

PHP - wprowadzenie

Beata Pańczyk - PHP (Wykład 1)

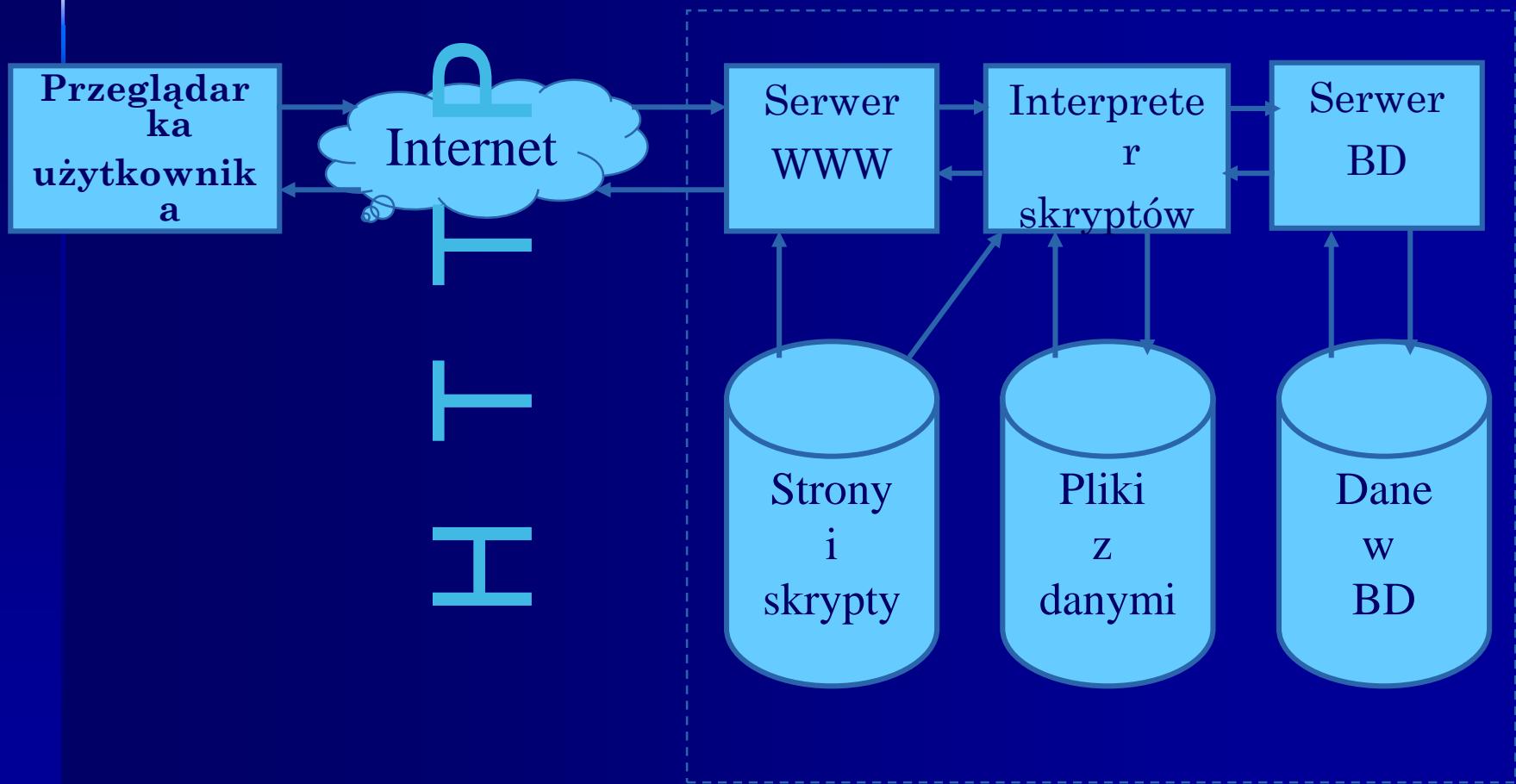
Plan wykładu

- Środowisko aplikacji internetowej
- Protokół HTTP
- Co potrafi PHP?
- Co jest potrzebne do pracy z PHP
- Elementy konfiguracji (plik php.ini)
- PHP na stronach HTML
- Składnia języka
- Typy danych
- Praca z formularzami
- Funkcje pomocnicze

Literatura

- <http://www.php.net/manual/en/index.php>
<http://www.php.net/manual/pl/index.php>
- <http://www.w3schools.com/php/default.asp>

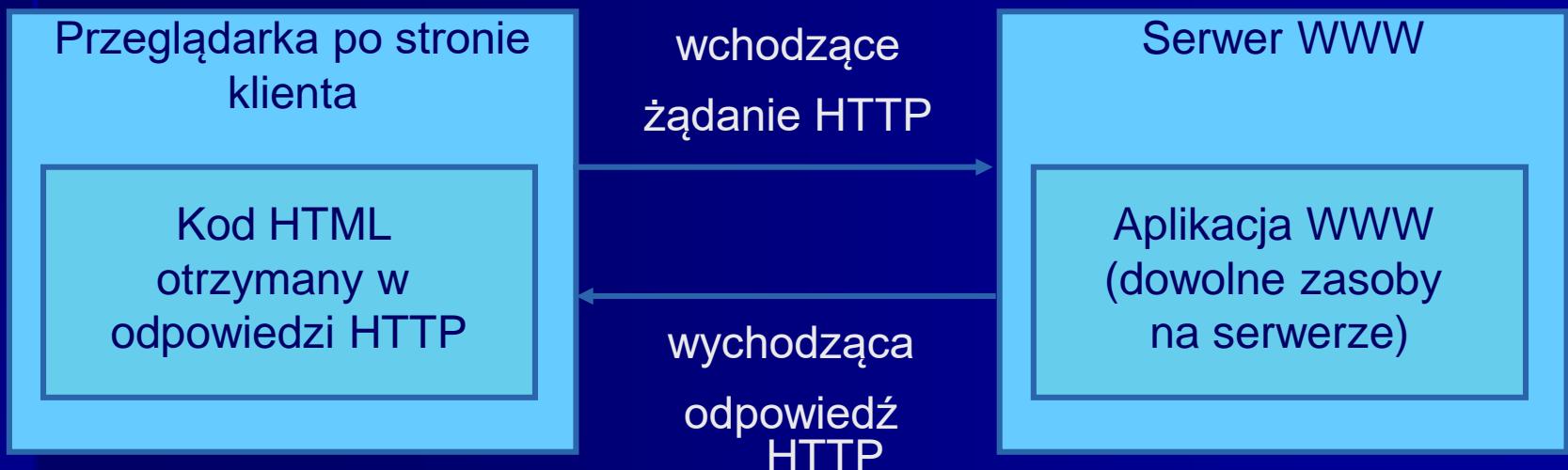
Środowisko aplikacji internetowej



Protokół HTTP

- Służy do transportu dokumentów udostępnianych przez serwery HTTP
- Tekstowy, oparty na protokole TCP, domyślny port 80
- Model klient-serwer, styl żądanie-odpowiedź
- Po dostarczeniu dokumentu połączenie jest zamykane (protokół bezstanowy)

Cykl żądania i odpowiedzi HTTP



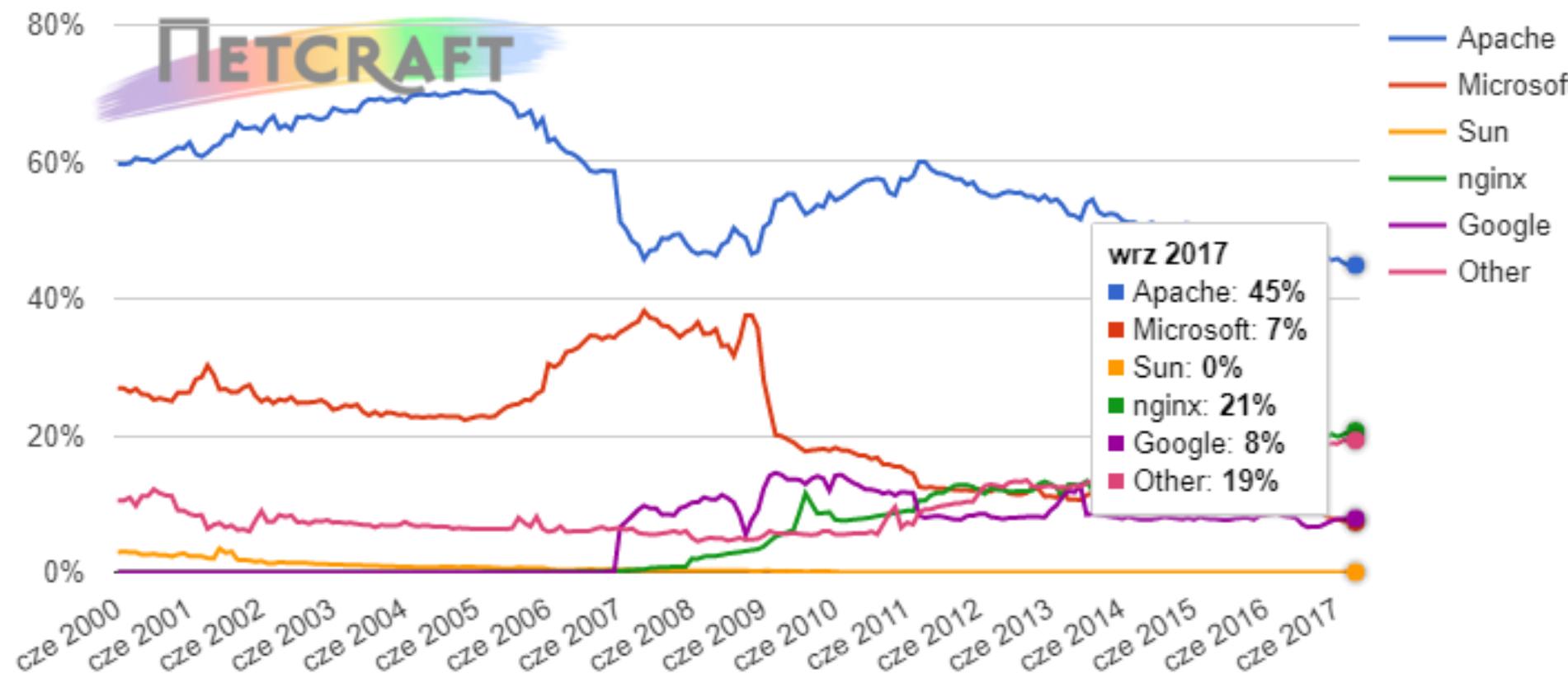
Serwer WWW

- Serwer WWW (Web server) to program, którego zadaniem jest obsługa aplikacji WWW i inne usługi (np. zapewnienie bezpieczeństwa, obsługa protokołu FTP, usługi pocztowe itp.)
- Najbardziej popularne serwery WWW:
 - Apache
 - IIS
 - Serwery aplikacji Java EE: Apache Tomcat, GlassFish

Najpopularniejsze serwery HTTP

<http://news.netcraft.com> (IX 2017)

Web server developers: Market share of active sites



Protokół HTTP

- Łączność HTTP - forma wymiany **komunikatów**
- Formaty zapisu **żądania** HTTP i **odpowiedzi** HTTP są podobne, zawierają zwykle:
 - wiersz początkowy (np. adres URL dokumentu)
 - zero lub wiele wierszy nagłówkowych
 - wolny wiersz
 - opcjonalne ciało komunikatu (np. treść dokumentu)
- Wiersze nagłówkowe zawierają metadane opisujące żądanie HTTP lub odpowiedź HTTP

Kody stanu i nagłówki HTTP

- Stan żądania jest równoważny stanowi HTTP pliku, który został zażądzany
- Kody stanu HTTP reprezentują odpowiedź serwera zależną od statusu żądanego pliku
- Dla żądań HTTP dostępnych jest pięć kodów stanu:
 - Informacyjny: **1xx**
 - Powodzenie: **2xx**
 - Przekierowanie: **3xx**
 - Błąd klienta: **4xx**
 - Błąd serwera: **5xx**

Wybrane kody odpowiedzi serwera HTTP

Kod	Znaczenie
200	Żądanie zakończone sukcesem
202	Żądanie zaakceptowane do przetwarzania asynchronicznego
400	Nieprawidłowe żądanie
403	Dostęp zabroniony
404	Brak żądanego dokumentu (pliku)
500	Wewnętrzny błąd serwera

Komunikaty HTTP

Pola nagłówkowe żądania

HTTP Headers

http://cs.pollub.pl/fax.jpg

GET /fax.jpg HTTP/1.1

Host: cs.pollub.pl

User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; pl; rv:1.9.2.13) Ge

Accept: image/png,image/*;q=0.8,*/*;q=0.5

Accept-Language: pl,en-us;q=0.7,en;q=0.3

Accept-Encoding: gzip,deflate

Accept-Charset: ISO-8859-2,utf-8;q=0.7,*;q=0.7

Keep-Alive: 115

Komunikaty HTTP

Pola nagłówkowe odpowiedzi

HTTP Headers

HTTP/1.1 200 OK

Date: Tue, 01 Feb 2011 19:53:45 GMT

Server: Apache

Last-Modified: Thu, 14 Oct 2004 09:58:31 GMT

Etag: "3c18a-369f-5a49bfc0"

Accept-Ranges: bytes

Content-Length: 13983

Keep-Alive: timeout=15, max=100

Connection: Keep-Alive

Content-Type: image/jpeg

Formaty plików MIME

- MIME określa sposób opisu formatów plików
- Content-Type:
 - dokumenty tekstowe: text/* (css, html, plain)
 - pliki dźwiękowe: audio/* (midi, mpeg, x-wav)
 - pliki graficzne: image/* (gif, jpeg, png)
 - pliki wideo: video/* (mpeg, quicktime)
 - pliki obsługiwane przez dedykowaną aplikację: application/* (pdf, zip)

[Headers](#)[Generator](#)[Config](#)[About](#)

HTTP Headers

http://cs.pollub.pl/

GET / HTTP/1.1

Host: cs.pollub.pl

User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; pl; rv:1.9.2.1...

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q...

Accept-Language: pl,en-us;q=0.7,en;q=0.3

Accept-Encoding: gzip,deflate



[Save All...](#)

[Replay...](#)

Capture

[Clear](#)

[Close](#)

Rozkaz GET i POST

- GET - reprezentuje żądanie klienta HTTP
- POST - również reprezentuje żądanie klienta HTTP, lecz do żądania tego dołączone jest ciało, które reprezentuje dane wysyłane przez klienta HTTP do serwera HTTP (np. parametry, pliki); w związku z obecnością ciała żądania, komunikat sieciowy zawiera pola nagłówkowe „Content-Type” i „Content-Length”, a najpopularniejszym formatem przekazywanej treści jest: **Content-Type: application/x-www-form-urlencoded** (dane formularza HTML przekształcone do postaci ASCII metodą „URL-encoding”)

Formularze HTML – podstawa interfejsu użytkownika

```
<form action="index3.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="Wyślij" /><br />
</form>
```

Przesyłanie danych z formularzy

Metoda GET

`http://localhost/studenci/index3.php?nazwisko=Nowak
&srednia=4.76&wyslij=Wyslij`

The screenshot shows a window titled "Przykład formularza". Inside the window, there are two text input fields. The first field contains "Nazwisko:" followed by "Nowak". The second field contains "Średnia:" followed by "4.76". Below these fields are two buttons: "Wyczyść formularz" and "Wyslij". The "Wyslij" button is highlighted with a blue border.

Metoda GET w żądaniu HTTP

HTTP Headers

http://localhost/studenci/index3.php?nazwisko=Nowak&srednia=4.76&wyslij=Wyslij

GET /studenci/index3.php?nazwisko=Nowak&srednia=4.76&wyslij=Wyslij HTTP/1.1

Host: localhost

User-Agent: Mozilla/5.0 (Windows; U; Windows NT 5.1; pl; rv:1.9.2.13) Gecko/2010...

Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8

Accept-Language: pl,en-us;q=0.7,en;q=0.3

Accept-Encoding: gzip,deflate

Accept-Charset: ISO-8859-2,utf-8;q=0.7,*;q=0.7

Metoda POST w żądaniu HTTP

HTTP Headers

http://localhost/studenci/index3.php

POST /studenci/index3.php HTTP/1.1

Host: localhost

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

Content-Length: 41

nazwisko=Nowak&srednia=4.76&wyslij=Wyslij

Co to jest PHP

- **PHP ("PHP: Hypertext Preprocesor")** jest to język skryptowy działający po stronie serwera
- Skrypt PHP jest niezależny od systemu operacyjnego i od rodzaju przeglądarki, zależy jedynie od oprogramowania serwera
- 1994 – twórca PHP - Rasmus Lerdorf - użył tego języka na swojej stronie domowej (niepublikowana wersji PHP - zbieranie informacji o osobach odwiedzających witrynę, możliwość umieszczania pytań SQL na stronach WWW)

Co potrafi PHP?

- wolność wyboru systemu operacyjnego i serwera WWW
- programowanie proceduralne lub obiektowe
- tworzenie obrazów, plików PDF, animacji Flash
- możliwość wyprowadzania na wyjście dowolnych danych tekstowych (na przykład XHTML, XML) - PHP może automatycznie generować te pliki i zapisywać je w systemie plików
- Szyfrowanie danych
- Programowanie sieciowe (gniazda, tworzenie serwera, praca z protokołem HTTP, FTP, itp.)

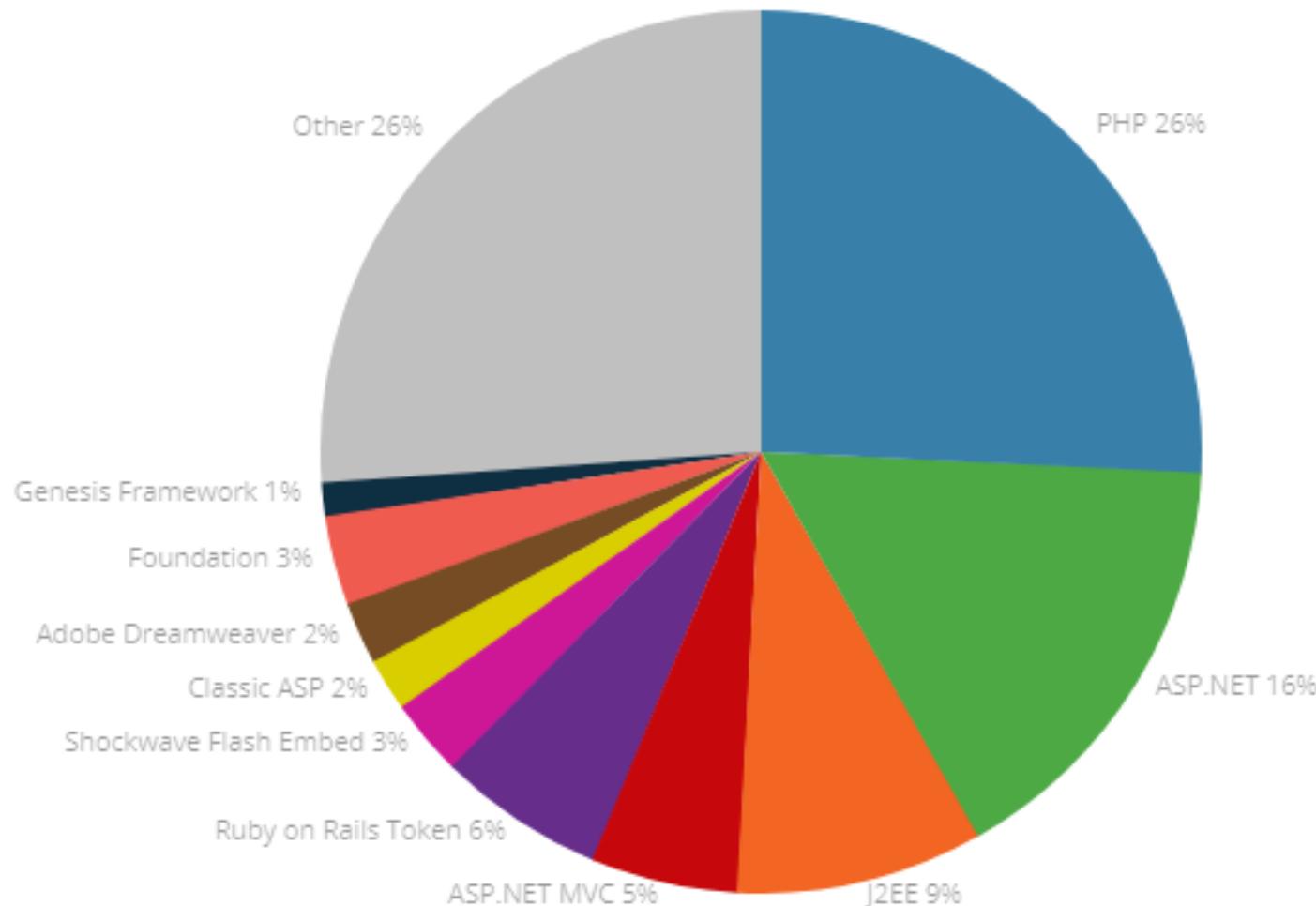
Co potrafi PHP?

- obsługa wielu rodzajów baz danych poprzez funkcje ukierunkowane na konkretną bazę (np. Oracle, PostgreSQL, mSQL, MySQL itp.)
- Możliwość wykorzystania warstwy abstrakcyjnej PDO
- Obsługa standardu Open Database Connection (rozszerzenie ODBC)

<http://trends.builtwith.com/framework>

IX 2017 (PHP – 26%, ASP.NET – 16%, JEE - 9%, ASP.NET MVC – 5%)

Statistics for websites using Framework technologies



Popularność języków programowania

<https://www.tiobe.com/tiobe-index/> (IX 2017)

Sep 2017	Sep 2016	Change	Programming Language	Ratings	Change
1			Java	12.687%	-5.55%
2			C	7.382%	-3.57%
3			C++	5.565%	-1.09%
4			C#	4.779%	-0.71%
5			Python	2.983%	-1.32%
7		▲	PHP	2.210%	-0.64%
6		▼	JavaScript	2.017%	-0.91%
9		▲	Visual Basic .NET	1.982%	-0.36%
10		▲	Perl	1.952%	-0.38%
12		▲	Ruby	1.933%	-0.03%

Co jest potrzebne?

- Do pisania skryptów po stronie serwera (najbardziej tradycyjne i główne pole działania PHP) potrzebne są: PHP, serwer WWW i przeglądarka internetowa
- Kod źródłowy PHP - na stronie <http://www.php.net/downloads.php>
- Szczegóły instalacji dla poszczególnych systemów operacyjnych i serwerów można znaleźć w podręczniku:
<http://www.php.net/manual/pl/install.php>

Apache i PHP w XAMPP

W celu uruchomienia skryptu PHP należy:

- uruchomić serwer Apache (plik konfiguracyjny zwykle **httpd.conf**, zawiera ustawienia np.:
ServerRoot "C:/Program Files/xampp/apache"
DocumentRoot "C:/Program Files/xampp/htdocs"
Listen 80)
- umieścić skrypt w folderze **htdocs** w instalacji Apache'a (może to być również inny folder np. **www**)
- wskazać skrypt przeglądarce jako (localhost:80)
localhost/nazwa_skryptu.php lub
127.0.0.1/nazwa_skryptu.php

Konfiguracja PHP – plik php.ini

- short_open_tag = On (możliwość używania krótkich znaczników)
- register_globals = Off (rejestracja zmiennych globalnych)
- file_uploads = On (możliwość wysyłania plików na serwer)
- upload_max_filesize = 16M (wielkość wysyłanego pliku)
- upload_tmp_dir = "C:\Program Files\xampp\tmp" (ścieżka zapisu przesyłanych plików)
- error_reporting = E_ALL (sposób raportowania błędów)

PHP na stronach HTML

PHP - zagnieżdżony w HTML język skryptowy.
Wszystkie części skryptu PHP muszą być zamknięte
znacznikami początku i końca PHP

Style znaczników PHP:

- styl XML (stosowany w XHTML) : <? php ... ?>
- styl krótki: <? ... ?>
- styl SCRIPT: <script language='php'> ... </script>
- styl ASP: <% ... %>

PHP - zaczynamy

- 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

Przykład 1 - PHP na stronach HTML

Kod źródłowy - rozszerzenie **.php**

```
<html><head><title> Pierwszy skrypt PHP</title></head>
<body>
    <p>PHP oznacza:</p>
    <?php      //początek kodu PHP
print("<h2>PHP Hypertext Preprocessor\n</h2>");
                    //wydruk
?>    <!-- koniec kodu PHP -->
    <p>Aktualna wersja: PHP5 </p>
</body> </html>
```

Przykład 1 - PHP na stronach HTML

Kod źródłowy
widoczny
w przeglądarce:
<html><head>

```
<title>  
Pierwszy skrypt PHP  
</title>  
</head>  
<body> <p>PHP oznacza:</p>  
<h2>PHP Hypertext Preprocessor </h2>  
<p> Aktualna wersja: PHP5 </p> </body> </html>
```



Identyfikatory, zmienne

Identyfikatory:

- dowolna długość (litery, cyfry, kreski podkreślenia, znaki dolara)
- nie mogą rozpoczynać się cyfrą
- rozróżniana wielkość liter

Zmienne:

- przekazane z formularza (umieszczane w odpowiednich tablicach \$_GET, \$_POST)
- zadeklarowane przez użytkownika (nie konieczna deklaracja typu zmiennej przed użyciem; zmienna jest tworzona po pierwszym przypisaniu jej wartości)

Typy zmiennych, stałe

- Typy danych w PHP:
 - **Integer, Double, String, Array, Object**
np. **\$wartosc=1.3; \$wartosc="PHP";**
- Rzutowanie typu:
`$ilosc=0; $wartosc=(double)$ilosc;`
- Zmienne zmiennych - użycie wartości jednej zmiennej jako nazwy drugiej:
`$nazwa="ilosc"; $$nazwa=15; // $ilosc=15;`
- Stałe (oznaczane umownie dużymi literami):
`define("CENA", 100); echo CENA;`

Zasięg zmiennych

- zmienne globalne zadeklarowane w skrypcie - widoczne w całym skrypcie, ale nie wewnątrz funkcji!
- zmienne używane w obrębie funkcji są ograniczone do tej funkcji (zmienne lokalne)
- zmienne używane w obrębie funkcji, które są deklarowane jako globalne, odnoszą się do zmiennej globalnej o tej samej nazwie

Operatory

- operatory arytmetyczne: **+, -, *, /, %**
- operator łączenia ciągów: **.**
np. `$a="PHP"; $b="-operatory"; $c=$a.$b;`
`// $c=$a.-operatory; lub $c="$a-operatory";`
- operatory przypisania: **=, +=, -=, *=, /=, %=, .=**
- operatory pre- i postinkrementacji: **++, --**
- operator referencji: **&**
np. `$a=5; $b=$a; $a=10; //a=10, b=5`
`$a=5; $b=&$a; $a=10; //a=10, b=10`
- operatory porównań: **= =, !=, <>, <, >, <=, >=,**
= = = (operator identyczności - zwraca true, jeśli
operandy są równe i są tego samego typu)

Operatory

- operatory logiczne: **!**, **&&**, **||**, **and** (jak **&&** ale mniejszy priorytet), **or** (jak **||** ale mniejszy priorytet) , **xor**
(\$a xor \$b zwraca true jeśli \$a lub \$b wynosi true, ale nie jednocześnie)
- inne operatory: **,** (przecinek - oddzielenie argumentów funkcji, list składników), **new** (tworzenie obiektów klas), **->** (dostęp do składowych klasy), **[]** (operator tablic), **?:**, **@** (operator tłumienia błędów - np. `$a=@(5/0);` bez operatora @ wygenerowane ostrzeżenie o dzieleniu przez 0, z operatorem @ błąd zostanie stłumiony - należy zastosować kod obsługujący błędy;)

Priorytety operatorów

- ()
- new
- []
- ! ++ -- (int) (double) (string) (array) (object) @
- * / %
- + - .
- < <= > >=
- == != ===
- &
- &&
- ||
- ?:
- przypisania
- and xor or

Instrukcje warunkowe

- if (warunek) { instrukcje; }
- else { instrukcje; }
- if (warunek) { instrukcje; }
- elseif (warunek) { instrukcje; }
- elseif (warunek) { instrukcje; }
- ...
- switch (wartość){ case "wartość1": instrukcje; break;
- case "wartość2": instrukcje; break;
- ...
- default: instrucje;
- }

Instrukcje iteracyjne

- `while (warunek)`
 { instrukcje;
 }
- `for (wyrażenie1; warunek; wyrażenie2)`
 { instrukcje;
 }
- `do`
 { instrukcje;
 }
 `while (warunek);`
- `foreach` – dla tablic asocjacyjnych

Pozostałe instrukcje

- break - przerywa wykonywania instrukcji iteracyjnych lub instrukcji switch
- continue - dla instrukcji iteracyjnych przerywa bieżącą i zaczyna kolejną iterację
- exit - zakończenie wykonywania całego skryptu np.

```
if ($a == 0)
{
    echo "Komunikat o błędzie<br>";
    exit;
}
```

Przykład 2 - skrypt PHP do generowania tabeli

```
<table border="1"><tbody>
<tr>
    <td> Odległość w km </td>
    <td> Koszt w PLN</td></tr>
<?php $d=50;
while($d<=250)
{
    echo "<tr><td> $d </td>";
    echo "<td>".($d/10)."  
" </td>";
    echo "</tr>";
    $d+=50;
}
?>
</tbody></table>
```

Odległość w km	Koszt w PLN
50	5
100	10
150	15
200	20
250	25

Przetwarzanie danych z formularza

- Formularz - dokument HTML
- Parametry formularza:
`<form method="get" action="http://localhost/wyniki.php">` informują, że wynik formularza ma być wysłany do skryptu np. **wyniki.php**
- **wyniki.php** - skrypt PHP przetwarzający informacje otrzymane z formularza HTML
- dane wpisane w polach formularza są pamiętane w specjalnych tablicach asocjacyjnych **`$_GET`** i **`$_POST`** a indeksy, pod którymi są dostępne to nazwy pól formularza (name)
- od wersji PHP 4.1 istnieje również tablica **`$_REQUEST`**, która z definicji zawiera elementy tablic **`$_GET`**, **`$_POST`**⁴³ i **`$_COOKIE`**

Przykład 3 - form1.htm

Dane osobowe:

Nazwisko: Kowalski

Imię: Jan

Adres e-mail: Jank@o2.pl

Proszę zaznaczyć zamawiane produkty:

turbo pascal c++ java

wyslij zamówienie

anuluj zamówienie

Przykład 3 - form1.html

```
<body>
  <h4>Dane osobowe:</h4>
  <form method="get" action="/podstawy/form1_wyniki.php"><p>
    Nazwisko: <input name="nazw" size="30"/><br/>
    Imię: <input name="imie" size="30"/><br/>
    Adres e-mail:<input name="email" size="30"/></p>
    <h4>Proszę zaznaczyć zamawiane produkty:</h4><p>
    <input name="tp" type="checkbox"/>turbo pascal
    <input name="c" type="checkbox"/>c++
    <input name="java" type="checkbox"/>java <br/>
    <input type="submit" value="wyslij zamówienie"/>
    <input type="reset" value="anuluj zamówienie"/></p>
  </form></body>
```

Przykład 3 - form1_wyniki.php

Poniżej znajdują się dane z wysłanego przez Ciebie formularza:

Nazwisko: Kowalski

Imię: Jan

Adres e-mail: Jank@o2.pl

Zamawiane produkty:

- C++
- Java

Przykład 3 - form1_wyniki.php

```
<body>
<h4>Poniżej znajdują się dane z wysłanego przez Ciebie 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['tp']) ) print("- Turbo Pascal<br />");
      if ( isset($_GET['c']) ) print("- C++<br />");
      if ( isset($_GET['java']) ) print("- Java<br />");
?
</body>
```

Metoda GET formularza

- Adres URL z poprzedniego slajdu ma postać:
localhost/form1_wyniki?nazw=Kowalski&imie=Jan&...&C=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 informacja z formularza jest przesyłana metodą **GET**, skrypt PHP, czytając adres URL, automatycznie tworzy zmienne z odpowiednimi wartościami
- Formularze używające metody **POST** nie drukują nazw i wartości zmiennych w adresie URL

Przykład 4 - Przekazywanie danych przez formularze i łącza

- **form2.html** - formularz, który pobiera dane od użytkownika i generuje łącza zawierające informacje, które zgromadził skrypt.
- skrypt **form2_linki.php** - pobiera dane z formularza i tworzy trzy łącza, które następnie są wykorzystane w celu przesyłania informacji odpowiednio do skryptów **form2_kon.php**, **form2_firm.php**, **form2_hobby.php**.

Przykład 4 - form2.html

Informacje kontaktowe:

Nazwisko: Kowalski

Imię: Jan

Telefon: 55555555

Informacje zawodowe:

Nazwa firmy: ABC

Telefon służbowy: 44444444

Zainteresowania:

narciarstwo brydż szachy

Przykład 4 - Przekazywanie danych przez formularze i łącza - form2.html

```
<body><h4>Informacje kontaktowe:</h4>
<form method="get" action="/podstawy/form2_linki.php"><p>
Nazwisko: <input name="nazw" size="30"/><br/>
Imię: <input name="imie" size="30"/><br/>
Telefon:<input name="telp" size="30"/></p>
<h4>Informacje zawodowe:</h4><p>
Nazwa firmy: <input name="firma" size="30"/><br/>
Telefon służbowy:<input name="tels" size="10"/></p>
<h4>Zainteresowania:</h4><p>
narciarstwo<input type="radio" name="hobby" value="narciarstwo"/>
brydż<input type="radio" name="hobby" value="brydż"/>
szachy<input type="radio" name="hobby" value="szachy"/><br/>
<input type="submit" value="Wyslij"/>
<input type="reset" value="Anuluj"/></p></form>
```

Przykład 4 - form2_linki.php

Łącza utworzone na podstawie danych użytkownika:

[Informacje kontaktowe](#)

[Informacje służbowe](#)

[Ulubione hobby](#)

Przykład 4 - form2_linki.php

```
<h4>Łącza utworzone na podstawie danych użytkownika:</h4>
<?php
    $nazw=$_GET['nazw']; $imie=$_GET['imie']; $telp=$_GET['telp'];
    $firma=$_GET['firma']; $tels=$_GET['tels']; $hobby=$_GET['hobby'];
    print("<p>
        <a href=\"form2_kon.php?nazw=$nazw&imie=$imie&telp=$telp\">";
        /*utworzenie łącza do form2_kon.php z równoczesnym przekazaniem
        wartości zmiennych nazw, imie, telp pobranych z formularza form2.html */
    print("Informacje kontaktowe</a>");
    print("<p><a href=\"form2_firm.php?firma=$firma&tels=$tels\">";
    print("Informacje służbowe</a>"); //utworzenie łącza do form2_firm.php
    print("<p><a href=\"form2_hobby.php?hobby=$hobby\">");
    print("Ulubione hobby</a>"); //utworzenie łącza do form2_hobby.php
?>
```

Przykład 4 - form2_kon.php

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

Informacje kontaktowe

Imię: Jan

Nazwisko: Kowalski

Telefon domowy: 55555555

Przykład 5 – Prosty formularz

```
<?php //drukuje formularz i jednocześnie pobiera
      //i wyświetla wpisane w nim dane
if ($_GET['tekst']) //jest wpisana wartość w formularzu
{
    $tekst=$_GET['tekst'];
    print "Wpisano: $tekst <br/>";
    print '<a href="form3.php"> Powrót do formularza</a>';
}
else //nie ma wpisanych danych - wyświetl formularz
{
    print 'Podaj tekst :<form method="get"
                      action="/podstawy/form3.php"> ';
    print '<input type="tekst" name="tekst"/>';
    print '<input type="submit" value="Wyslij"/>';
    print '</form>';
}
?>
```

Przykład 5 – wynik działania 1

Podaj tekst :

Wpisano: PHP5
[Powrót do formularza](#)

Przykład 5 – wynik działania 2

Podaj tekst :

<h1>PHP5</h1>

Wpisano:

PHP5

[Powrót do formularza](#)

Przykład 5 – wynik działania 3

Podaj tekst :



Przykład 5a – bezpieczne wyświetlanie danych

- `htmlspecialchars` – zamienia znaczniki HTML na kody „bezpieczne” do wyświetlania na stronach WWW
- W przykładzie 5 należy wprowadzić jedną modyfikację:
`$tekst = htmlspecialchars($_GET['tekst']);`

Podaj tekst :

<h1>PHP5</h1>

[Wyslij](#)

Wpisano: <h1>PHP5</h1>

[Powrót do formularza](#)

Usuwanie pustych przestrzeni w 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
- **ltrim()** - j.w. ale tylko z początku ciągu
- **chop()** - j.w. ale tylko z końca ciągu
np.
`$nazwa= " a\nb\tc ";
$nazwa = trim($nazwa);`

Wyświetlanie sformatowanych ciągów

- **nl2br()** - pobiera串 i zamienia w nim wszystkie znaki końca linii na znacznik HTML

- **echo** - konstrukcja wyświetlająca串 w przeglądarce
- **print()** - działa jak echo, ale zwraca wartość 0 lub 1
- **printf()** - wyświetla sformatowany串 w przeglądarce; odpowiednik funkcji z języka C
- np.
`echo "Wartość zamówienia wynosi $wartosc";`
`printf("Wartość zamówienia wynosi %s", $wartosc);`

Zmiana wielkości liter w ciągu

- **strtoupper()** - zamiana ciągu na wielkie litery
- **strtolower()** - zamiana ciągu na małe litery
- **ucfirst()** - zamiana na wielką pierwszej litery ciągu, jeżeli jest ona literą
- **ucwords()** - zamiana na wielką pierwszej litery każdego wyrazu ciągu, jeżeli jest ona literą np.

```
$tekst="Język PHP";  
strtoupper($tekst);
```

Formatowanie ciągów do przechowania w bazie danych

- Znaki " (cudzysłów podwójny) , ' (cudzysłów pojedynczy), \ (lewy ukośnik), NULL - baza może interpretować jako znaki kontrolne; aby tego uniknąć umieszcza się przed nimi lewy ukośnik
- **addslashes()** - dodaje znaki ukośnika przed każdym powyższym znakiem; zwraca串 przefomatowany do przechowania w bazie danych
- **stripslashes()** - usunięcie ukośników w串 pobranym z bazy danych
np. \$tekst=sddslashes(\$tekst);
\$tekst=stripslashes(\$tekst);
- **strip_tags** – zwraca łańcuch bez znaczników HTML i PHP (poza wskazanymi w drugim parametrze metody)

Przykład - funkcje addslashes, stripslashes

```
<?php  
    $tekst='Cytat:"cytat" z pliku C:\wykłady.pdf';  
    echo $tekst."<br/>";  
    $tekst=addslashes($tekst); echo $tekst."<br />";  
    $tekst=stripslashes($tekst); echo $tekst."<br />";;  
?  
?
```

Cytat:"cytat" z pliku C:\wykłady.pdf
Cytat:\"cytat\" z pliku C:\\wykłady.pdf
Cytat:"cytat" z pliku C:\\wykłady.pdf

Pobieranie danych z pól formularzy do zapisu w bazie danych

- addslashes(trim(\$_POST['imie']))
- addslashes(strip_tags(trim(\$_POST['imie'])))