwer.pl
 - **http://www.serwer.pl/db.txt
(wyświetlony kod źródłowy!!!)**
- Rozwiążanie - zaznaczenie pliku jako **db.php** a jeszcze lepiej umieszczenie pliku z hasłami do bazy poza katalogiem głównym serwera WWW

Ukrywanie PHP

- Ustawienie w pliku ***php.ini*** opcji **expose_php On** - serwer WWW zgłasza obecność PHP przy każdym zapytaniu. Ustawienie **expose_php Off** ukrywa wykorzystanie PHP w większości przypadków, jeżeli jednak skrypt generuje komunikat o błędzie to fakt ten jest natychmiast zgłoszany
- Rzeczywiste ukrycie PHP - wymuszenie ukrywania komunikatów o błędach przez ustawienie:
display_errors Off (utrudni proces debugowania),
przy jednoczesnym ustawieniu
log_errors On (możliwa jest analiza błędów z dziennika błędów z pliku np. ***php/logs/php_error_log***)
- Ustawienie funkcji ***error_reporting(0)*** w skrypcie PHP spowoduje, że nie zostanie wyświetlony żaden błąd w trakcie wykonania skryptu (jeśli zaistnieje)

Ataki CSS i HTML injection

- Ataki typu Cross-Site Scripting (XSS, CSS) zaliczają się do bardzo groźnych, polegają na dodaniu kodu HTML czy JavaScript wykonywanego po stronie odbiorcy strony (nie serwera)
- Ataki tego typu najczęściej polegają na kradzieży zmiennych Cookie z komputera ofiary poprzez wykonanie na nim złośliwego kodu JavaScript
- Skrypt jest wrażliwy na tego typu ataki jeżeli nie sprawdza dokładnie jakie dane wprowadził użytkownik (np. posty na forum dyskusyjnym)
- **Zabezpieczenie to filtrowanie i walidacja danych**, w PHP:
 - **htmlspecialchars/FILTER_SANITIZE_FULL_SPECIAL_CHARS** - konwertuje znaki specjalne HTML na bezpieczne encje
 - **strip_tags/FILTER_SANITIZE_STRING** - usuwa z łańcucha tagi HTML i PHP

Atak typu Directory Traversal

- Umożliwia dostęp do katalogów, do których dostępu być nie powinno
- Luka częsta w skryptach umożliwiających download plików, na przykład w galeriach listujących zawartość katalogów
- <http://www.php.rk.edu.pl/w/p/luki-bezpieczenstwa-w-skryptach-php/>
- **Zabezpieczenie:**
 - standardowo **strip_tags** (lub **FILTER_SANITIZE_STRING**)
 - uniemożliwienie w skrypcie stosowania znaków ../ w ścieżce:

```
if( strstr('../', $_GET['patch']))  
{  
    die('Strona odporna na atak');  
}
```

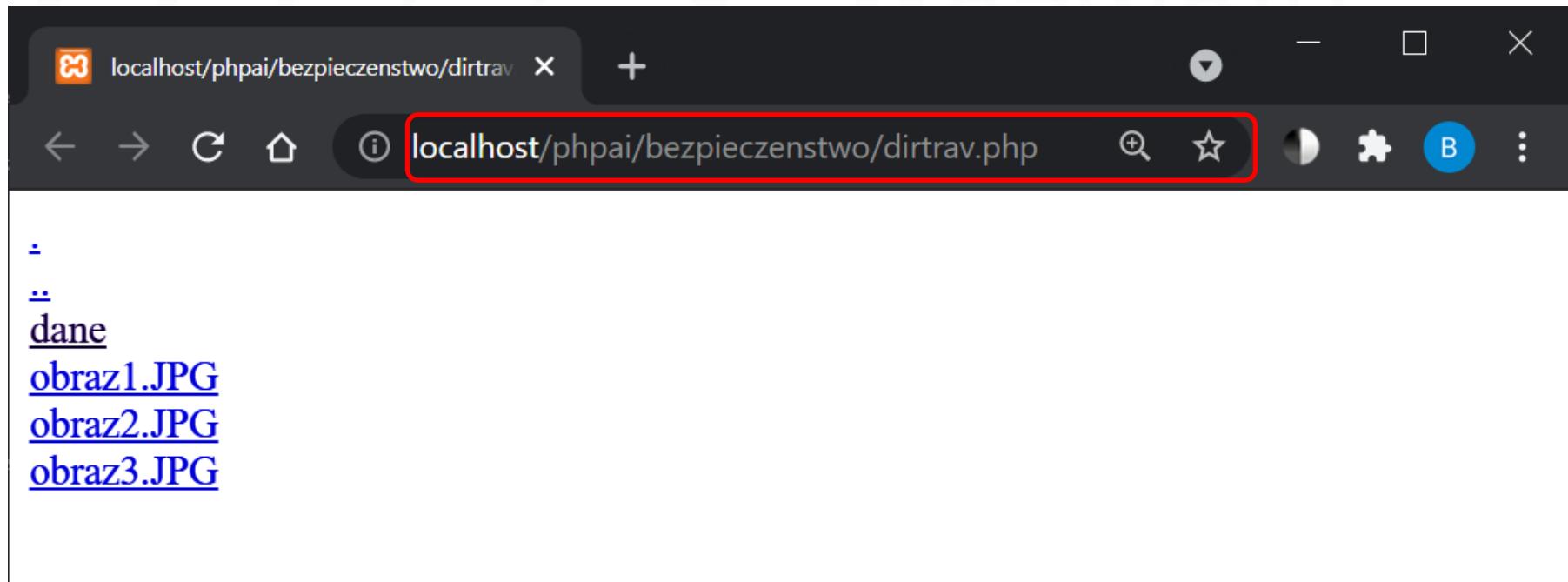
Przykład - atak Directory Traversal

```
function download($patch = 'mini')
{ $katalog=@dir($patch) or die ('Brak dostępu do katalogu.');
while ($plik_kat = $katalog->read())
    if(is_file($patch.'/'.$plik_kat))
        echo '<a href="'. $patch.'/'.$plik_kat.'"> '.$plik_kat.
              '</a><br />';
    elseif (is_dir($patch.'/'.$plik_kat))
    { echo '<a href="dirtrav.php?patch='.$patch.'/'.$
          $plik_kat.'">'.$plik_kat.'</a><br />';
        //echo '- <a href="'. $patch.'/'.$plik_kat.'">
        //<b>'.$plik_kat.'</b></a><br />';
    }
    $katalog->close();
}
```

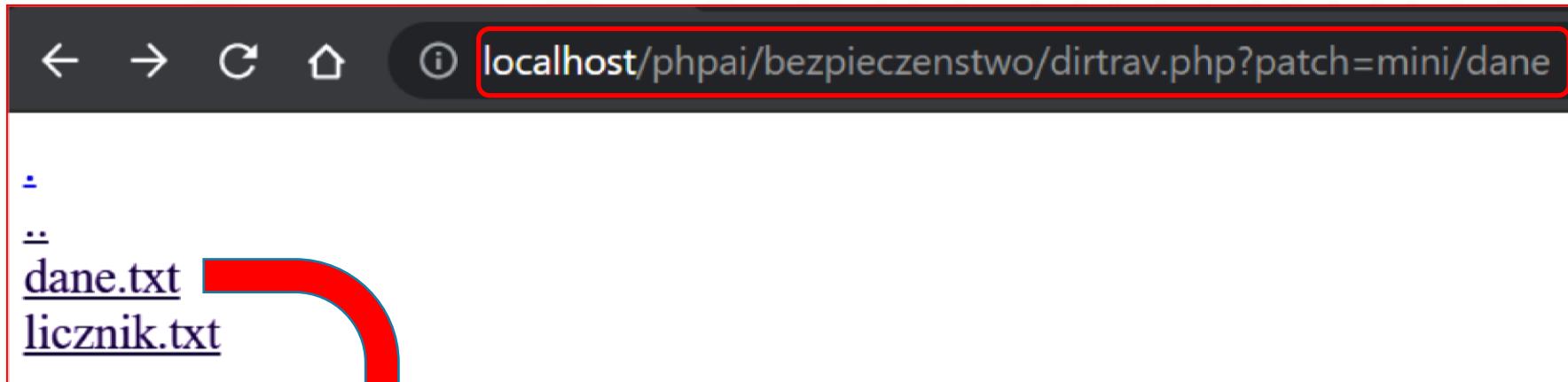
Directory Traversal - parametr przekazany przez GET

//skrypt wywołujący funkcję download:

```
If (!isset($_GET['patch'])) { download(); }
else { download($_GET['patch']); }
```

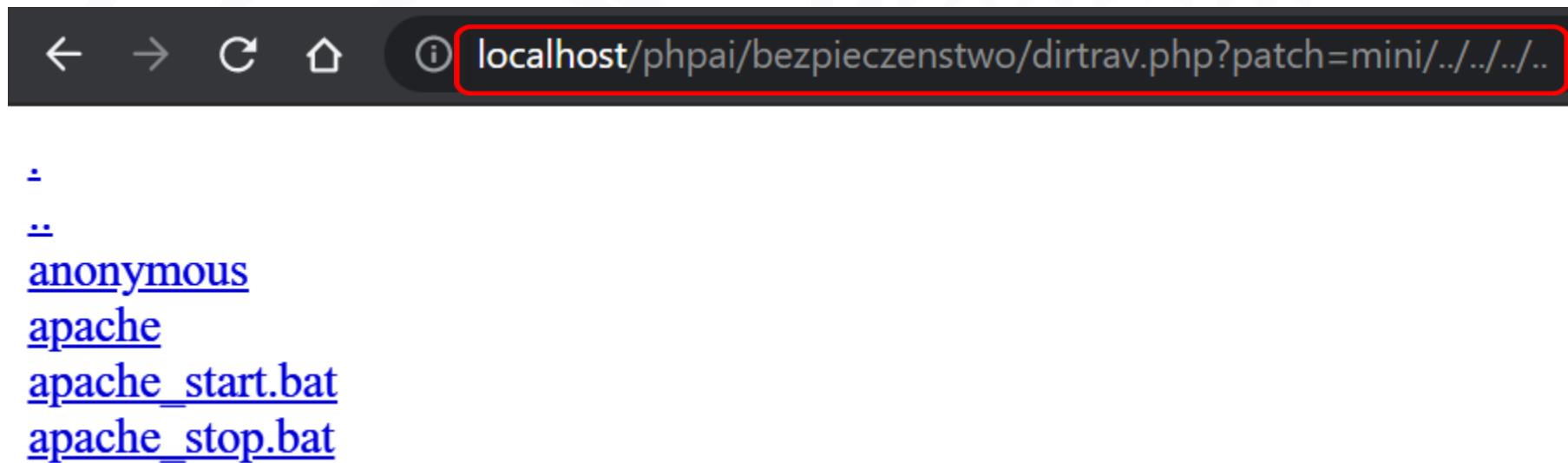


Directory Traversal - wynik



Directory Traversal - atak

`http://localhost/phpai/bezpieczenstwo/dirtrav.php?patch=mini/../../..`



`..` oznacza przejdź do katalogu nadzielnego, czyli można wyjść z narzuconego przez programistę katalogu i przejrzeć pliki z serwera

Atak typu File Inclusion

- Skrypt **wyklady.php** :

```
<?php  
echo "<h2> Wykłady z PHP</h2>";  
if (isset($_GET["nr"]))  
    include ($_GET["nr"].".txt");  
else  
    include("spistresci.txt");  
?>
```

- Plik **spistresci.txt**:

```
<ol>  
    <li><a href="wyklady.php?nr=1">Wstęp</a></li>  
    <li><a href="wyklady.php?nr=2">Instrukcje</a></li>  
</ol>
```

Wykłady z PHP

1. [Wstęp](#)
2. [Instrukcje](#)

Atak File Inclusion

- Wejście na stronę **http://localhost/wyklady.php?nr=1** pokazuje plik **1.txt** postaci:
`<h3>Wstęp</h3>`
`<p> Informacje wstępne ... </p>`
- Jeśli istnieje plik np. **trojan.txt** postaci:
<?php phpinfo(); ?>
to wejście na stronę:
http://localhost/wyklady.php?
nr=http://localhost/trojan
powoduje załadowanie i wykonanie (!)
dowolnego kodu (rozszerzenie txt - brak
parsowania PHP)

Wykłady z PHP

PHP Version 8.0.2

System

Build Date

Build System

Compiler

Zabezpieczenie przed atakiem File Inclusion

- Sprawdzenie poprawności przekazywanych danych lub stosowanie filtra **FILTER_VALIDATE_INT**):

```
<?php  
echo "<h2> Wykłady z PHP</h2>";  
if ($wyklad = (int)($_GET['nr']))  
    include ($wyklad.".txt");  
else  
    include("spistresci.txt");  
?>
```

Atak SQL Injection

Skrypt logowanie.php (1)

```
<?php
if (isset($_POST['login']) && isset($_POST['haslo']))
{ $login=$_POST['login']; $haslo=$_POST['haslo'];
$conn = new mysqli('localhost', 'root', '', 'sqlinj');
if ($conn->connect_errno) {
    printf("Nie udało się połączenie z serwerem: %s\n",
           $conn->connect_error); exit(); }
$query="SELECT poziom FROM users WHERE login='$login' AND
          haslo='$haslo'";
if ( $wynik = $conn->query($query) ){
    if($row = $wynik->fetch_array())
    { switch ($row[0])
        { case 0: print "Jesteś zwykłym użytkownikiem"; break;
          case 1: print "Jesteś administratorem!";
        }
    }
else print "Błąd logowania - spróbuj ponownie!";
$conn->close();
}}
```

login	haslo	poziom
admin	admin123	1
beata	beata123	0

logowanie.php (2) i atak

```
else { //drukuj formularz:  
    echo "<form action='logowanie.php' method='post'><p> ";  
    echo "Login: <input name='login' /><br/>";  
    print "Hasło: <input name='haslo' /><br/>";  
    print "<input type='submit' value='Zaloguj' /></form></p>";  
} ?>
```

Login:

Hasło:

Jesteś zwykłym użytkownikiem

Login:

Hasło:

Jesteś administratorem!

Atak SQL Injection i zabezpieczenie

- Atak możliwy, gdy w ***php.ini*** wyłączona opcja: **magic_quotes_gpc = Off**
SELECT poziom FROM users WHERE **login='admin' #' AND haslo='złe hasło'**
Znak **#** baza danych potraktowała jako znak sterujący (komentarz)
- **Zabezpieczenie:**
`$login=addslashes($_POST['login']);`
//lub lepiej z filtrem:
`$login = filter_input(INPUT_POST, "login", FILTER_SANITIZE_ADD_SLASHES);`
//lub przy pracy z mysqli:
`$mysqli = new mysqli("localhost", "user", "password", "users");`
`$login = $mysqli->real_escape_string($login);`
- Zapytanie po zabezpieczeniach będzie postaci:
SELECT poziom FROM users WHERE **login='admin\' #' AND haslo='złe hasło'**
Znak **\ przed apostrofem** baza danych traktuje jako element danych do zapytania a nie jako znak sterujący

Możliwe ataki SQL injection

- Niekontrolowany dostęp do danych np.:

... UNION SELECT ...

Dołączenie do zapytania operatora UNION umożliwia odczytanie z bazy danych zupełnie innych tabel niż przewidywane przez programistę

- Zmiana sposobu działania aplikacji:

... OR 1=1

Dołączenie do zapytania warunku logicznego zawsze spełnionego i powiązanie go spójnikiem OR umożliwia uzyskanie pełnego dostępu do danych, które programista usiłował filtrować

- W wielu interpreterach SQL jest możliwe wykonanie kilku działań w jednym wierszu np.:

... ; DELETE * FROM ...

W ten sposób można wykonać polecenie DELETE tam, gdzie zamiarem programisty było np. polecenie INSERT

Polecenia systemowe

- Jeśli wykonywane są polecenia systemowe serwera na parametrach pobranych z formularza np.:
`<?php system("w \$parametr"); ?>`
//w - pokaż kto jest zalogowany i co robi
a użytkownik poda ciąg postaci:
`;ps -aux` //pokaż wszystkie procesy
lub
`;cat /etc/passwd` //plik z hasłami
to zapytanie takie **wyświetli procesy w systemie lub plik z hasłami**
- Zabezpieczenie - analogiczne jak w przypadku bazy danych tylko z wykorzystaniem funkcji **escapeshellcmd**

Dołączanie plików

- Jeśli włamywaczowi uda się zmienić wartość \$PATH - może do naszego skryptu dołączyć swój dowolny plik PHP (który może wyświetlić wszystkie zmienne globalne i wykonać inne niebezpieczne funkcje)
- Zabezpieczanie z poziomu serwera:
 - ustawienie opcji **safe_mode** jako **on** uniemożliwia wykonywanie poleceń systemowych z poziomu WWW i skryptów (domyślnie off)
 - **safe_mode_exec_dir** - ustala jakie katalogi mogą być dostępne z poziomu PHP
 - **safe_mode_allowed_env_vars** - określa jakie zmienne środowiskowe mogą być zmieniane przez skrypt PHP

Sesja HTTP i atak session hijack

- Emulacja sesji opiera się na założeniu, że serwer aplikacji generuje dla każdego klienta HTTP niepowtarzalny identyfikator sesji, zwykle zapisywany w zmiennej Cookie, dzięki czemu id sesji powraca do serwera aplikacji przy każdym kolejnym żądaniu HTTP zgłoszonym przez tego samego klienta HTTP
- Po stronie serwera aplikacji znajduje się tzw. tablica sesji, w której z każdą wartością identyfikatora sesji związany jest zbiór programowych obiektów reprezentujących stan sesji
- Serwer aplikacji rozpoznaje użytkownika końcowego i udostępnia mu stan sesji **wyłącznie** w oparciu o wartość identyfikatora sesji
- Serwery aplikacji dbają o to, aby nie generować identycznych identyfikatorów sesji dla różnych użytkowników, lecz może się zdarzyć, że ktoś celowo zmodyfikuje wartość własnej zmiennej Cookie na wartość identyfikatora sesji innego użytkownika (atak session hijack)

Ataki związane z sesjami

- Przechwycenie sesji (ang. **Session Hijacking**) - odnosi się do próby uzyskania dostępu do istniejącej sesji użytkownika, tj. gdy identyfikator został przydzielony już wcześniej może nastąpić próba jego przechwycenia
- Wymuszenie sesji (ang. **Session Fixation**) - użytkownik zostaje nieświadomie skłoniony do użycia sesji zainicjalizowanej przez atakującego, np. poprzez odpowiednio spreparowany odnośnik z dołączonym identyfikatorem sesji
- Jeśli id sesji jest dołączany do adresu URL to każda osoba używająca tego samego komputera może poznać identyfikator sesji z historii stron przeglądarki. Ponadto adresy URL często zapisywane są przez serwery proxy, dzienniki zdarzeń itp., a w wyniku ich analizy można przeczytać id sesji
- Użytkownik, przekazując adres URL innym osobom, nieumyślnie udostępnia także identyfikator sesji - naturalnym wyborem do przekazywania identyfikatora sesji są ciasteczka

Dyrektywy konfiguracyjne dla id sesji w ciasteczku w PHP

- **session.use_cookies** - ustawia przekazywanie identyfikatora w ciasteczkach
- **session.use_only_cookies** - ustawia pobieranie identyfikatora **tylko** z ciasteczek
- **session.trans_sid** - przekazywanie identyfikatora poprzez adres URL i formularze WWW należy wyłączyć
- Parametry dotyczące samego ciasteczka sesji:
 - session.cookie_lifetime, session.cookie_domain, session.cookie_secure, session.cookie_httponly
 - session.cookie_path - domyślne ustawienie dla całego głównego katalogu (domyślny "/" zwiększa pole do manewru dla atakującego)

Przechowywanie danych sesyjnych

- PHP domyślnie przechowuje dane sesyjne w plikach na serwerze lokalnym w katalogu **/tmp**
- Na serwerach współdzielonych, prawa odczytu i zapisu do tego katalogu mają wszyscy użytkownicy i w praktyce mogą uzyskać dostęp do wszystkich identyfikatorów oraz danych sesji a atakujący może to wykorzystać do podmiany pliku sesji lub danych w nim zawartych
- **Zabezpieczenie:** dyrektywa konfiguracyjna **session.save_path**, która wskazuje nowy katalog do przechowywania plików sesji np.:
ini_set('session.save_path', '/bezpieczna/lokalizacja/sesji');

Kontrola czasu trwania sesji

- Sesje powinny wygasnąć w odpowiednio krótkim czasie aby zminimalizować ryzyko ataku
- **Sesja powinna być przerwana, gdy:**
 - nie stwierdzono żadnej aktywności przez pewien okres czasu, (np. 30 min.)
 - nastąpi błąd bezpieczeństwa
 - użytkownik przerwie sesję, poprzez opcję wylogowania
- Mechanizm powinien automatycznie usuwać z magazynu danych te identyfikatory i dane sesyjne, których czas ważności został przekroczony
- Ciasteczka również powinny mieć ograniczony czas istnienia
- **Przeciwdziałanie wymuszeniu sesji:**
 - przy każdej zmianie poziomu uprawnień użytkownika należy wygenerować **nowy id sesji** za pomocą funkcji `session_regenerate_id(true);`
 - niezbędne minimum - zmiana następuje po przejściu procesu uwierzytelniania

Kontrola czasu trwania sesji

<http://www.php.pl/Wortal/Artykuly/Bezpieczenstwo/Sesja-uzytkownika-w-PHP-zagrozenia-i-ochrona/Session-Hijacking>

```
session_start();
$now = time(); //czas rozpoczęcia sesji
$expiryTime = 1800; //czas ważności sesji w sekundach
if ( !isset($_SESSION['last_trace']) ) //sesja wygasła
{ $_SESSION['last_trace'] = $now; }
elseif ((int)$_SESSION['last_trace'] + $expiryTime < $now)
{ $sessionId = session_name();
$_SESSION = [];
if (isset($_COOKIE[$sessionId]))
{ setcookie($sessionId, '', $now-3600, '/'); }
session_destroy();
echo 'sesja wygasła!';
}
```

Regeneracja id sesji - skrypt php

```
session_start();
$now = time();
$regenerateSec = 1800; //czas do regeneracji sesji
$regenerateReq = 10; //liczba żądań http
if (!isset($_SESSION['started']))
{ $_SESSION['started'] = true; $_SESSION['time'] = $now;
  $_SESSION['req'] = 1; }
else $_SESSION['req']++;
//regeneracja sesji po określonym czasie:
if (isset($_SESSION['time']) && ((int)$_SESSION['time'] +
$regenerateSec < $now))
{ session_regenerate_id(true); $_SESSION['time'] = $now; }
//regeneracja sesji po określonej liczbie żądań:
if (isset($_SESSION['req']) && ((int)$_SESSION['req'] 
>$regenerateReq))
{ session_regenerate_id(true); $_SESSION['req'] = 1; }
```

Zapobieganie adopcji identyfikatora

- Uniemożliwienie adopcji sesji (ang. Session Adoption), polega na ignorowaniu identyfikatorów pochodzących od użytkownika, które nie znajdują się w magazynie danych - akceptowalne są tylko te, które system sam utworzył
- Np.:

```
session_start();
if (!isset($_SESSION['our_own']))
{
    session_regenerate_id();
    $_SESSION['our_own'] = true;
}
```

Atak Cross-Site Request Forgery

- Atak CSRF polega na tym, że atakujący umieszcza adres do pewnego skryptu jako np. adres obrazka
- Skrypt ten najczęściej jest zabezpieczony przed wykonaniem dla użytkowników nieupoważnionych, lecz jeżeli obrazek zostanie otwarty przez zalogowanego użytkownika, mającego stosowne uprawnienia w systemie (np. administratora), to może nieświadomie wykonać niebezpieczną akcję (np. skasowania zasobu)
- Szczególnie niebezpieczny, gdy parametry są przekazywane metodą GET
- **Zabezpieczenie:**
 - wysyłanie kluczowych elementów metodą POST (nie zawsze skuteczne, ponieważ żądania metodą POST można wysyłać przy użyciu JavaScript)
 - nie odwiedzanie innych stron, przy zalogowaniu w innym serwisie
 - we własnym serwisie należy uniemożliwić umieszczanie kodu JavaScript

Atak Denial of Service

- Ataki typu Denial of Service (DoS, DDoS) polegają na generowaniu dużej liczby sztucznych żądań HTTP, które powodują przeciążenie atakowanego systemu. Ich skutkiem może być istotne pogorszenie jakości obsługi użytkowników lub całkowita zapaść systemu (wyczerpanie się zasobów pamięci operacyjnej)
- Wczesne rozpoznawanie tego typu ataków jest trudne, gdyż sztucznie generowane żądania nie różnią się od żądań przesyłanych przez rzeczywistych użytkowników
- W zasadzie nie istnieje żadna niezawodna metoda ochrony przed takimi atakami

"Nie ma skryptów bez luk, są tylko takie, w których tych luk jeszcze nie znaleziono"



PROGRAMOWANIE APLIKACJI INTERNETOWYCH

Wykład 8

Model modułowej aplikacji internetowej na przykładzie strony fikcyjnej firmy

Beata Pańczyk



**Fundusze
Europejskie**
Wiedza Edukacja Rozwój



**Rzeczpospolita
Polska**

Unia Europejska
Europejski Fundusz Społeczny



Wprowadzenie

- Współczesne aplikacje internetowe - kilkadziesiąt stron, przeważnie generowanych dynamicznie i posiadających pewne cechy wspólne np.:
 - nagłówek z nazwą serwisu
 - stopkę z adresami webmasterów, zastrzeżeniami praw
 - panel nawigacyjny z odnośnikami do podstron serwisu
- Cały serwis - podzielony na moduły
- Szkielet strony:
 - zawiera poszczególne elementy **powtarzające się** oraz skrypty generujące treść dla każdej z podstron
 - jeżeli podstrona nie zawiera treści generowanych dynamicznie to treść może być pobierana bezpośrednio z pliku HTML

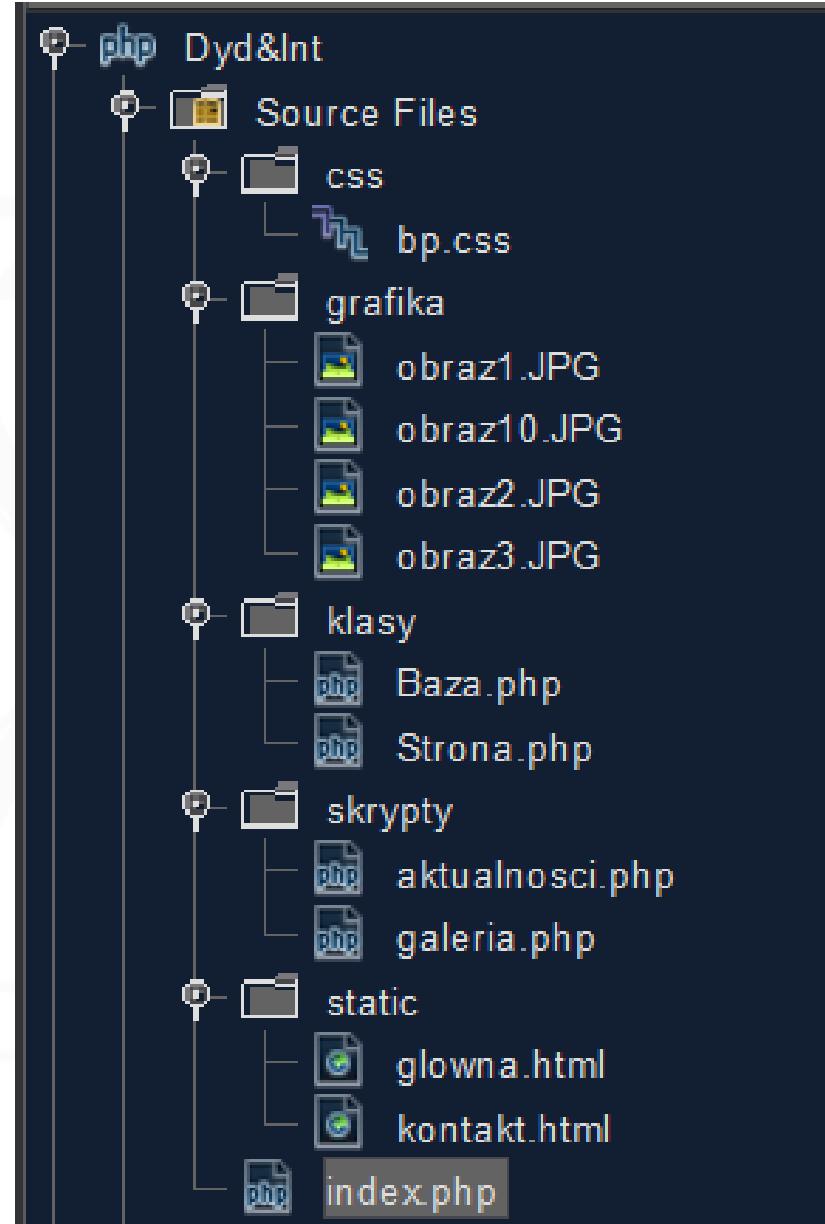
Model modułowej aplikacji WWW

- Projekt serwisu WWW - jednolity i spójny wygląd, łatwa nawigacja i orientacja:
 - w tym samym miejscu umieszczone menu serwisu, nazwa oraz treść strony
 - treść strony i tytuł wyświetlany na belce okna przeglądarki ulega zmianie (elementy dynamiczne) - pozostałe elementy stałe (statyczne)
- Poszczególne elementy serwisu - w odpowiednich folderach, np.:
 - **Dyd&Int** - katalog główny aplikacji
 - **static** - pliki HTML zawierające statyczne elementy treści stron
 - **klasy** - pomocnicze pliki z definicjami klas
 - **skrypty** - skrypty PHP generujące treść dynamiczną poszczególnych podstron serwisu
 - **grafika** - elementy graficzne
 - **css** - reguły stylistyczne

Przykład - modułowy serwis firmy

- Strona startowa ***index.php*** (w folderze głównym projektu)
 - generuje stronę główną (w oparciu o klasę **Strona** (wykład 4) z linkami do podstron: ***glowna, kontakt, galeria, aktualnosci***)
- Podstrony:
 - z treścią statyczną:
 - ***glowna.html***
 - ***kontakt.html***
 - z treścią dynamiczną:
 - ***galeria.php*** - wyświetla obrazki znajdujące się w folderze **grafika**
 - ***aktualosci.php*** - pobiera i wyświetla informacje z bazy danych korzystając z definicji klasy **Baza** (wykład 5)

Struktura folderów i plików aplikacji



Szablon strony

- Szablon strony:
 - podstawowy element każdej strony tworzony jest w oparciu o obiekt klasy **Strona**
 - zawiera definicję ogólnego wyglądu strony i jest wykorzystywany przez każdy skrypt generujący poszczególne podstrony w aplikacji
- Modyfikacja treści szablonu strony:
 - **tytul_strony** - ustawiany metodą **ustaw_tytul()** klasy **Strona**
 - **przyciski nawigacji** - ustawiane metodą **ustaw_przyciski()** klasy **Strona**
 - główna zawartość strony - ustawiane metodą **ustaw_zawartosc()** klasy **Strona** (w przypadku galerii i aktualności - zawartość jest generowana dynamicznie, w pozostałych przypadkach zawartość statyczna pobierana jest z odpowiednich plików HTML)

Dołączanie elementów statycznych

- Statyczna zawartość strony głównej i kontakt wczytywana jest z odpowiedniego pliku wskazanego przez przekazywany metodą GET parametr **strona=glowna** lub **strona=kontakt** (domyślnie odczyt następuje z pliku **glowna.html**)
- Należy sprawdzić czy wartość parametru zawarta jest w odpowiednim zbiorze stron - jeśli nie wyświetlana jest strona z zawartością pobraną domyślnie z pliku **glowna.html**

Plik index.php (1)

```
<?php
    require_once("klasy/strona.php");
    $strona_glowna=new Strona();
    $strona_glowna->ustaw_style("css/bp.css");
$menu = ["Strona główna"=>"index.php?strona=glowna",
         "Aktualności"=>"skrypty/aktualnosci.php",
         "Galeria"=>"skrypty/galeria.php",
         "Kontakt"=>"index.php?strona=kontakt"];
$strona_glowna->ustaw_przyciski($menu);
if (filter_input(INPUT_GET,'strona')) //zażądano strony statycznej
{ $strona = filter_input(INPUT_GET, 'strona');
switch ($strona)
{   case 'kontakt': $strona = 'kontakt'; break;
    //inne strony z treścią statyczną
    default : $strona = 'glowna'; //domyślnie
}
} else $strona="glowna";
```

Plik index.php (2)

```
$plik="static/".$strona.".html"; //plik, z którego należy  
//pobrać zawartość  
  
$tresc = "";  
$pl=fopen($plik,"r");  
if ($pl)  
{ // Wczytanie treści strony statycznej z pliku  
    while (!feof($pl))  
    { $tresc.= fgets($pl); }  
    $tytul="Firma Dyd&Int - ".$strona;  
    $strona_glowna->ustaw_tytul($tytul);  
}  
fclose($pl);  
$strona_glowna->ustaw_zawartosc($tresc);  
$strona_glowna->wyswietl();  
?>
```

Zawartość statyczna

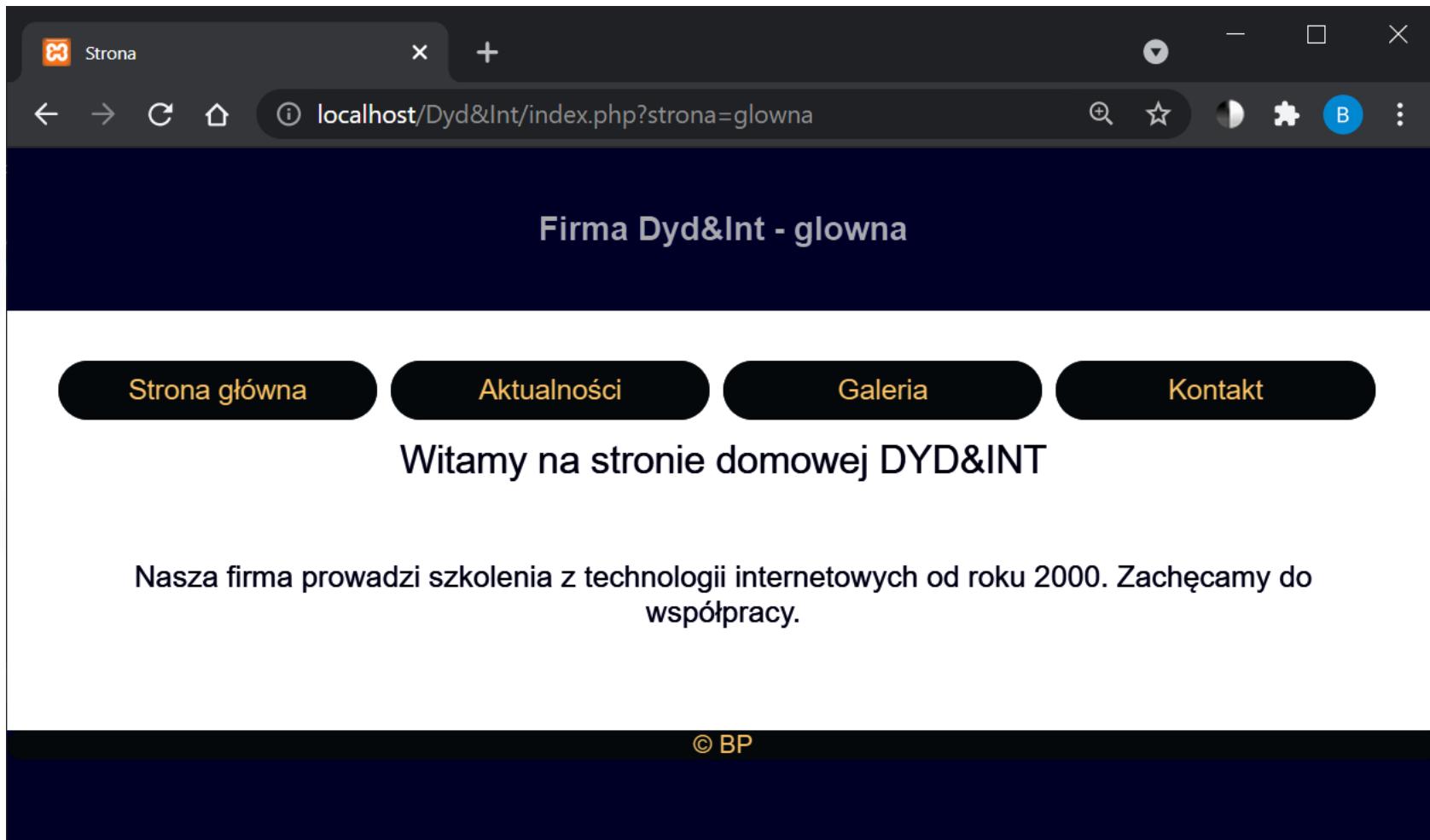
- **Plik `glowna.html`:**

```
<h2> Witamy na stronie domowej DYD&INT</h2>
<h3> Nasza firma prowadzi szkolenia z technologii
    internetowych od roku 2000. Zachęcamy do współpracy.
    </h3>
```

- **Plik `kontakt.html`:**

```
<h2>Informacje kontaktowe</h2>
<table>
<tr><td>Adres:</td>
      <td>ul. Miła 13 <br /> 99-999 XXX</td></tr>
<tr><td>Telefon:</td><td>999-999-999</td></tr>
<tr><td>e-mail:</td><td>dydint@xxx.pl</td></tr>
</table>
```

Widok strony głównej serwisu



Widok strony kontakty

The screenshot shows a web browser window with the following details:

- Title Bar:** Strona
- Address Bar:** localhost/Dyd&Int/index.php?strona=kontakt
- Content Area:**
 - Section Header:** Firma Dyd&Int - kontakt
 - Navigation Buttons:** Strona główna, Aktualności, Galeria, Kontakt
 - Section Title:** Informacje kontaktowe
 - Contact Information Table:**

Adres:	ul. Miła 13 99-999 XXX
Telefon:	999-999-999
e-mail:	dydint@xxx.pl
 - Page Footer:** © BP

Strona dynamiczna galeria.php

```
<?php
require_once("../klasy/strona.php");
$strona_glowna=new Strona();
$strona_glowna->ustaw_style("../css/bp.css");
$strona_glowna->ustaw_tytul("Dyd&Int - Galeria");
$menu=[ "Strona główna"=>"index.php?strona=glowna",
        "Aktualności"=>"skrypty/aktualnosci.php",
        "Galeria"=>"skrypty/galeria.php",
        "Kontakt"=>"index.php?strona=kontakt"];
$strona_glowna->ustaw_przyciski($menu);
$tresc=<h2>Nasza galeria</h2>;
$katalog="../grafika"; $kat = @opendir($katalog);
while ($plik = readdir($kat)) {
    if ($plik !=="." && $plik !=="..")
        $tresc.=<img src='../grafika/$plik' alt='$plik' /> ";
closedir($kat);
$strona_glowna->ustaw_zawartosc($tresc);
$strona_glowna->wyswietl();
```

Widok strony galerii

The screenshot shows a web browser window with the following details:

- Address Bar:** Strona | localhost/Dyd&Int/skrypty/galeria.php
- Page Title:** Dyd&Int - Galeria
- Navigation Buttons:** Strona główna, Aktualności, Galeria, Kontakt
- Section Header:** Nasza galeria
- Image Grid:** Four images are displayed in a row:
 - A coastal scene with rocks and turquoise water.
 - A yellow seaplane flying over a rocky cliffside.
 - A close-up of a green pine tree against a blue sky.
 - A wide shot of a large, rugged mountain range under a clear blue sky.
- Page Footer:** © BP

Strona dynamiczna aktualnosci.php

```
<?php
require_once("../klasy/strona.php");
require_once("../klasy/Baza.php");
$strona_glowna = new Strona();
$strona_glowna->ustaw_style("../css/bp.css");
$strona_glowna->ustaw_tytul("Dyd&Int - Aktualności");
$menu=[ "Strona główna"=>"index.php?strona=glowna",
        "Aktualności"=>"skrypty/aktualnosci.php",
        "Galeria"=>"skrypty/galeria.php",
        "Kontakt"=>"index.php?strona=kontakt"];
$strona_glowna->ustaw_przyciski($menu);
$tresc = "<h2>Informacje z bazy</h2>";
$ob = new Baza("localhost", "root", "", "dane");
$sql = "select * from news";
$pola = array("Nagłówek", "Treść");
//wyswietl dane z bazy:
$tresc .= $ob->select($sql, $pola);
$strona_glowna->ustaw_zawartosc($tresc);
$strona_glowna->wyswietl();
```

Widok strony aktualności

Dyd&Int - Aktualności

Strona główna Aktualności Galeria Kontakt

Informacje z bazy

Artykuł o PHP	Treść artykułu na temat podstaw PHP.
Artykuł o systemach operacyjnych	Trochę więcej rzeczy na temat systemów operacyjnych i procesach systemowych.
Szkielet programistyczny Laravel	Jak rozpocząć pracę z wykorzystaniem wzorca MVC na przykładzie framework'a Laravel ..



PROGRAMOWANIE APLIKACJI INTERNETOWYCH

Wykład 9, część 1

Projektowanie obiektowe. Wprowadzenie do wzorców projektowych. Elementy języka UML. Aplikacje wielowarstwowe i wzorzec projektowy MVC.

Beata Pańczyk



**Fundusze
Europejskie**
Wiedza Edukacja Rozwój



**Rzeczpospolita
Polska**

Unia Europejska
Europejski Fundusz Społeczny



Czym jest projektowanie?

- **Problemy** programistów aplikacji internetowych:
 - Jak obsługiwać napływające żądania?
 - Jak konwertować zawarte w nich dane do instrukcji wykonywalnych wewnątrz aplikacji?
 - Jak pozyskiwać dane ze źródeł danych?
 - Jak prezentować wyniki?
- **Projektowanie** kodu to definiowanie systemu, to określenie wymagań i zakresu jego zadań:
 - Co system powinien robić?
 - Czego potrzebuje do realizacji swoich zadań?
 - Jakie dane system generuje?
 - Czy spełniają one określone wcześniej wymagania?
- Na niższym poziomie projektowanie oznacza proces definiowania uczestników systemu i rozpoznawania zachodzących pomiędzy nimi relacji

Projektowanie obiektowe

- System obiektowy składa się z klas - trzeba zdecydować o naturze klas uczestniczących w systemie
- Klasy składają się częściowo z metod - definiując klasy trzeba zdecydować o grupowaniu metod
- Klasy często uczestniczą w relacjach dziedziczenia, które zapewniają im spełnienie wymogów wspólnych dla poszczególnych części systemu interfejsów
- Pierwsze wyzwanie w projektowaniu systemu - rozpoznanie i wytypowanie interfejsów
- Klasy mogą również wchodzić w inne relacje - mogą składać się z innych klas i typów (relacja kompozycji), albo utrzymywać listy egzemplarzy innych typów itp.

Czym są wzorce projektowe?

- "Wzorzec to rozwiązanie problemu w danym kontekście" - The Gang of Four (Erich Gamma, Richard Helm, Ralph Johnson and John Vlissides), Design Patterns: Elements of Reusable Object-Oriented Software
- Polskie wydanie: Wzorce projektowe. Elementy oprogramowania obiektowego wielokrotnego użytku, Helion 2010
- Wzorce są pomocne w rozwiązywaniu najczęstszych problemów

Po co nam wzorce projektowe?

- Wzorzec projektowy:
 - definiuje problem
 - definiuje rozwiązanie
 - jest niezależny od języka programowania - definiuje obiekty i rozwiązania obiektowe
 - definiuje słownictwo
 - jest wypróbowany
- Wzorce mają współpracować - zastosowanie jednego wzorca powinno tworzyć warunki do zastosowania innego
- Wzorce promują prawidła projektowania obiektowego

Aplikacje i warstwy

- Wiele wzorców projektowych promuje podział aplikacji na szereg jak najmniej zależnych od siebie warstw
- Warstwy w systemie korporacyjnym pełnią rolę specjalizującą, podobnie do klas, tyle, że na większą skalę
- Typowy podział warstwowy systemu to:
 - **warstwa widoku**
 - **warstwa poleceń i kontroli**
 - **warstwa logiki biznesowej**
 - **warstwa danych**
- W projektach warstwy te mogą być ze sobą łączone i mogą być wdrażane różne sposoby komunikacji pomiędzy nimi

Warstwy typowego systemu korporacyjnego

Generuje żądanie kierowane do warstwy poleceń i kontroli

Interpretuje żądanie i odpytuje warstwę logiki

Przetwarza zadanie biznesowe

Obsługuje pozyskiwanie i utrwalanie danych



Podstawy języka UML

- UML (ang. Unified Modeling Language) to ujednolicony język modelowania
- UML to **graficzny opis systemów obiektowych**
- Graficzne diagramy klas wykorzystywane są do opisu struktur danych i podziału zadań
- Diagramy klas są podstawą opisu relacji zachodzących w systemach obiektowych, ale to tylko jeden z wielu elementów języka UML

Reprezentowanie klas

- Klasa jest reprezentowana na diagramie prostokątem postaci:



- Klasy abstrakcyjne są wyróżniane albo pochyleniem czcionki nazwy albo umieszczonym poniżej nazwy słowem abstrakcyjna:



- Interfejsy reprezentowane są podobnie jak klasy, ale uzupełnione słowem <<interfejs>>



Atrybuty

- Atrybuty odwzorowują składowe klasy:

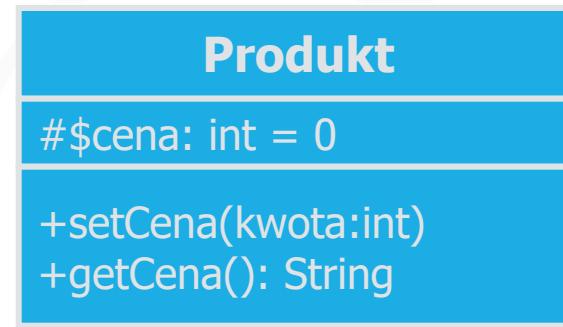


- Symbole widoczności atrybutów:

Symbol	Widoczność	Znaczenie
+	Publiczna	Atrybut dostępny ogólnie
-	Prywatna	Atrybut dostępny tylko w ramach bieżącej klasy
#	Chroniona	Atrybut dostępny w ramach bieżącej klasy i jej pochodnych

Operacje

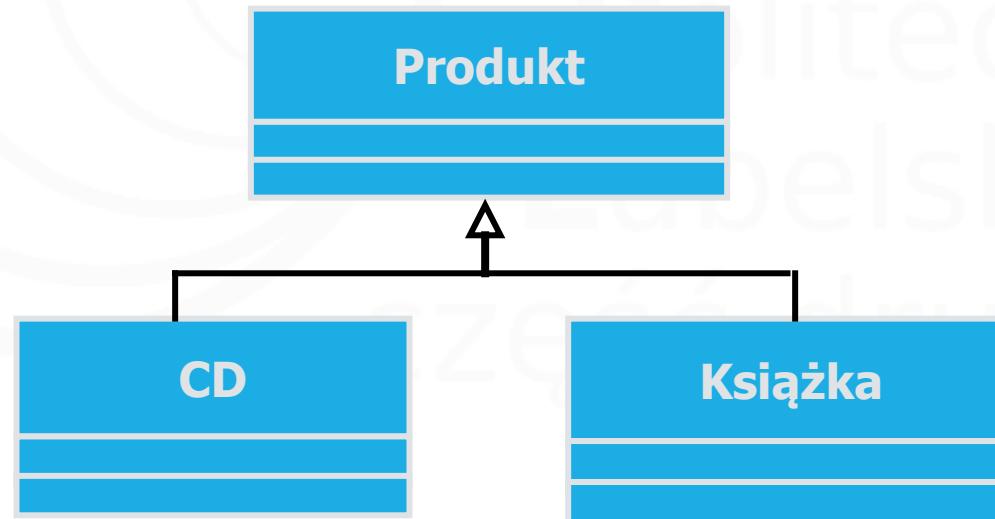
- Operacje reprezentują metody:



- Każdy parametr metody składa się z nazwy, dwukropka i typu
- Składnia ta jest traktowana elastycznie - można na przykład pominąć typ zwracanego wyniku czy nazwy argumentów metody

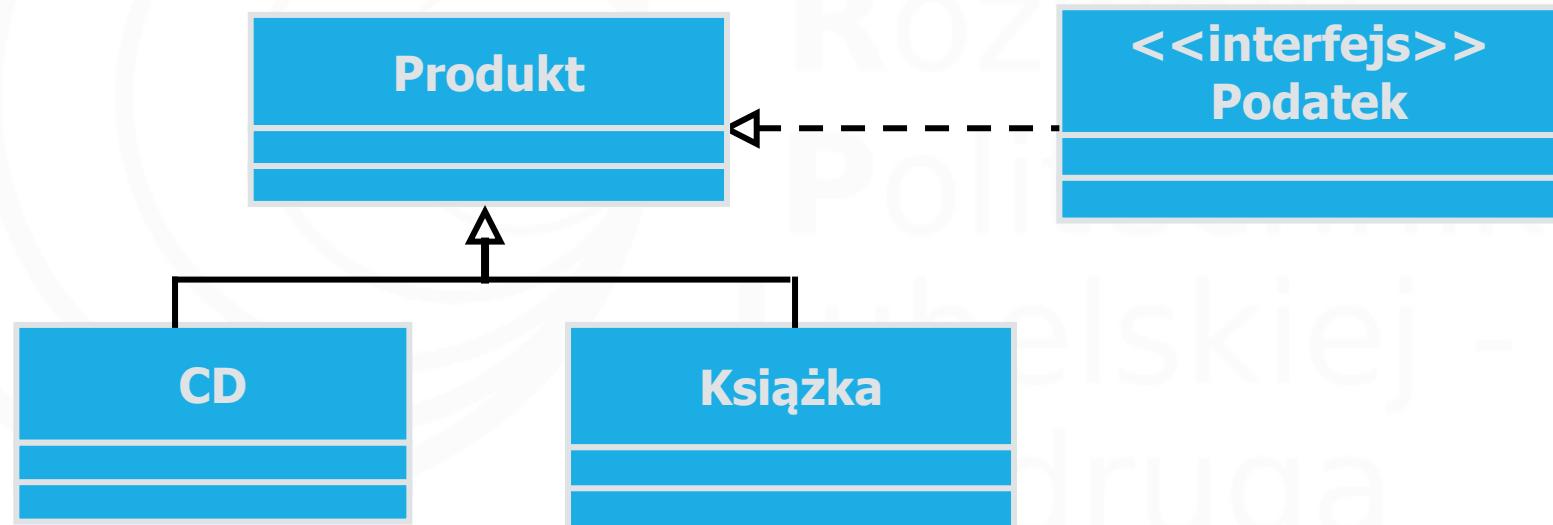
Relacja dziedziczenia

- UML opisuje relację dziedziczenia jako relację uogólnienia, czyli ‘generalizacji’ klas pochodnych w klasie bazowej
- Relacja ta jest reprezentowana na diagramie przez linię ciągłą wiodącą od klasy pochodnej do klasy bazowej zakończoną niewypełnioną strzałką:



Relacja implementacji (realizacji)

- Relacja pomiędzy interfejsem a klasą implementującą interfejs to w UML relacja **realizacji**:



Relacja powiązania (asocjacji)

- Relacja **powiązania** zachodzi gdy składowa klasy przechowuje referencję egzemplarza (albo egzemplarzy) innej klasy, np.: relacje powiązania pomiędzy klasami **Nauczyciel** i **Uczeń**:

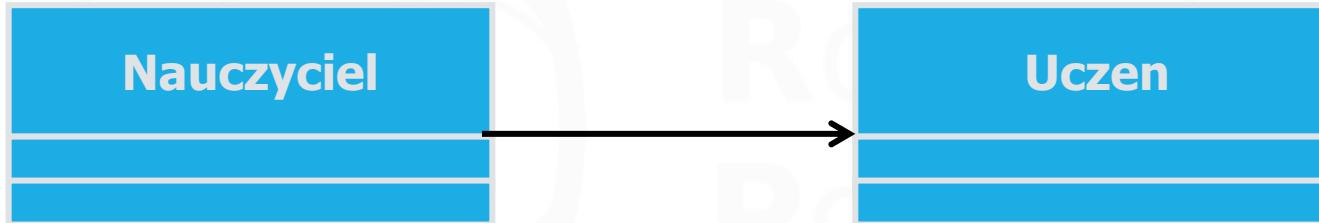


Opis ten nie informuje jednoznacznie o rodzaju powiązania – wiadomo jedynie, że obiekt klasy **Nauczyciel** będzie przechowywał referencję do jednego lub wielu obiektów klasy **Uczeń** lub odwrotnie

- Relacja powiązania może być również dwustronna

Relacja powiązania (asocjacji)

- Do określenia kierunku relacji powiązania służą strzałki, np. jeśli obiekt klasy **Nauczyciel** ma przechowywać referencję do obiektu klasy **Uczeń** (ale nie odwrotnie) - strzałka prowadzi od klasy **Nauczyciel** do **Uczeń**:

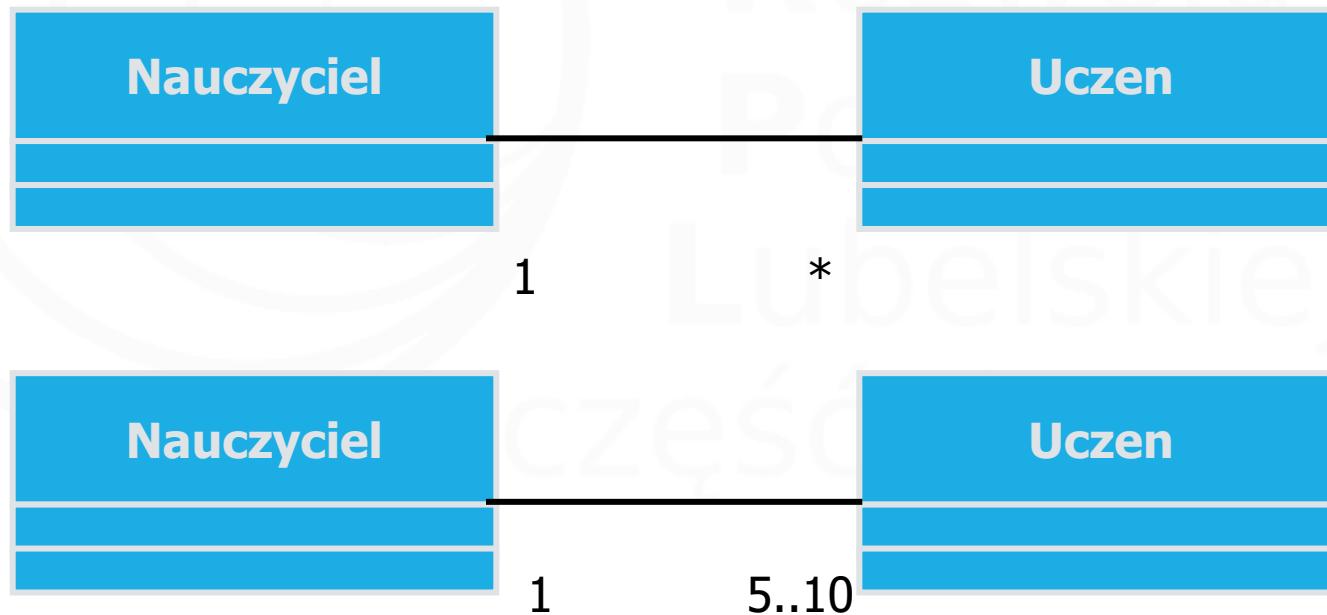


- Gdyby obiekty obu klas przechowywały referencję do obiektów drugiej klasy – to taką dwukierunkową relację sygnalizuje się strzałkami w obu kierunkach:



Krótkości relacji powiązania

- W relacji powiązania można wyszczególnić liczbę egzemplarzy klasy, do której odwołuje się każdy obiekt - na diagramie oznacza się to za pomocą liczb albo zakresów:



Agregacja i kompozycja

- Agregacja i kompozycja to relacje o charakterze zbliżonym do powiązania - wszystkie one opisują sytuację, w której klasa przechowuje trwałą referencję do jednego lub wielu obiektów innej klasy
- Przy agregacji i kompozycji obiekty te jednak wchodzą w skład obiektu bieżącej klasy:
 - przy agregacji - obiekt zawierający się w obiekcie danej klasy jest jego nieodłączną częścią, chociaż może być równocześnie zawierany w innych obiektach (na diagramie relacja taka jest obrazowana symbolem **puстego rombu**)
 - przy kompozycji - do obiektu zawieranego może odwoływać się wyłącznie obiekt go zawierający (na diagramie relacja taka jest obrazowana symbolem **wypełnionego rombu**)

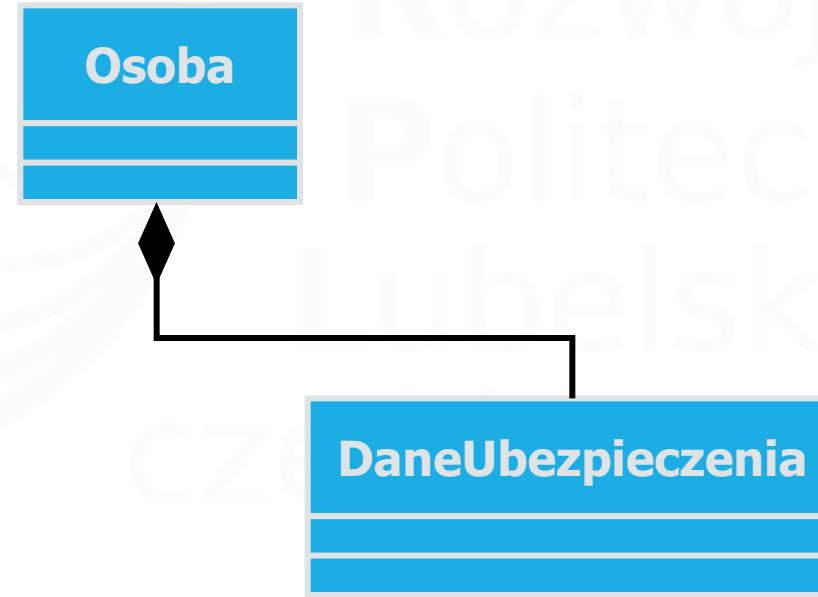
Agregacja

- Uczniowie tworzą grupę zajęciową, równocześnie poszczególni uczniowie mogą należeć do więcej niż jednej grupy. Odwołanie zajęć grupy nie oznacza zwolnienia uczniów do domu ponieważ mogą mieć jeszcze inne zajęcia w innej grupie



Kompozycja

- Klasa Osoba zawiera referencję do obiektu DaneUbezpieczenia - nie może być osoby o więcej niż jednym numerze ubezpieczenia

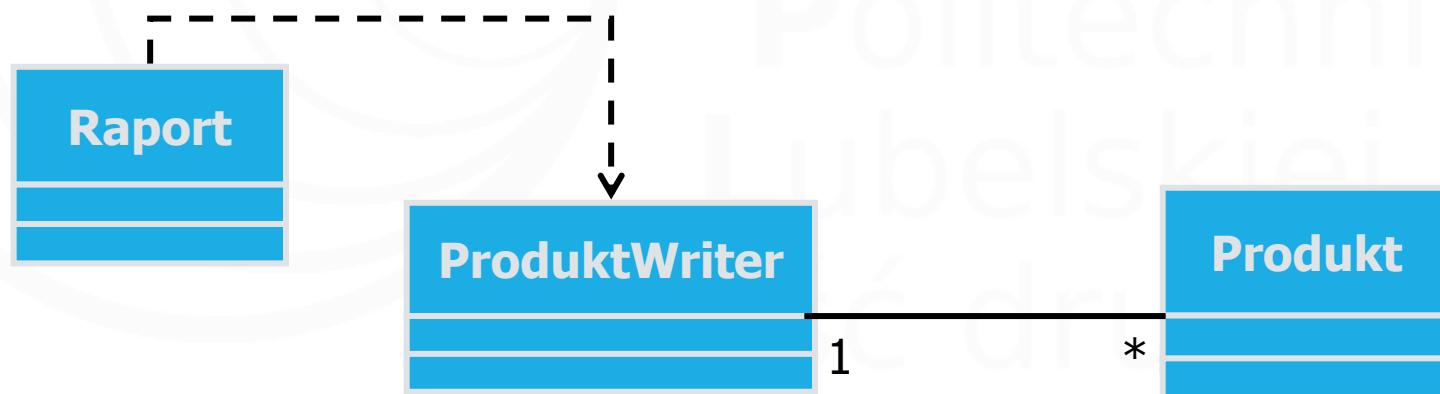


Relacja użycia

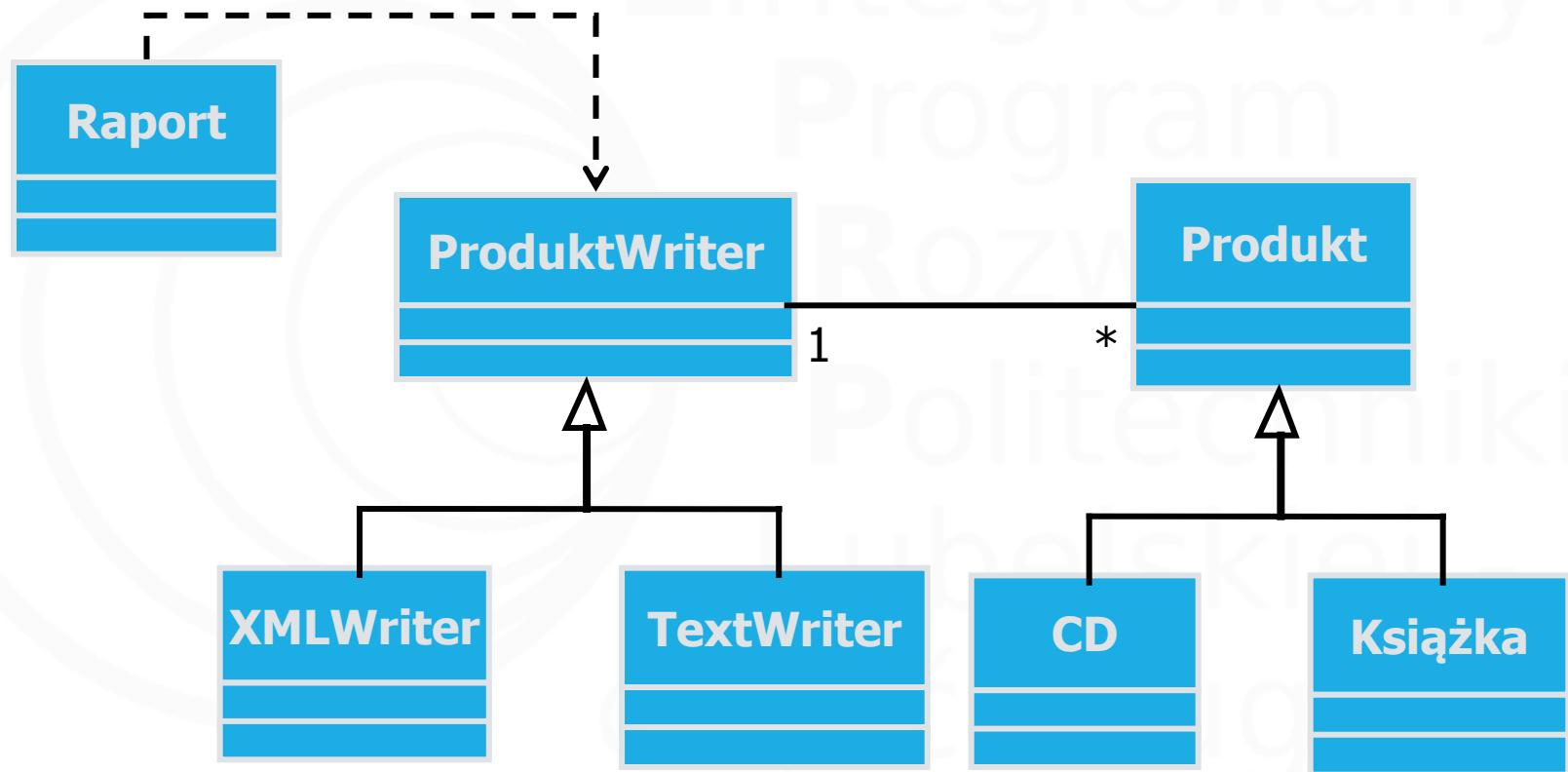
- W języku UML jest opisywana jako ‘zależność’
- Najsłabsza z relacji - nie opisuje stałego, a jedynie przejściowe powiązanie pomiędzy klasami
- Obiekt klasy używanej może zostać przekazany do obiektu klasy używającej za pośrednictwem argumentu wywołania metody, może też zostać pozyskany jako wartość zwracana z wywołania metody
- Relacja użycia jest reprezentowana przerywaną linią i otwartą strzałką łączącą dwie klasy

Relacja użycia

- Klasa **Raport** używa obiektu klasy **ProduktWriter** - nie oznacza to jednak, że klasa **Raport** przechowuje referencję do obiektu **ProduktWriter**
- Z kolei obiekt klasy **ProduktWriter** przechowuje trwale tablicę obiektów klasy **Produkt**



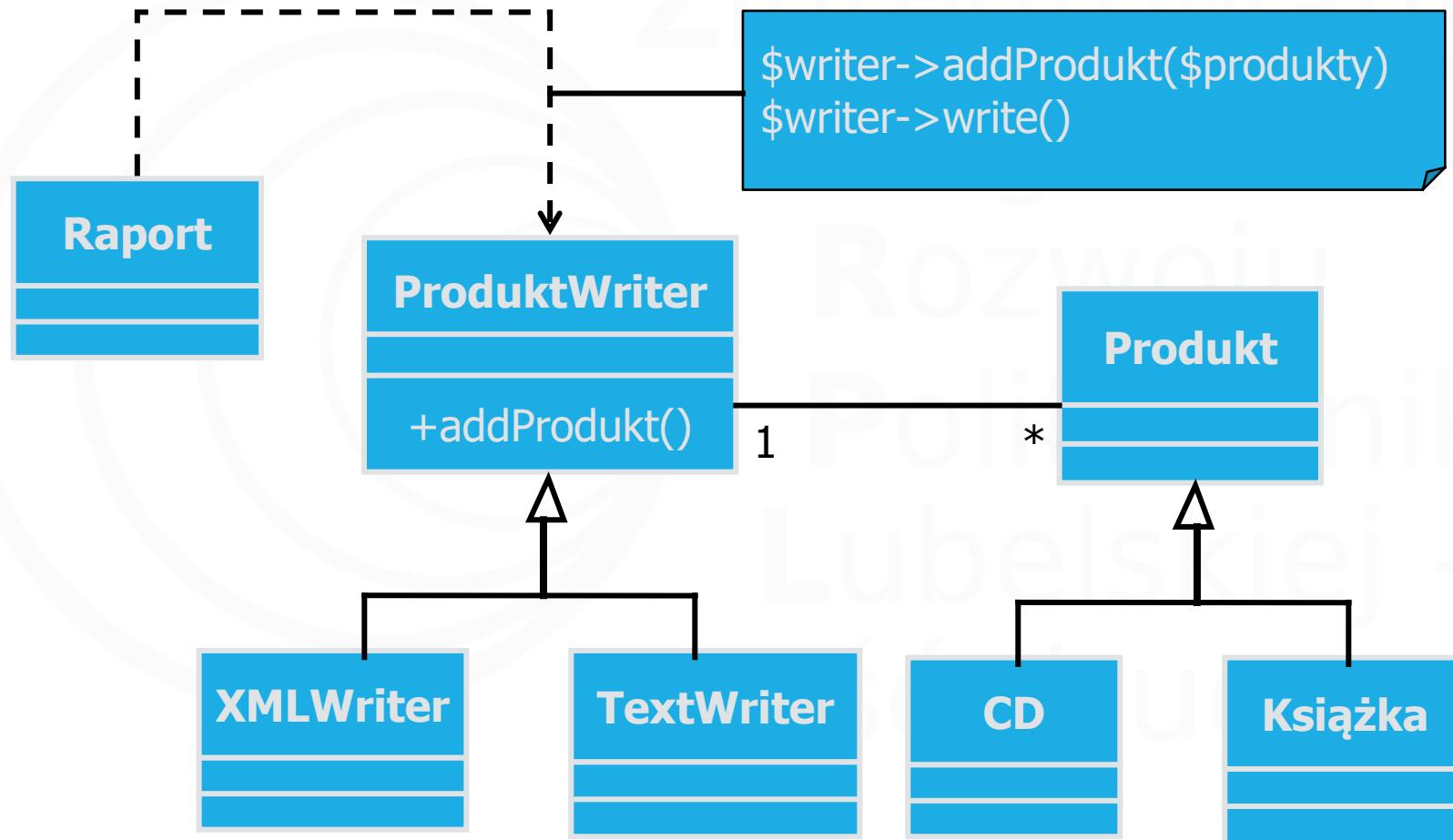
Przykładowy diagram klas



Notki

- Diagramy klas dobrze odzwierciedlają strukturę systemu obiektowego - nie dają jednak poglądu na proces odbywający się w systemie
- Przykładowy diagram (poprzedni slajd) ilustrował klasy uczestniczące w systemie:
 - klasa **Raport** używa obiektu klasy **ProduktWriter**, nie wiadomo jednak na czym to użycie polega
 - Sens tego użycia można przybliżyć umieszczając na nim notki, które często zawierają fragmenty pseudokodu
 - Dzięki notce (następny slajd) widać, że obiekt klasy **Raport** używa obiektu klasy **ProduktWriter** do wyprowadzenia danych o produkcie

Przykładowy diagram klas z notką



Wzorzec MVC

- MVC (ang. Model-View-Controller) to architektoniczny wzorzec projektowy w informatyce do organizowania struktury aplikacji posiadających graficzne interfejsy użytkownika
- Zwykle traktowany jako pojedynczy wzorzec, lecz może on być także uważany za złożony wzorzec wykorzystujący idee wzorców prostych takich, jak Obserwator, Strategia czy Kompozyt - oba podejścia nie wykluczają się
- MVC to sposób na oddzielenie poszczególnych części aplikacji, definiuje **trzy podstawowe** elementy, które są od siebie niezależne i pozwalają na dużą elastyczność podczas budowania, użytkowania i rozszerzania aplikacji:
 - **logikę biznesową** reprezentuje **Model**
 - **logiki prezentacyjną** reprezentuje widok **View**
 - **sterowanie** całą aplikacją zapewnia **Controller**

Elementy MVC - Model

- Odpowiedzialny za tzw. logikę biznesową (m.in. za przechowywanie danych)
- Udostępnia spójny interfejs i ukrywa właściwą implementację - obiekt używający modelu nie musi wiedzieć, czy dane przechowywane są w bazie danych czy w plikach
- Dla obiektu ważne jest, aby model miał API, które on zna
- Mając dwie klasy modelu, jedną operującą na bazie danych i drugą pracującą z plikami - obie mając ten sam interfejs, który można wymieniać bez obawy, że program przestanie działać

Elementy MVC - Widok

- Odpowiedzialny za wyświetlenie danych
- Wymieniając widok można zmieniać sposób wyświetlania danych, np. pomiędzy formatem HTML, XML, PDF itp.
- Widok powinien pobierać dane z modelu i formatować je do pożądanego formatu
- Widok nie wie w jaki sposób przechowywane są dane
- Widok nie może w żaden sposób zmieniać danych - może je tylko wyświetlać

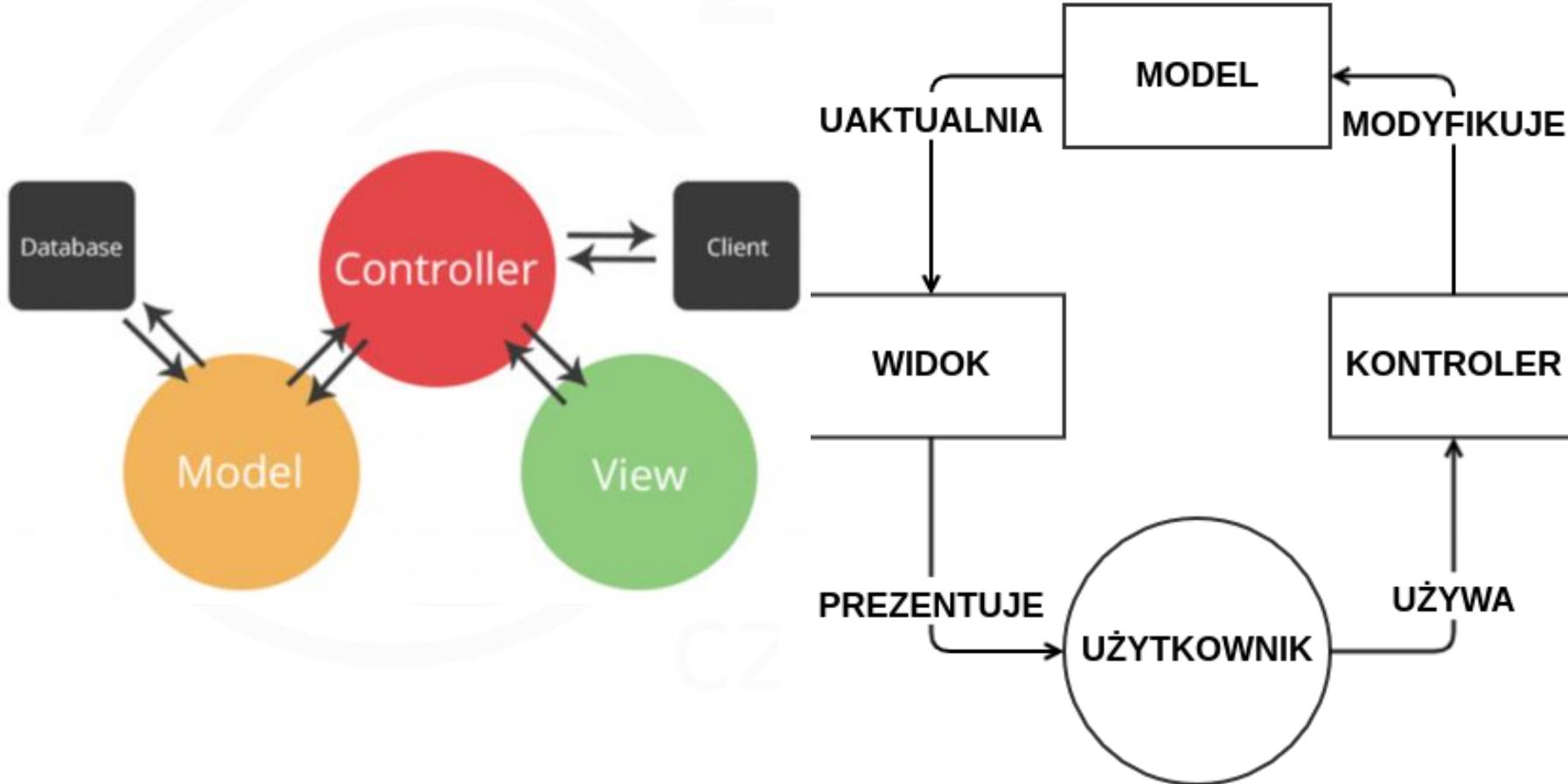
Elementy MVC - Kontroler

- Serce całej aplikacji - punkt wejścia
- Do niego trafiają żądania HTTP (wraz z parametrami przesyłanymi metodą GET/POST), z których kontroler pobiera akcję jaką chce wykonać użytkownik
- Odpowiedzialny za przetwarzanie zadań i wywoływanie właściwej akcji lub widoku
- Jeżeli aplikacja ma wyświetlić pewne dane, wywoływany jest widok a jeżeli dane trzeba zmienić to do sterownika (kontrolera) należy wprowadzić odpowiednie metody (akcje)
- Akcja - pojedyncza czynność wykonywana przez aplikację, najczęściej akcja jest metodą, która wywołana przez kontroler zwraca nazwę widoku do pokazania użytkownikowi

Schemat zależności w MVC

<https://pl.wikipedia.org/wiki/Model-View-Controller#/media/Plik:Mvc-diagram.png>

<https://geek.justjoin.it/mvc-vs-mvvm-dlaczego-młodszy-brat-zyskuje-na-popularności>



Zastosowanie MVC

- Wzorca MVC należy używać jeżeli:
 - aplikacja jest duża i skomplikowana
 - aplikacja ma być elastyczna i ma mieć możliwość rozbudowy
 - pracuje nad nią wielu programistów (można wówczas w łatwy sposób podzielić odpowiedzialność za poszczególne elementy systemu)



PROGRAMOWANIE APLIKACJI INTERNETOWYCH

Wykład 9, część 2

Modele, widoki, kontrolery i ich implementacja w PHP

Beata Pańczyk



**Fundusze
Europejskie**
Wiedza Edukacja Rozwój



**Rzeczpospolita
Polska**

Unia Europejska
Europejski Fundusz Społeczny



Model prostej aplikacji internetowej

- Dla prostych aplikacji, bez kontrolera, skrypt wejściowy *index.php* najczęściej jest postaci (wykład 8):

```
<?php  
    switch ($_GET['strona'])  
    {   case "news": $page=new NewsRenderer; break;  
        case "links": $page=new LinksRenderer; break;  
        default: $page=new HomePageRenderer; break;  
    }  
    $page->display();  
?>
```

- Przykład prezentuje pomieszany kod proceduralny i obiektowy, ale w przypadku prostych aplikacji internetowych, takie podejście jest wystarczające

Przykład - klasa PostModel.php

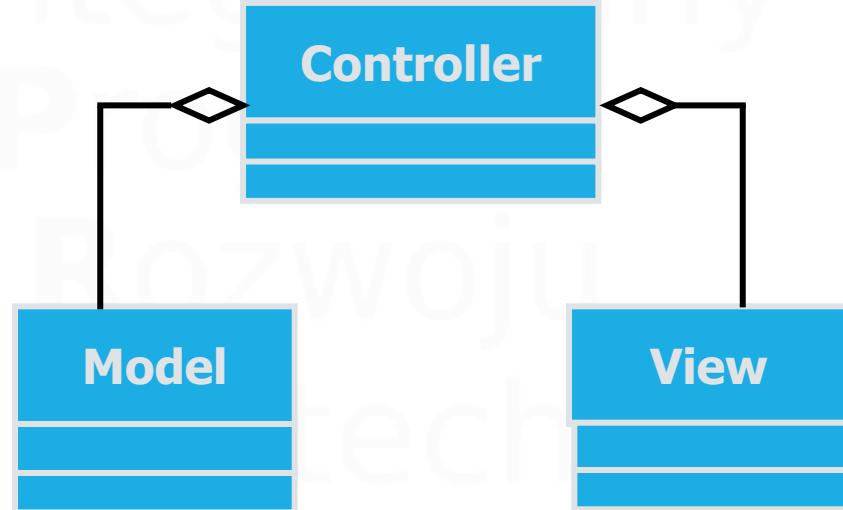
```
class PostModel {  
    private $tytul;  
    private $naglowek;  
    private $tresc;  
    public function __construct($tytul = 'Tytuł',  
        $naglowek = 'Nagłówek', $tresc = 'treść') {  
        $this->tytul = $tytul;  
        $this->naglowek = $naglowek;  
        $this->tresc = $tresc;  
    }  
    function getTytul(){  
        return $this->tytul;    }  
    function setTytul($tytul) {  
        $this->tytul = $tytul;    }  
//pozostałe metody get/set dla pól $tresc i $naglowek  
}
```

Przykład - klasa PostView.php

```
class PostView {  
    private $model;  
  
    public function __construct($model) {  
        $this->model=$model;  
    }  
  
    public function wyswietlWidok() {  
        echo '<html><head><title>' .  
            $this->model->getTytul() . '</title>';  
        echo '</head><body><h1>' .  
            $this->model->getNaglowek() . '</h1>';  
        echo '<p>' . $this->model->getTresc() .  
            '</p></body></html>';  
    }  
}
```

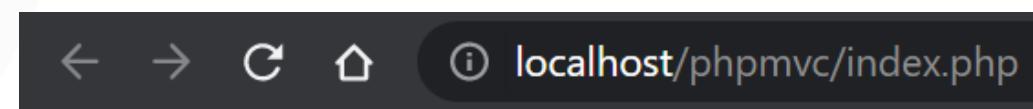
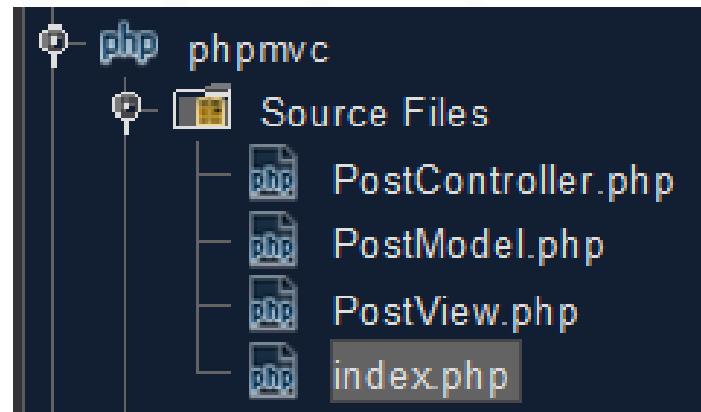
Przykład - klasa PostController.php

```
/** * klasa kontrolera obsługuje model oraz uruchamia widok */
class PostController {
    //obiekty:
    private $model;
    private $view;
    //metody:
    function getModel() {
        return $this->model;    }
    function getView() {
        return $this->view;    }
    function setModel($model) {
        $this->model = $model;    }
    function setView($view) {
        $this->view = $view;    }
    function uruchomWidok(){ $this->view->wyswietlWidok();    }
    public function __construct($model, $view) {
        $this->model = $model; $this->view = $view;    }
}
```



Przykład - wykorzystanie klas MVC

```
<?php  
    //uruchomienie prostego wzorca MVC  
    require_once "PostModel.php";  
    require_once "PostView.php";  
    require_once "PostController.php";  
    $model = new PostModel();  
    $view = new PostView($model);  
    $controller = new PostController($model,$view);  
    $controller->uruchomWidok();
```



Aplikacja MVC - praca z danymi z BD

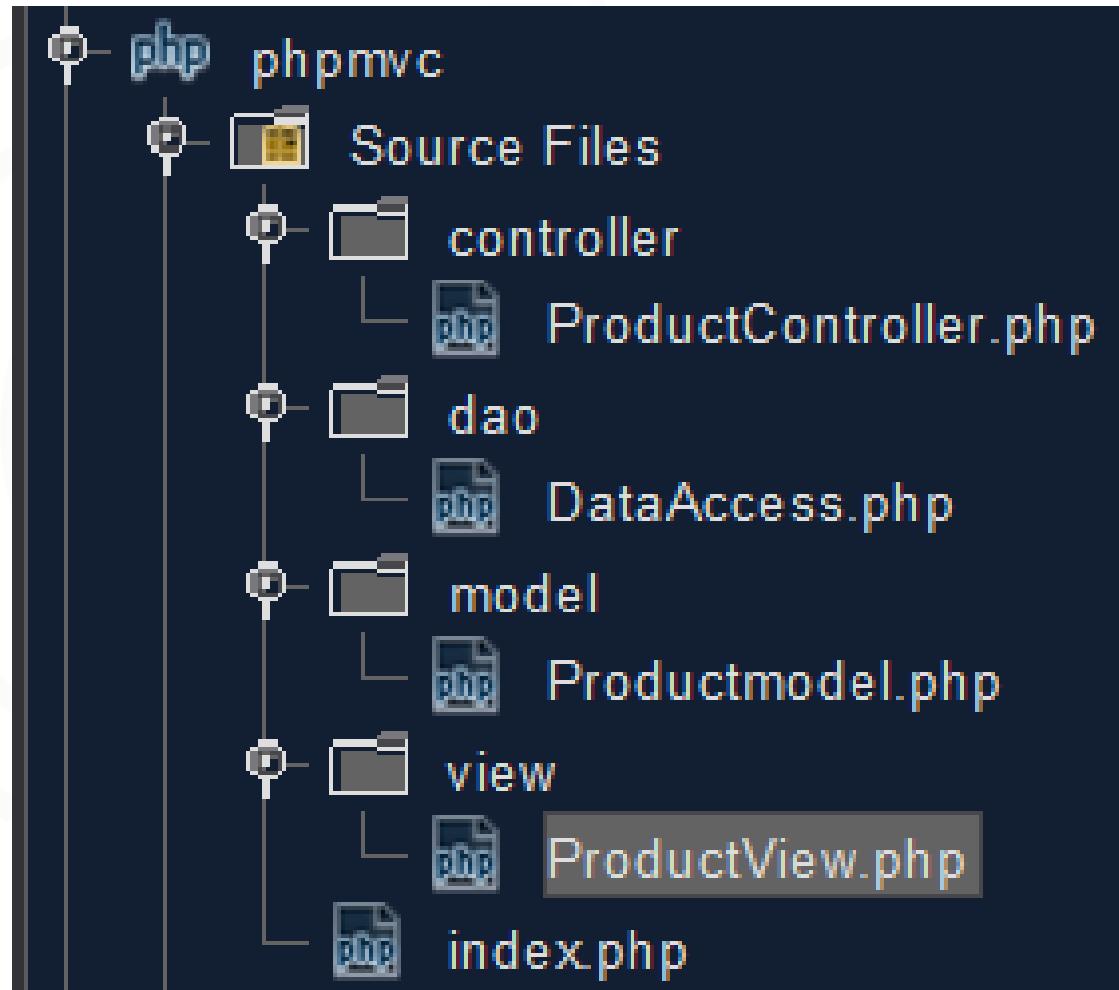


Tabela PRODUCT w bazie Dane

Kod SQL tworzenia tabeli PRODUCT i dodawania przykładowych rekordów:

```
CREATE TABLE PRODUCT(PRODUCTID INTEGER NOT NULL PRIMARY KEY, CODE  
VARCHAR(5), NAME VARCHAR(100), UNITPRICE NUMERIC(6,2) NOT NULL);  
INSERT INTO PRODUCT VALUES(10,'AB123','Leather Sofa',1000);  
INSERT INTO PRODUCT VALUES(20,'AB456','Baby Chair',200.25);  
INSERT INTO PRODUCT VALUES(30,'AB789','Sport Shoes',250.60);  
INSERT INTO PRODUCT VALUES(40,'PQ123','Sony Digital Camera',399);  
INSERT INTO PRODUCT VALUES(50,'PQ456','Hitachi HandyCam',1050);  
INSERT INTO PRODUCT VALUES(60,'PQ789','GM Saturn',2250.99);
```

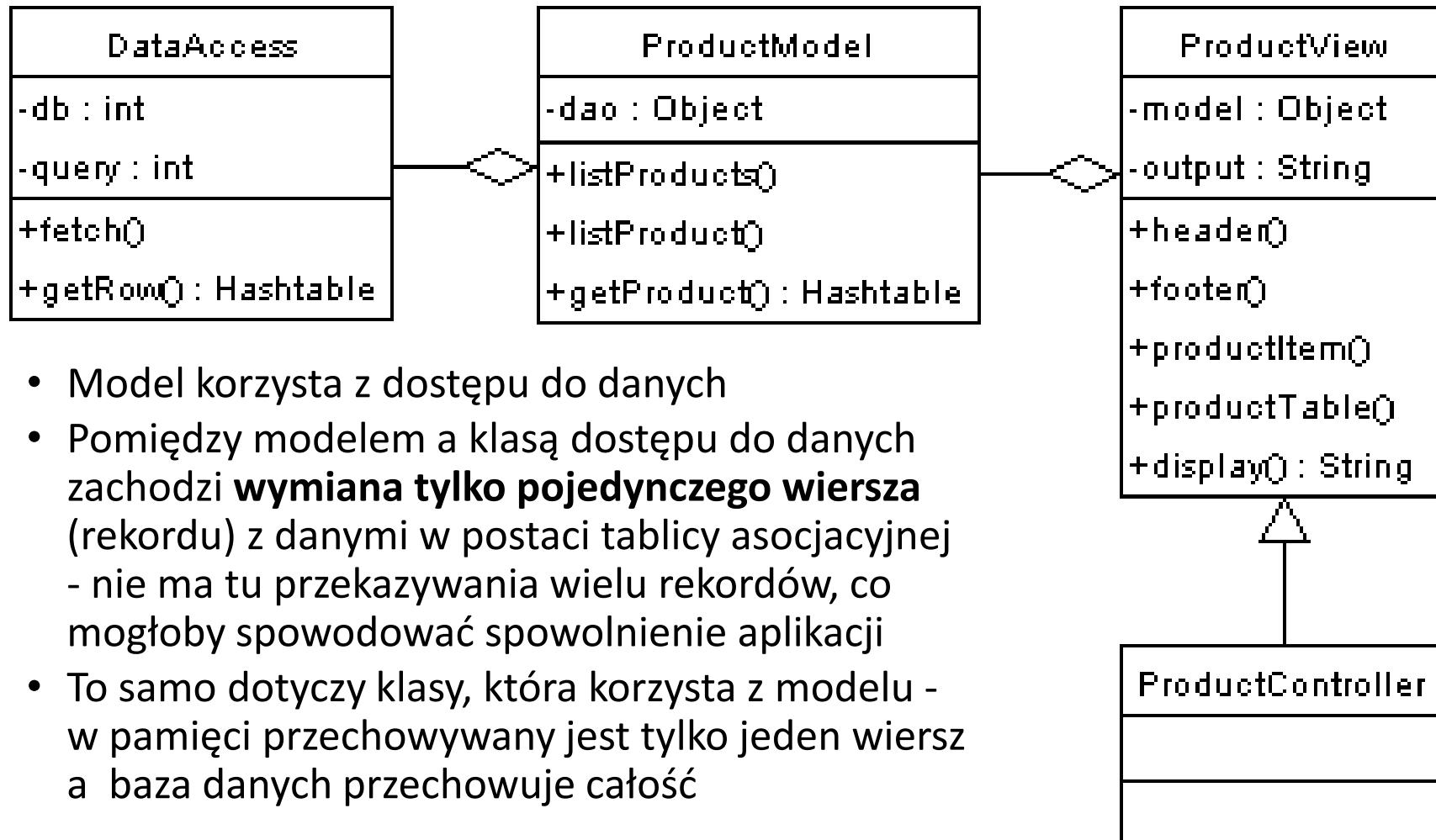
Tabela product w phpMyAdmin

The screenshot shows the phpMyAdmin interface with the following details:

- Server:** 127.0.0.1 » **Baza danych:** dane » **Tabela:** product
- Navigation:** Przeglądaj, Struktura, SQL, Szukaj, Wstaw, Eksport
- Current View:** Struktura tabeli (Structure) is selected.
- Table Structure:**

#	Nazwa	Typ	Metoda porównywania napisów	Atrybuty	Null	Ustawienia domyślne
1	PRODUCTID	int(11)			Nie	Brak
2	CODE	varchar(5)	utf8_general_ci		Tak	NULL
3	NAME	varchar(100)	utf8_general_ci		Tak	NULL
4	UNITPRICE	decimal(6,2)			Nie	Brak

Przykładowe powiązania między klasami MVC



- Model korzysta z dostępu do danych
- Pomiędzy modelem a klasą dostępu do danych zachodzi **wymiana tylko pojedynczego wiersza** (rekordu) z danymi w postaci tablicy asocjacyjnej
 - nie ma tu przekazywania wielu rekordów, co mogłoby spowodować spowolnienie aplikacji
- To samo dotyczy klasy, która korzysta z modelu - w pamięci przechowywany jest tylko jeden wiersz a baza danych przechowuje całość

Klasa DataAccess (1)

```
/** * Klasa do realizacji zapytań do MySQL */
class DataAccess {
    protected $db; //przechowuje uchwyt do bazy danych
    protected $result; //przechowuje wynik zapytania

    //Konstruktor
    /** * Tworzy nowy obiekt DataAccess
        * @param $host string - serwer BD
        * @param $user string - nazwa użytkownika
        * @param $pass string - hasło
        * @param $db string - nazwa BD      */
    function __construct($host,$user,$pass,$db) {
        $this->db=new mysqli($host,$user,$pass,$db);
        $this->db->set_charset("utf8"); //zmień kodowanie
    }
}
```

Klasa DataAccess (2)

```
//Metoda fetch
/** * pobiera wynik zapytania i zapamiętuje w zm. $result
 * @param $sql string - zapytanie do wykonania
 * @return void      */
function fetch($sql) {
    $this->result=$this->db->query($sql); }

//Metoda getRow
/** * zwraca tablicę asocjacyjną reprezentującą pojedynczy
 * wiersz z wyniku zapytania
 * @return mixed      */
function getRow () {
    if ( $row=$this->result->fetch_assoc()) return $row;
    else return false;
}
function __destruct() { $this->db->close(); }
} //koniec klasy DataAccess
```

Klasa ProductModel (1)

```
/** * Klasa reprezentuje model danych */
class ProductModel {
    /** * $dao - instancja klasy DataAccess */
    protected $dao;
    //Konstruktor
    /** * tworzy nowy obiekt ProductModel
     * @param $dao - obiekt klasy DataAccess */
    function __construct($dao) { $this->dao = $dao; }

    //Metoda listProducts
    /** * pobiera listę produktów realizując odpowiednie zapytanie
     * @param $start - numer rekordu startowego
     * @param $rows - liczba rekordów do pobrania
     * @return void */
    function listProducts($start = 1, $rows = 10) {
        $this->dao->fetch("SELECT * FROM product LIMIT " . $start .
            ", " . $rows);
    }
}
```

Klasa ProductModel (2)

```
//Metoda listProduct
/** * pobiera pojedynczy produkt o wskazanym id
 * @param $id a primary key for a row
 * @return void      */
function listProduct($id) {
    $this->dao->fetch("SELECT * FROM product WHERE PRODUCTID= '" .
        $id . "'");
}

//Metoda getProduct
/** * zwraca pojedynczy produkt w postaci tablicy asocjacyjnej
 * @return mixed      */
function getProduct() {
    if ($product = $this->dao->getRow()) return $product;
    else return false;
}
} //koniec klasy ProductModel
```

Klasa ProductView (1)

```
/** *Generuje odpowiedni kod HTML z danych o produkcie */
class ProductView {
    protected $model; //obiekt ProductModel
    protected $output; //wynikowy kod HTML
    //Konstruktor - tworzy nowy obiekt ProductView
    /** * @param $model - instancja klasy ProductModel */
    function __construct($model) { $this->model = $model; }
    //Metoda header - tworzy nagłówek strony HTML
    /** * @return void */
    function header() {
        $this->output = '<!doctype html><html><head>' .
            '<title> Our Products </title>' .
            '<link rel="stylesheet" href="styl.css" type="text/css" />' .
            '</head><body><div class="title">Our Products</div>';
        $this->output.= "<div><a href='index.php'>Start</a></div>";
    }
}
```

Klasa ProductView (2)

```
// Metoda footer - tworzy stopkę dokumentu HTML
/** * @return void */
function footer() {
    $this->output.= "</body></html>";
}

//Metoda productItem - formatuje pojedynczy produkt
/** * @return void */
function productItem($id = 10) {
    $this->model->listProduct($id);
    while ($product = $this->model->getProduct()) {
        $this->output.= "<p><b>Name </b>:" . $product['NAME'] .
            "</p><p><b>Price </b>:" . $product['UNITPRICE'] . "</p>" .
            "<p><b>Code </b>:" . $product['CODE'] . "</p>";
    }
}
```

Klasa ProductView (3)

```
//Metoda - formatuje tabelę produktów
/**      * * @return void      */
function productTable($rownum = 1) {
    $rowsperpage = '10';
    $this->model->listProducts($rownum, $rowsperpage);
    $this->output.= "<table ><tr><th>Name</th>
                            <th>Price</th></tr>";
    while ($product = $this->model->getProduct()) {
        $this->output.= "<tr><td><a href='?view=product&id=' .
            $product['PRODUCTID'] . "' >" .
            $product['NAME'] . "</a></td>" . "<td>" .
            $product['UNITPRICE'] . "</td></tr>";
    }
    $this->output.= "<tr>";
```

Klasa ProductView (4)

```
if ($rownum != 0 && $rownum > $rowsperpage) {  
    $this->output.= "<td><a href='?view=table&rownum=" .  
                    ($rownum - $rowsperpage) . "'> Prev</a></td>"; }  
else { $this->output.= "<td>&ampnbsp</td>"; }  
if ($product['PRODUCTID'] < ($rownum + $rowsperpage)) {  
    $this->output.= "<td><a href='?view=table&rownum=" .  
                    ($rownum + $rowsperpage) . "'>Next </a></td>"; }  
else { $this->output.= "<td>&ampnbsp</td>"; }  
$this->output.= "</tr></table>";  
}  
  
//Metoda - zwraca gotową stronę HTML  
/** * * @return string */  
function display() {  
    return $this->output;  
}  
} //koniec klasy ProductView
```

Klasa ProductController

```
class ProductController extends ProductView {  
    //Konstruktor - tworzy nowy obiekt klasy ProductController  
    /** * @param $model - obiekt klasy ProductModel  
     * @param $getvars - tablica parametrów żądania HTTP GET */  
    function __construct($model, $getvars = null) {  
        parent::__construct($model);  
        $this->header();  
        $view = @$getvars['view']; $id = @$getvars['id'];  
        if (empty($view)) $view = "";  
        if (empty($id)) $id = 10;  
        switch ($view) {  
            case "product": $this->productItem($id); break;  
            default: if (empty($getvars['rownum'])) {  
                $this->productTable(); }  
            else {  
                $this->productTable($getvars['rownum']); }  
            break;  
        }  
        $this->footer();  
    }  
} //koniec klasy ProductController
```

Skrypt główny index.php

```
<?php
    require_once('dao/DataAccess.php');
    require_once('model/ProductModel.php');
    require_once('view/ProductView.php');
    require_once('controller/ProductController.php');

    $dao = new DataAccess('localhost', 'root', '', 'dane');
    $productModel = new ProductModel($dao);
    $productController = new
        ProductController($productModel, $_GET);
    echo $productController->display();
?>
```

MVC w akcji

Our Products

[Start](#)

Name	Price
Baby Chair	200.25
Sport Shoes	250.60
Sony Digital Camera	399.00
Hitachi HandyCam	1050.00
GM Saturn	2250.99
Baby Chair	200.25
Sport Shoes	250.60
Sony Digital Camera	399.00
Hitachi HandyCam	1050.00
GM Saturn	2250.99

[Next](#)

Our Products

[Start](#)

Name Price

[Baby Chair](#) 200.25

← → C ⌂ ⓘ localhost/phpmvc/index.php?view=product&id=20

Our Products

[Start](#)

Name :Baby Chair

Price :200.25

Code :AB456

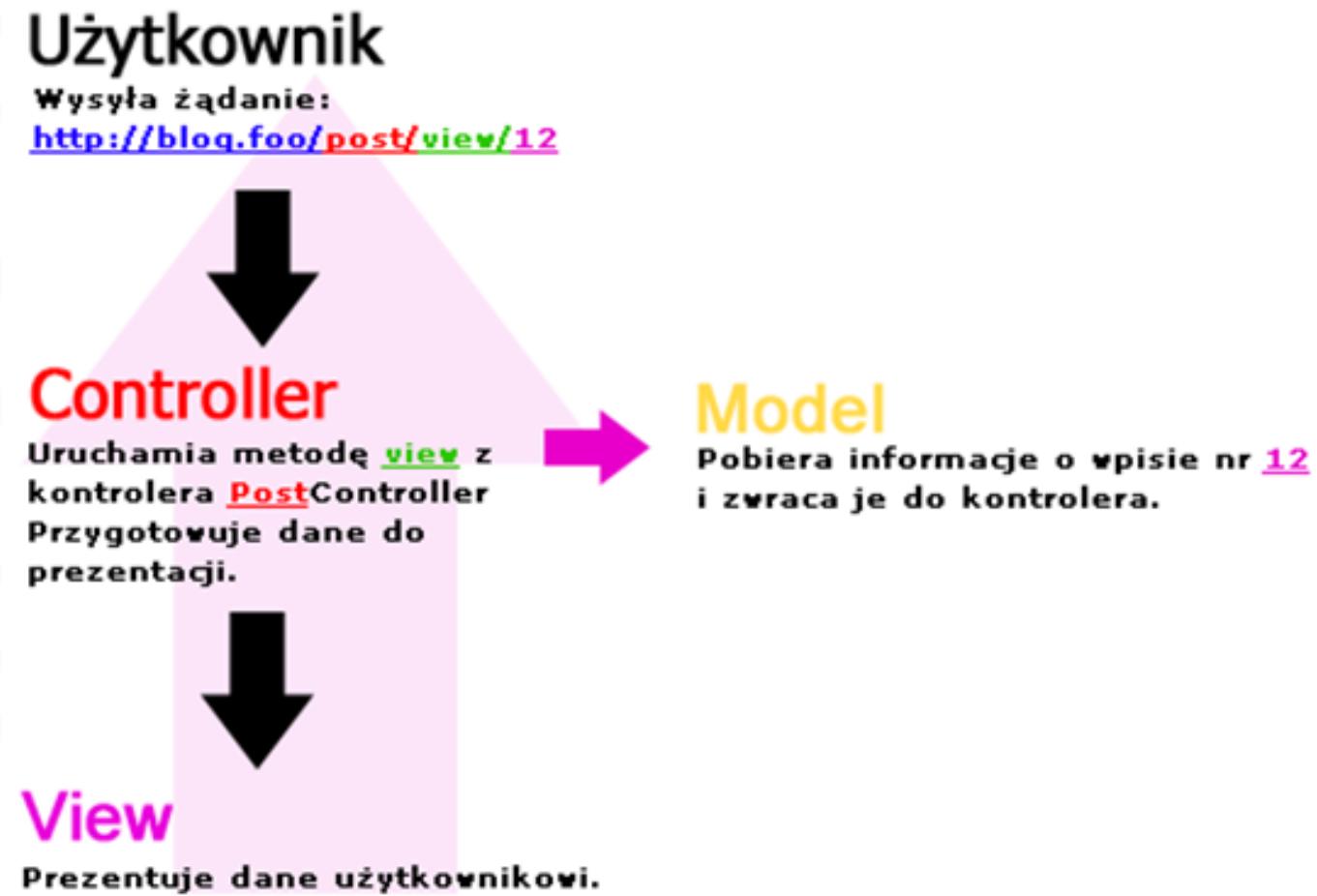
[Hitachi HandyCam](#) 1050.00

[GM Saturn](#) 2250.99

[Prev](#) [Next](#)

Interakcja pomiędzy obiektami MVC w aplikacji internetowej

Sytuacja:
Użytkownik
chce obejrzeć
wpis na blogu
o id=12



MVC w aplikacjach internetowych - fundamentalne zasady

- Najczęściej tworząc aplikacje internetowe w szkieletach programistycznych deweloper jest odpowiedzialny za pisanie kodu, który mapuje żądania użytkownika na odpowiednie strony z nimi związane
- Kod wykonywalny jest interpretowany jako **akcje** (ang. actions) obsługiwane przez metody klasy kontrolerów
- **Mapowanie żądań użytkownika na wywołanie odpowiednich akcji** (zdefiniowanych w metodach kontrolerów) jest realizowane poprzez mechanizm tzw. **routingu** a wynik (widok) wyświetlany w przeglądarce jest renderowany zwykle za pomocą odpowiednich technologii szablonów (ang. templates) np. w PHP są to Smarty, Twig, Blades, w Javie JSP, JSF, Thymeleaf, w C# - Razor

Elastyczny mechanizm routingu

Obsługa żądania HTTP i routing URL



- **Routing (trasowanie)** udostępnia użytkownikom systemu **przyjazne adresy URL** oraz przekierowuje ich żądania do odpowiednich akcji kontrolerów
- Nazwa domenowa stosowana w routingu jest:
 - łatwa do zapamiętania i informuje o zawartości strony
 - odzwierciedla strukturę aplikacji (użytkownik może ręcznie modyfikować URL i łatwo nawigować w tej strukturze)
 - nie ulega zmianie wraz z upływem czasu i modyfikacją aplikacji
- Mechanizm routingu dopasowuje podane w adresie parametry do wskazanego wzorca

Routing a wybór kontrolera

/Home/Index	→ controller: Home, action: Index
/Home	→ controller: Home, action: Index
/	→ controller: Home, action: Index
/About	→ controller: About, action: Index
/News>Show/2	→ controller: News, action: Show, id: 2

Typ (klasa kontrolera) wybierana jest na podstawie routingu:

/Home/Index

controller: Home → klasa: HomeController

/News>Show/2

controller: News → klasa: NewsController

Wybór kontrolera

/Home/Index

controller: Home → klasa: HomeController

/News>Show/2

controller: News → klasa: NewsController

Konwencja

Ustawienie reguł routingu na przykładzie frameworka Laravel

//w pliku konfiguracyjnym Laravel v8:

```
Route::get('/number', [NumberController::class, 'number']);
```

URL

Nazwa klasy kontrolera

Metoda kontrolera

//Klasa kontrolera z metodą number()

```
class NumberController extends Controller {
```

```
    public function number() {
```

```
        $number = random_int(0, 100);
```

```
        return view('number', [ 'number' => $number ]);
```

```
}
```

```
}
```

Strona widoku

Przekazanie danych do widoku

Algorytm obsługi żądań we frameworkach

1. **Przygotowanie reguł routingu** - dodanie tras (np. `/number`, `/show/id`) do pliku konfiguracyjnego frameworka
2. **Utworzenie klas kontrolerów** do obsługi żądań (np. klasy `NumberController` z metodą `number()`). Klasy kontrolerów są zwykle umieszczane w folderze `controllers`
3. Przygotowanie klas dostępu do danych (DAO), z których metod korzystają akcje kontrolerów do pracy z danymi. Dane są reprezentowane przez klasy modeli (np. User, Product). Najczęściej do pracy z danymi framework korzystają z technologii **ORM** (ang. Object-Relation Mapping), o czym będzie mowa na ostatnim wykładzie
4. Przygotowanie plików z widokami, zwykle z zastosowaniem odpowiedniego **silnika szablonów (widoków)** w zależności od stosowanego frameworka. Widoki umieszczane są zwykle w folderze o nazwie `views` (lub `templates`). Pliki widoków to dokumenty o strukturze HTML z możliwością pobierania danych z modeli, utworzonych przez akcje kontrolera



PROGRAMOWANIE APLIKACJI INTERNETOWYCH

Wykład 10

Transmisja danych z serwera w trybie asynchronicznym -
dynamiczna interakcja z użytkownikiem za pomocą obiektu
Ajax, jQuery i PHP.

Beata Pańczyk



**Fundusze
Europejskie**
Wiedza Edukacja Rozwój



**Rzeczpospolita
Polska**

Unia Europejska
Europejski Fundusz Społeczny



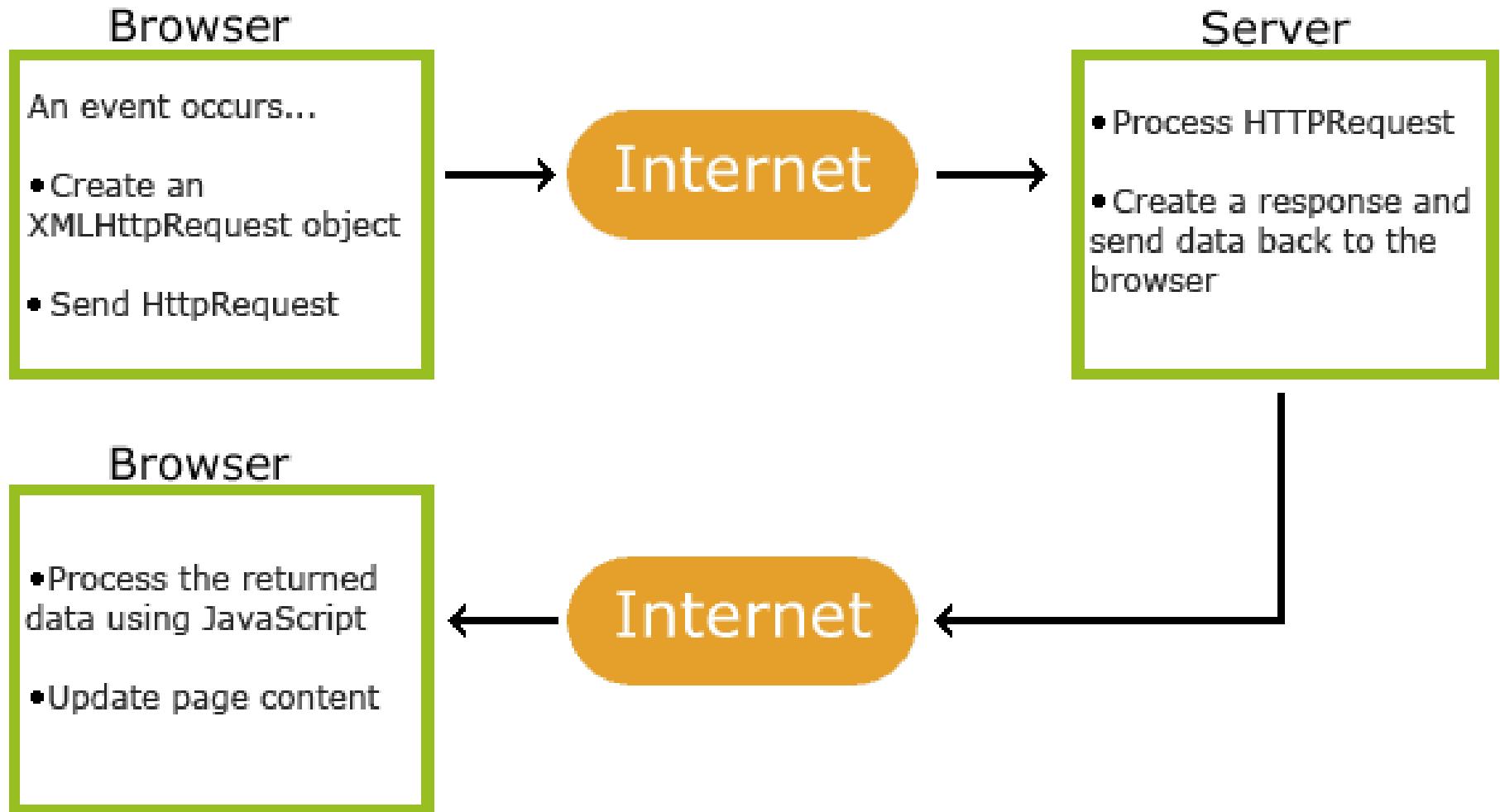
HTTP Request - przypomnienie

Na żądanie HTTP składają się 4 elementy:

1. Punkt końcowy (ang. endpoint or route) - żądany URL
2. Metody HTTP (ang. method) - typ żądania (GET, POST, PUT, PATCH, DELETE)
3. Nagłówki HTTP (ang. headers) - służą do dostarczania informacji zarówno klientowi, jak i serwerowi. Mogą być wykorzystywane do uwierzytelniania, dostarczania informacji o zawartości treści, itp.
4. Dane lub ciało żądania (ang. data or body) - nazywane też „treścią” lub „wiadomością”, zawiera informacje, które mają być wysłane na serwer. Ta opcja jest używana tylko z żądaniami POST, PUT, PATCH lub DELETE

AJAX - żądania asynchroniczne

http://www.w3schools.com/ajax/ajax_intro.asp



JavaScript i interfejs FetchAPI do obsługi żądań asynchronicznych

- Do obsługi żądań HTTP w trybie asynchronicznym wykorzystuje się interfejs Fetch API i obiekty **Promise**, które otrzymują wynik w momencie pobrania odpowiedzi z serwera
- Obiekty **Promise** pozwalają na tworzenie łańcuchów kodu obsługującego dane z zapytań AJAX i dzięki temu można odpowiednio przetwarzać dane z zapytań
- Dzięki wykorzystaniu obiektów **Promise** bazujemy na obietnicy, że te dane mogą się pojawić i na tej podstawie definiujemy jakie operacje będą do wykonania na tych danych

Interfejsy powiązane z Fetch API

- W skład Fetch API wchodzą dwa interfejsy (obiekty JavaScript), które można przekazać podczas wykonywania zapytania do serwera:
 - **Headers**
 - **Request**
- oraz jeden obiekt, który może być zwrócony przy odpowiedzi z serwera:
 - **Response** - obiekt tego interfejsu jest przekazywany jako odpowiedź z serwera i zawiera wszystkie informacje zwrotne

Schemat pracy z Fetch API - pobieranie danych tekstowych

```
fetch("http://localhost/ajax/dane.txt")
  .then(response => {
    if (response.status !== 200) {
      return Promise.reject('Zapytanie się nie powiodło');
    }
    return response.text();
  })
  .then((dane) => {
    console.log(dane);
  })
  .catch((error)=>{
    console.log(error);
  });
}
```

Schemat pracy z Fetch API - wysyłanie danych na serwer

- W celu wysłania danych na serwer należy je dodać do właściwości **body** w opcjach metody fetch
- Format danych zależy jest od wybranej metody kodowania. Jeżeli nie ustawiono żadnego nagłówka, dane muszą być zakodowane w postaci URL:

```
fetch(url, {  
    method: "post",  
    body: uriEncode("name=Ania&surname=Nowak")  
} )
```

- Chcąc wysłać dane jako JSON, należy ustawić odpowiedni nagłówek i zakodować dane za pomocą JSON.stringify()
- Po stronie serwera, pod wskazanym adresem URL powinien się znajdować skrypt obsługujący dane

Wysyłanie danych JSON

```
const produkt = { nazwa : "Gruszki", cena : 10  }
//obiekt JS
fetch(url, {
    method: "post",
    headers: {
        "Content-Type": "application/json"
    },
    body: JSON.stringify(produkt)
})
.then(response => response.json() )
.then(response => {
    console.log("Wysłano produkt:");
    console.log(response);
})
```

Interfejs Form Data

- Interfejs udostępnia metody ułatwiające zarządzanie danymi do wysłania. Dzięki **FormData** bardzo prosto dodaje się dane do wysłania i dodatkowo nie trzeba ustawiać nagłówka - dane zawsze są wysyłane za pomocą kodowania **multipart/form-data**
- Wybrane metody interfejsu:
 - **append(key, value)** - dodaje nową wartość o danym kluczu do wysyłanych zmiennych
 - **delete(key)** - usuwa wartość o danym kluczu
 - **entries()** - zwraca iterator, który umożliwia realizację pętli po wszystkich parach klucz/wartość
 - **get(key)** - zwraca pierwszą wartość o danym kluczu
- Więcej na ten temat na stronie:
<http://kursjs.pl/kurs/ajax/xmlhttprequest.php#formData>

Wysyłanie danych - FormData

<http://kursjs.pl/kurs/ajax/fetch.php>

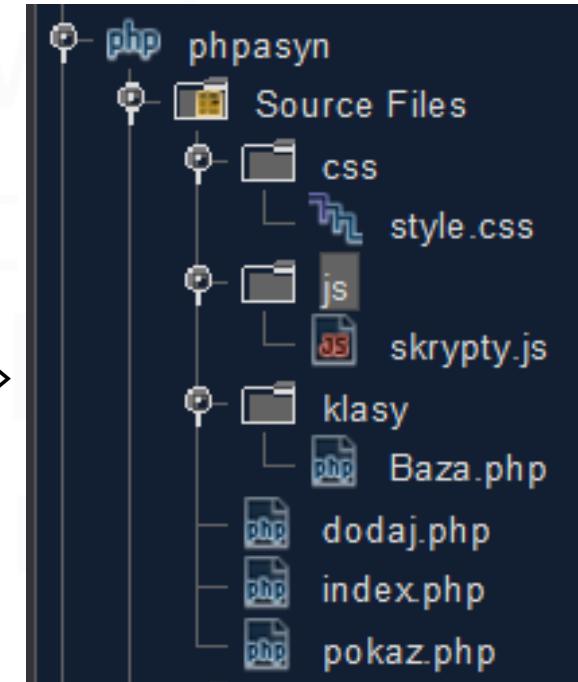
```
const formData = new FormData();
formData.append("title", "Test PAI 2021");
formData.append("body", "Test przesyłania danych");
formData.append("userId", 1);
fetch("https://jsonplaceholder.typicode.com/posts", {
    method: "post",
    body: formData
})
.then(response => response.json())
.then(response => {
    console.log("Dodano użytkownika:");
    console.log(response);
})
```

JSON Placeholder - dostępny online interfejs REST API do testowania i prototypowania:

- <https://jsonplaceholder.typicode.com/>
- <https://jsonplaceholder.typicode.com/guide.html>

Ajax, PHP i BD - index.php i struktura projektu

```
<!DOCTYPE html><html><head>
<meta charset="UTF-8">
<title>PHP i Ajax</title>
<meta name="viewport" content="width=device-width,
    initial-scale=1.0">
<link rel="stylesheet" href="css/style.css" />
</head>
<body>
<div>
    Tytuł artykułu: <input id="tytul"/> <br />
    Autor: <input id="autor"/> <br />
    Treść: <input id="tresc"/> <br />
    <button id="dodaj"> Dodaj nowy artykuł</button>
    <button id="pokaz"> Pokaż wszystkie artykuły
    </button>
</div>
<div id="main"></div>
<script src="js/skrypty.js"></script>
</body></html>
```



Ajax, PHP i BD - skrypty.js (1)

```
//Fetch API
document.addEventListener('DOMContentLoaded', () => {
    var bdodaj = document.getElementById('dodaj');
    bdodaj.addEventListener("click", () => {
        console.log("Dodawanie artykułu");
        dodaj();
    });
    var bpokaz = document.getElementById('pokaz');
    bpokaz.addEventListener("click", () => {
        console.log("Pobranie wszystkich artykułów");
        pokaz();
    });
});
```

Ajax, PHP i BD - skrypty.js (2)

```
function dodaj() {
    const formData = new FormData();
    formData.append("tytul", document.getElementById('tytul').value);
    formData.append("autor", document.getElementById('autor').value);
    formData.append("tresc", document.getElementById('tresc').value);

    fetch("http://localhost/phpsyn/dodaj.php", {
        method: "post",
        body: formData
    })
    .then( response => response.text())
    .then(response => {
        console.log("Dodano artykuł:");
        console.log(response);
    });
}
```

Ajax, PHP i BD - skrypty.js (3)

```
function pokaz() {
    fetch("http://localhost/phpsyn/pokaz.php")
        .then((response) => {
            if (response.status !== 200) {
                return Promise.reject('Coś poszło nie tak!');
            }
            return response.text();
        })
        .then((data) => {
            document.getElementById('main').innerHTML = data;
        })
        .catch((error) => {
            console.log(error);
        });
}
```

Ajax, PHP i BD - pokaz.php

← → C ⌂

localhost/phpasyn/index.php



Tytuł artykułu:

Autor:

Treść:

```
<?php  
include_once "klasy/Baza.php";  
$db = new Baza('localhost', 'root', '', 'dane');  
$sql = "select * from news";  
echo $db->select($sql,  
["Data", "Autor", "Nagłówek"]);
```

Dodaj nowy artykuł

Pokaż wszystkie artykuły

2021-07-14 Beata Pańczyk Artykuł o PHP

2021-07-06 Maciej Pańczyk Artykuł o systemach operacyjnych

2021-07-15 Beata Pańczyk Szkielet programistyczny Laravel

2021-07-14 Ola Olewska Aplikacje typu MPA

Ajax, PHP i BD - dodaj.php

```
<?php
include_once "klasy/Baza.php";
$db = new Baza('localhost', 'root', '', 'dane');
$tytul = filter_input(INPUT_POST, 'tytul', FILTER_SANITIZE_ADD_SLASHES);
$tresc = filter_input(INPUT_POST, 'tresc', FILTER_SANITIZE_ADD_SLASHES);
$autor = filter_input(INPUT_POST, 'autor', FILTER_SANITIZE_ADD_SLASHES);
if (!$autor && $tytul && $tresc) {
    echo "Nie wypełniono niektórych pól";
} else { $data = (new DateTime())->format("Y-m-d");
    $sql = "INSERT INTO news
VALUES(NULL, '$tytul', '$tresc', '$data', '$autor')";
    if ($db->insert($sql)) {
        print("<h4> Dane zostały wpisane!</h4>");
    } else {
        print("<h5> Pojawił się problem z dodaniem artykułu.
Spróbuj ponownie za jakiś czas. </h5>"); }
}
```

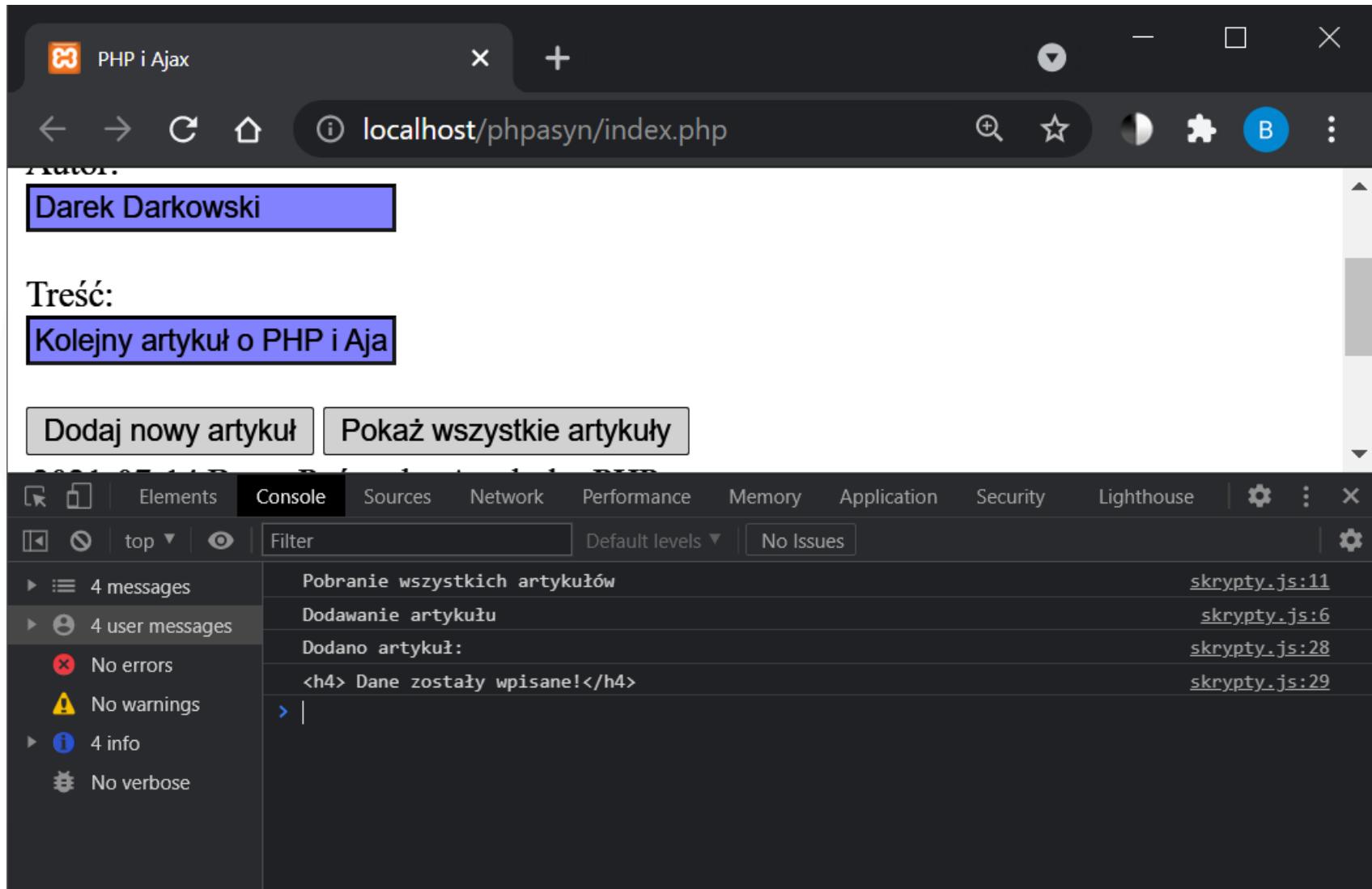
Widok żądań asynchronicznych

The screenshot shows a browser window titled "PHP i Ajax" displaying a local PHP application at "localhost/phpasyn/index.php". The page content includes a header with "Darek Darkowski", a section titled "Treść:" with "Kolejny artykuł o PHP i Aja", and two buttons: "Dodaj nowy artykuł" and "Pokaż wszystkie artykuły". Below the page content, the browser's developer tools Network tab is open, showing a list of requests. The requests listed are:

Name	Method	St...	Protoc...	Type	Initiator	Size	Time	Waterfall
style.css	GET	200	http/1.1	stylesheet	index.php	(mem...)	0 ms	
skrypty.js	GET	200	http/1.1	script	index.php	(mem...)	0 ms	
favicon.ico	GET	200	http/1.1	x-icon	Other	31.2 kB	2 ms	
pokaz.php	GET	200	http/1.1	fetch	skrypty.js:34	665 B	8 ms	
dodaj.php	POST	200	http/1.1	fetch	skrypty.js:22	285 B	9 ms	

At the bottom of the Network tab, summary statistics are displayed: 6 requests, 33.1 kB transferred, 33.6 kB resources, Finish: 1.5 min, DOMContentLoaded: 47 ms, and Load: 48 ms.

Przykładowe komunikaty w konsoli JS



PHP, Ajax i jQuery

- Metoda ajax() w jQuery umożliwia pracę w trybie asynchronicznym

```
$.ajax({name:value, name:value, ... })
```

- Parametry metody:

https://www.w3schools.com/jquery/ajax_ajax.asp

- Np.:

```
 $("button").click(function(){
    $.ajax({ url: "dane.php", success:
        function(result) {
            $("#div1").html(result);
        }
    });
});
```



PROGRAMOWANIE APLIKACJI INTERNETOWYCH

Wykład 11

Przegląd aktualnych platform programistycznych do tworzenia aplikacji internetowych. Wstęp do mapowania obiektowo relacyjnego (ORM).

Beata Pańczyk



Szkielety programistyczne (frameworki)

- Dobry framework - podstawa w programowaniu komercyjnych aplikacji webowych
- Przyśpieszają proces wytwarzania oprogramowania, pozwalają na wykorzystanie sprawdzonych rozwiązań
- Formują ujednoliconą architekturę informacji, np. wykorzystując model MVC pozwalają na pisanie lepszego, mniej wadliwego kodu
- Skracają czas debugowania kodu

Przykładowe frameworki/biblioteki

Server-side

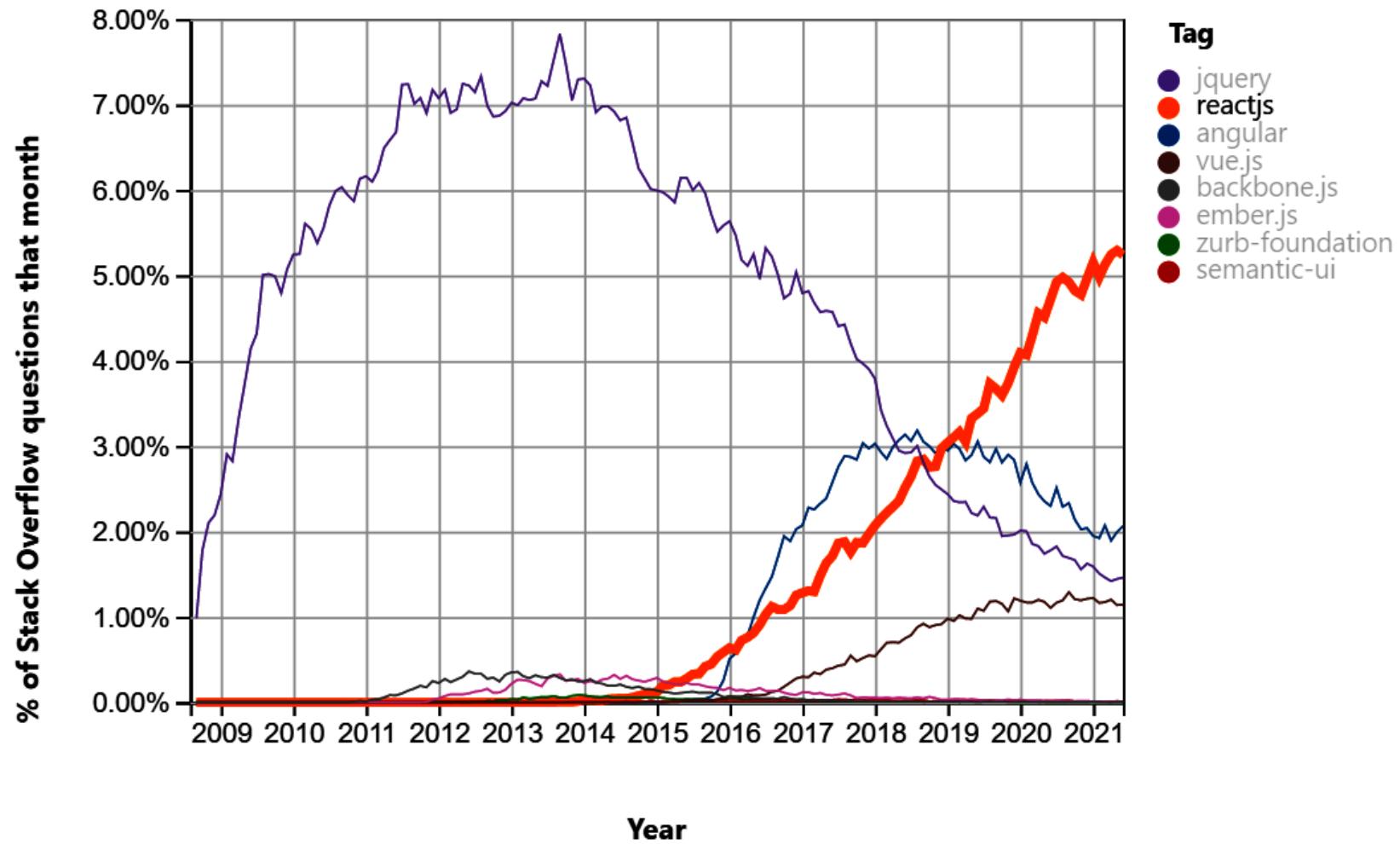
- PHP:
 - Symfony
 - Zend Framework
 - CakePHP
 - CodeIgniter
 - Kohana
- Java:
 - Struts
 - Spring
 - JSF
- C#:
 - ASP.NET
 - ASP.NET MVC
- Ruby: Ruby on Rails
- Python: Django
- JavaScript: Node.js

Front-end

- JavaScript:
 - jQuery
 - React
 - Angular.js/Angular
 - Vue
 - Ember
 - Backbone
 - Svelte

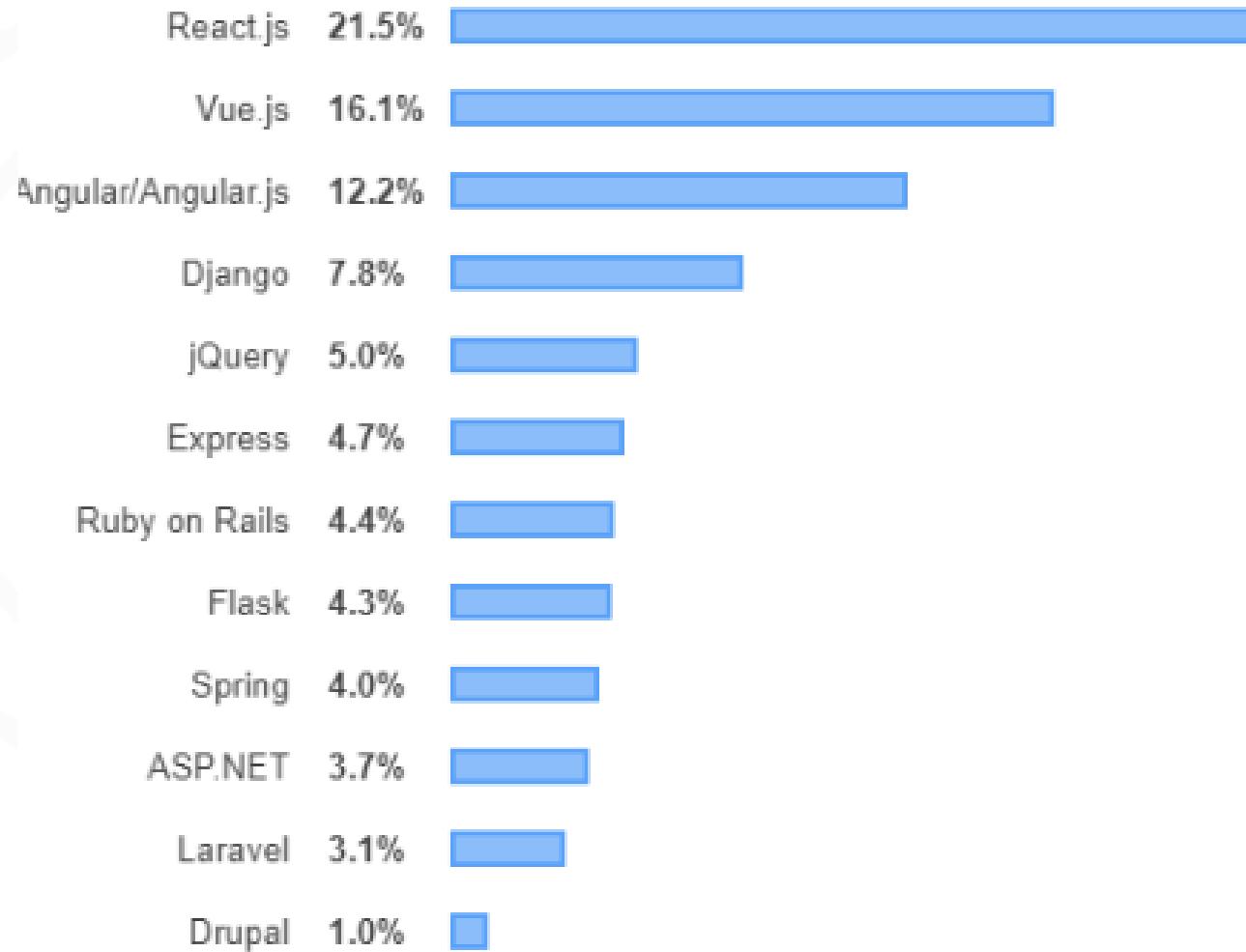
Popularność frameworków front-end

<https://www.simform.com/best-frontend-frameworks/>



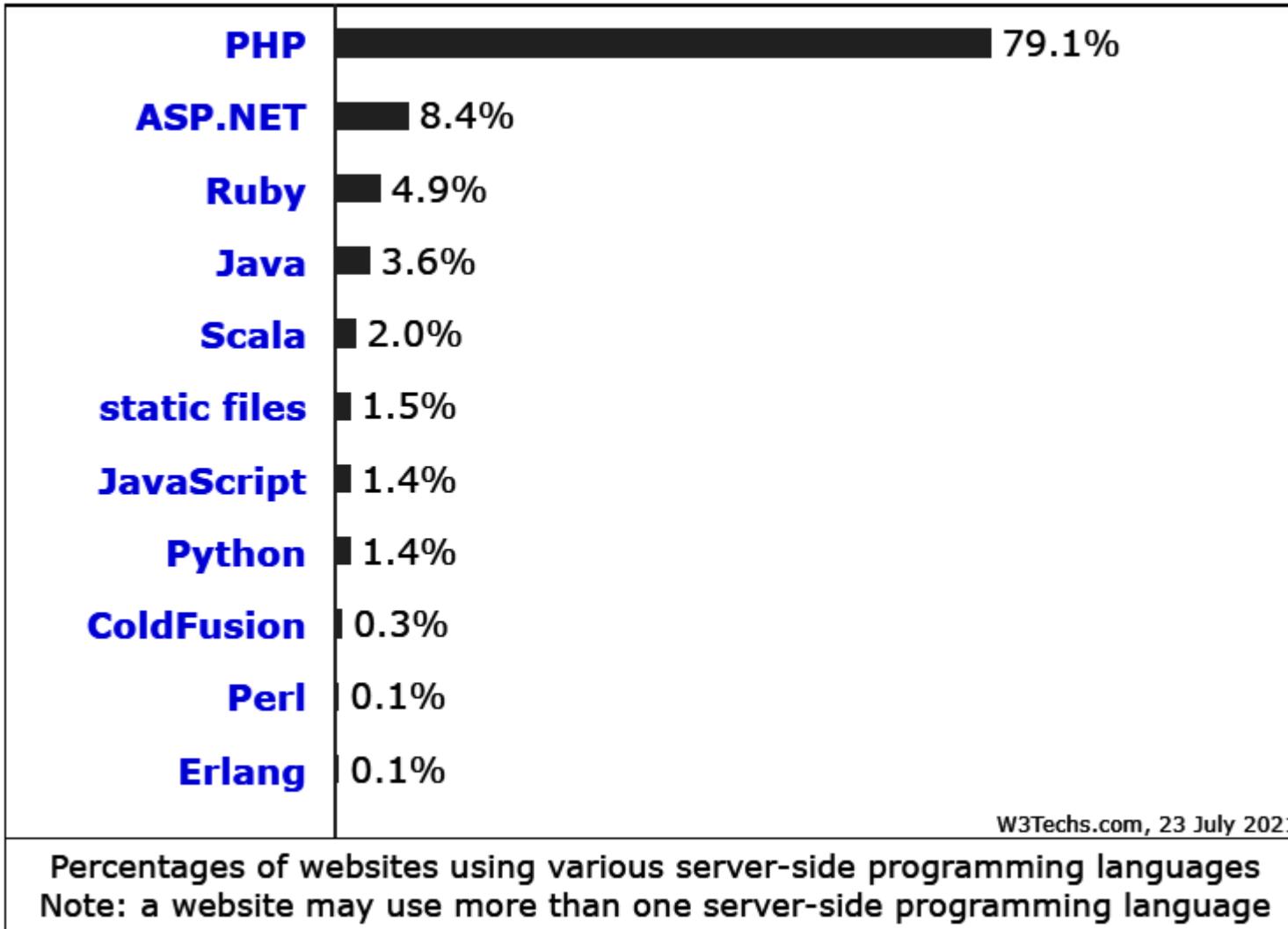
Popularność frameworków w 2020

<https://existek.com/blog/top-front-end-frameworks-2020/>



Server-side programming languages

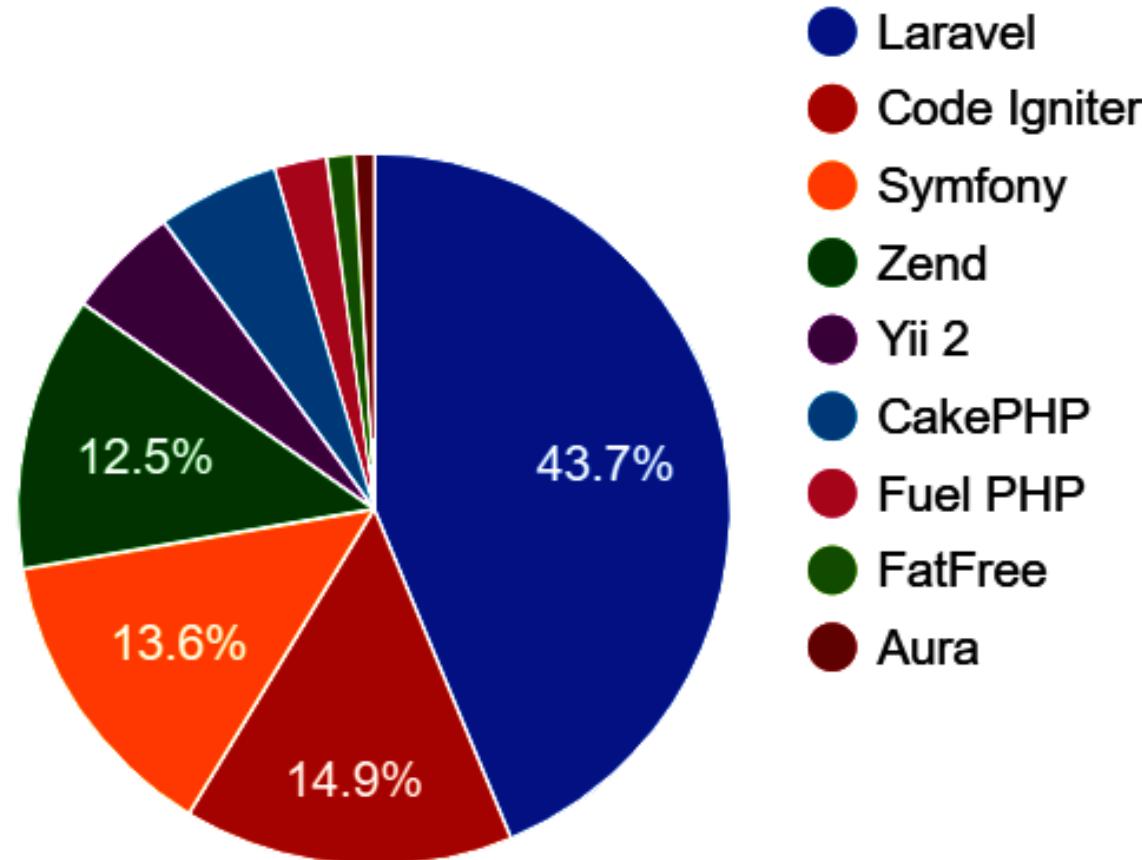
https://w3techs.com/technologies/overview/programming_language/all



Popularność frameworków PHP w 2020

<https://coderseye.com/best-php-frameworks-for-web-developers/>

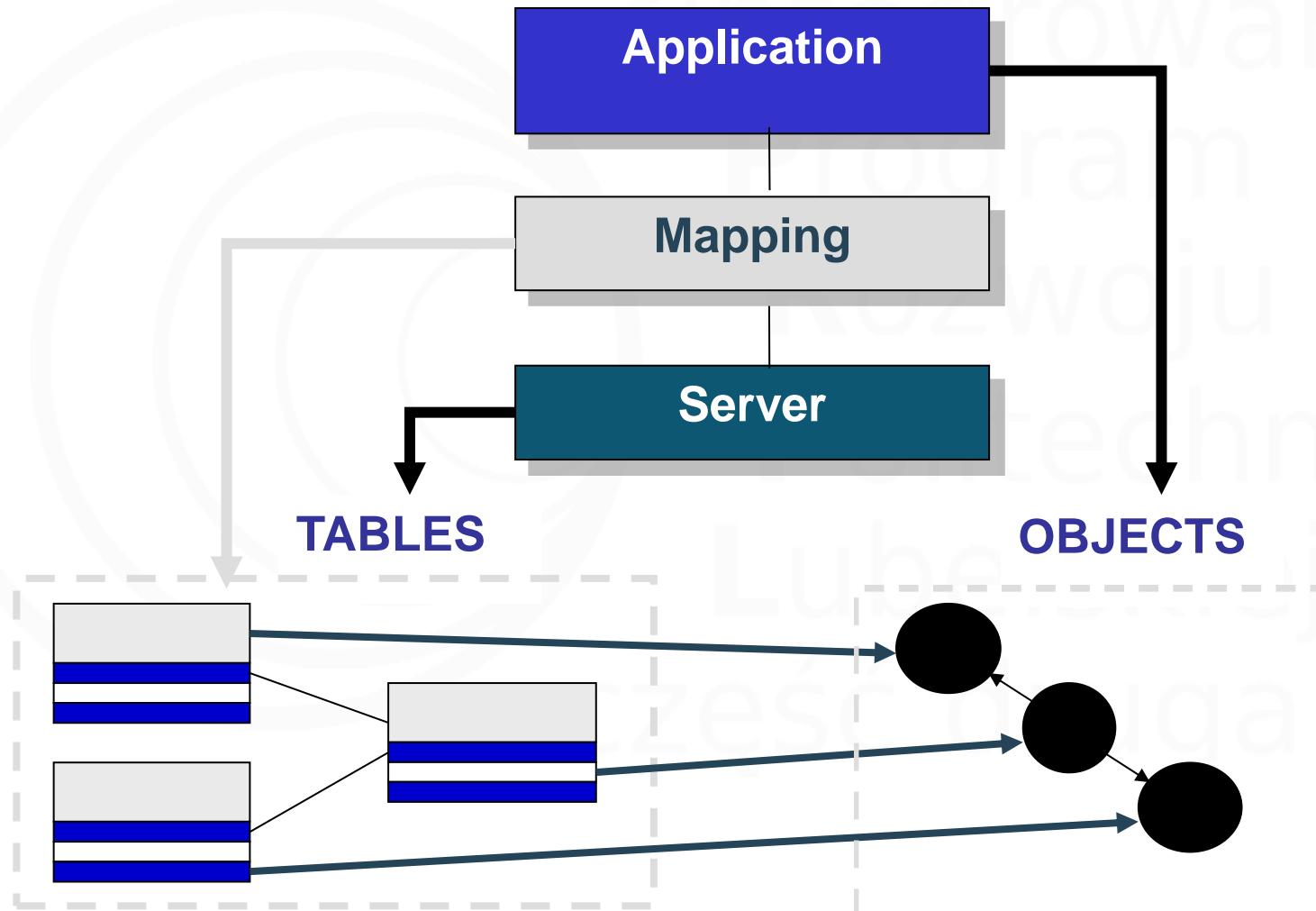
PHP Framework Used for Project Use



Mapowanie obiektowo-relacyjne (ORM)

- ORM, czyli mapowanie obiektowo-relacyjne (ang. Object-Relational Mapping ORM) to sposób odwzorowania obiektowej architektury systemu informatycznego na bazę danych (lub inny element systemu) o relacyjnym charakterze
- Implementacja takiego odwzorowania stosowana jest m.in. w przypadku, gdy tworzony system oparty jest na podejściu obiektowym, a system bazy danych operuje na relacjach
- Przykładowe technologie: Doctrine/Propel/Eloquent (PHP), Hibernate (Java), LINQ (.NET)

Mapowanie obiektowo-relacyjne



Odwzorowanie tabeli - obiekt

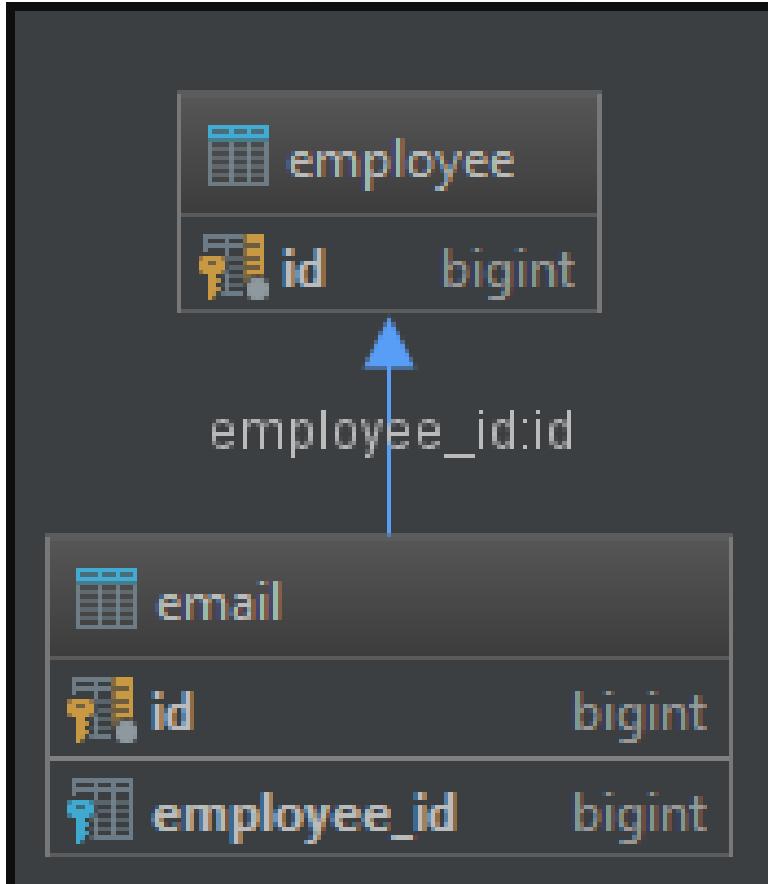
- **Tabela User w BD**

Pole	Typ
id	int(11) NOT NULL PRIMARY KEY
imie	varchar(25) NOT NULL
nazwisko	varchar(35) NOT NULL

- **Klasa User w aplikacji:**

```
class User{ private $id; //typu int  
           private $imie; //typu String  
           private $nazwisko; //typu String  
           //obowiązkowo musi być konstruktor bezparametryowy  
           public function getId() { return $this->id; }  
           public function setId($id): void { $this->id = $id; }  
           //obowiązkowo pozostałe metody get.set  
}
```

Idea ORM – relacje w tabelach i klasach



Klasy (modele) w aplikacji:

```
class Employee {  
    private id;  
    private emails; //tablica obiektów Email  
    //obowiązkowo metody get/set  
    //definicja relacji OneToMany  
}  
class Email {  
    //definicja relacji  
    //z klasą Employee  
    //JoinColumn - employee_id  
    private employee; //typu Employee  
}
```

Kiedy stosować ORM?

- Używanie narzędzia mapującego wiąże się z napisaniem i pielęgnowaniem metainformacji (często w postaci pliku XML) - przy standardowym podejściu, wszystkie meta informacje są rozrzucone po całej aplikacji w instrukcjach SQL
- Mimo początkowo większego nakładu pracy, podejście oparte o ORM okazuje się łatwiejsze w utrzymaniu i rozwoju aplikacji
- Biblioteki ORM są bardzo pomocne **w dużych projektach**, natomiast **w przypadku małych mogą stanowić niepotrzebną komplikację**

Zalety ORM:

- Mniejszy wysiłek i czas zaoszczędzony na pisanie aplikacji korzystającej z bazy danych
- Niezależność aplikacji od konkretnego systemu bazy danych daje możliwość uruchamiania oprogramowania na MySQL, PostgreSQL, Oracle itp., a dodatkowo zabezpiecza w pewnym stopniu przed przykrymi skutkami zmian nazw tabel i ich pól
- Brak tej zalety - jeśli w skryptach umieszczone są instrukcje SQL charakterystyczne dla dialekta określonego systemu baz danych

Wady ORM

- Wydajność - w przypadku niektórych zapytań SQL kod wygenerowany automatycznie przez warstwę pośrednią jest mniej efektywny niż napisany ręcznie
- Wydajność konkretnego programu zależy w dużej mierze od fizycznego modelu danych i może być wyznaczona dopiero w fazie testowania
- Ogólna zasada - narzędzia ORM najlepiej stosować dla stron, na których występuje dużo zapytań o pojedyncze obiekty z różnych tabel (np. sklep internetowy), natomiast nie są one efektywne w aplikacjach, gdzie nie pracuje się z obiektami, a z danymi tabelarycznymi (np. raporty, zestawienia) - wtedy standardowe podejście jest bardziej naturalne i może być efektywniejsze

Sposoby tworzenia modelu danych

- **Database First** - gdy jest już gotowa baza danych, można istniejące w bazie tabele, widoki oraz procedury składowane wykorzystać do utworzenia odpowiadających im klas
- **Model First** - za pomocą tego podejścia nie trzeba pisać żadnego kodu SQL - wystarczy stworzyć model danych w odpowiednim edytorze graficznym i na jego podstawie tworzona jest struktura bazy danych
- **Code First** - najpierw pisze się kod biznesowy (klasy oraz powiązania między nimi), na podstawie którego tworzony jest model danych oraz generowana struktura bazy danych



Materiały zostały opracowane w ramach projektu
„Zintegrowany Program Rozwoju Politechniki Lubelskiej – część druga”,
umowa nr **POWR.03.05.00-00-Z060/18-00**
w ramach Programu Operacyjnego Wiedza Edukacja Rozwój 2014-2020
współfinansowanego ze środków Europejskiego Funduszu Społecznego

