Technologie internetowe

CZĘŚĆ 4

PHP
Skryptowy język aplikacji WWW

Wykład opracowany na podstawie materiałów z:
PHP i MySQL. Tworzenie stron internetowych. Helion
http://pl.wikipedia.org/
http://www.kurshtml.edu.pl/
http://pl.docs.pld-linux.org/

Dr inż. Arkadiusz Rzucidło

Co to jest PHP?

- **PHP** (Hypertext Preprocesor) jest językiem skryptowym wykonywanym po stronie serwera (server-side)
  PHP jest składnikiem wielu średnich i dużych bazodanowych aplikacji WWW
- **Historia**
  - **1994** – Rasmus Lerdorf stworzył zbiór narzędzi do obsługi swojej strony domowej – mechanizm interpretacji zestawu makropoleceń; np.: książka gości, licznik odwiedzin (PHP – Personal Home Pages). System uzupełniał pakiet interpretujący formularze (FI).
  - **1997** – ok. 50000 witryn używa PHP/FI, projekt grupowy
  - **2003** – miliony serwerów o dowolnej konfiguracji korzysta z PHP4
  - ciągły rozwój języka (PHP5 = wersja obiektowa)

Cechy PHP?

- O popularności PHP decydują:
  - Oprogramowanie o otwartym źródle !
  - Skrypty PHP osadzane w plikach HTML (integracja z warstwą klienta)
  - Posiada ponad 15 bibliotek pozwalających na szybki dostęp do baz danych (ODBC, MySQL, Oracle i inne…)
  - Szybko wykonuje skrypty (wszystkie komponenty PHP działają w głównej przestrzeni pamięciowej)
  - Swobodny wybór platformy sprzętowej i systemu operacyjnego
  - Jest w pełni funkcjonalnym językiem programowania.
  - Nadaje się do tworzenia złożonych systemów

Funkcjonalność PHP

- PHP manual: "Celem języka jest umożliwienie programistom WWW szybkiego pisania stron generowanych dynamicznie„
- Najczęstsze zastosowania PHP:
  - Wykonywanie funkcji systemu (obsługa systemu plików, wykonywanie poleceń systemowych)
  - Zbieranie danych z formularzy (zapis do pliku, e-mail, zwrot przetworzonych danych)
  - Uzyskiwanie dostępu do baz danych (przeglądanie, dodawanie i modyfikacja elementów włącznie z ingerencją w metadane)
  - Tworzenie cookies, i uzyskiwanie dostępu do zmiennych lokalnych i globalnych
  - Rozpoczynanie sesji i używanie zmiennych i obiektów sesji
  - Wykorzystanie uwierzytelniania użytkownika PHP do ograniczenia dostępu do pewnych sekcji witryny WWW a także zasobów plikowych serwera
  - Tworzenie obrazów na bieżąco
  - Kodowanie danych

Przykłady zastosowania PHP

- Sklepy internetowe, aukcje, przetargi itp.
- Dynamiczne serwisy tematyczne, newslettery
- Obsługa poczty elektronicznej
- Obsługa zarządzania usługami serwera internetowego
- Forum dyskusyjne, mechanizm zamieszczania newsów
- Mechanizmy przeszukiwania serwisu, statystyki serwisu
- Serwer gier on-line (np. szachy, reversi)
- Liczba zastosowań jest otwarta ze względu na duża funkcjonalność języka

Jak interpretowany jest skrypt PHP?

- Skrypt PHP jest "mieszanką" kodu HTML i PHP czasami jeszcze JavaScript
- Plik ma rozszerzenie .php (lub .php3, phtml)
- Interpretacja:
  - Przeglądarka WWW żąda dokumentu .php posługując się ścieżką dostępu do pliku zawartą w adresie lub parametrami dostępu do pliku
  - Serwer WWW kojarzy plik z parserem (analizatorem składni) PHP i przesyła mu plik do analizy
  - Parser PHP przegląda plik w poszukiwaniu kodu PHP i sprawdza poprawność składni
  - Kompilator PHP wykonuje znaleziony kod, uzyskane wyniki umieszcza w miejscu uprzednio zajmowanym przez kod PHP
  - Nowy – wygenerowany plik wynikowy jest przesyłany z kompilatora do serwera WWW jako czysty HTML
  - Serwer wysyła plik do przeglądarki klienta również jako kod HTML
  - Przeglądarka wyświetla HTML w oknie użytkownika
  - **Uwaga:**
    - **przeglądarka nie ma dostępu do kodu źródłowego PHP, "widzi tylko HTML jako wynik skryptu**
    - **Przeglądarka sama w sobie nie potrafi zinterpretować składni PHP i zazwyczaj jeśli próbujemy uruchomić plik php lokalnie to przeglądarka wyświetla go jako zwykły plik tekstowy prezentując zawartość w oknie.**

Składnia PHP

- Znaczniki informujące o kodzie skryptu

Znacznik otwierający	Znacznik zamykający
<?php	?>
<script language= "php">	</script>

- Odstępy (puste linijki, tabulatory – kaskadowość) w skryptach PHP nie mają znaczenia, zwiększają jedynie czytelność
- Instrukcje mogą być przeplecione dowolną mieszanką spacji, tabulatorów, powrotów karetki, itp.
- Skrypt PHP jest ciągiem instrukcji, z których **każda kończy się średnikiem**
- Skrypt może się znajdować w dowolnym miejscu pliku,
- W pliku może znajdować się dowolną liczbą skryptów

Składnia PHP

- Komentowanie w skrypcie

```
<! - - to jest komentarz html  - - >
//  to jest komentarz jednowierszowy
#  to jest komentarz wielbicieli powłok (jednowierszowy)
/* tak można tworzyć komentarze wielowierszowe – takie na przykład jakie się
   spotyka w Javie */
```

- Prezentacja danych

```
echo "Witaj, świecie";
print "Witaj, świecie";
echo 123;
echo $zmienna;
echo "Witaj, " , "świecie";
//print i echo można umieszczać w nawiasach:
print ("Witaj");
echo "To działa";
echo ' tak samo jak to';
echo "Ten łańcuch zawiera ' : pojedynczy cudzysłów" ;
echo 'Ten łańcuch zawiera " : podwójny cudzysłów' ;
```

Zmienne - użycie

- Zmienna stanowi reprezentację określonej wartości ("zielony", "22„)
- Zmienne w PHP identyfikuje znak dolara, po którym następuje nazwa zmiennej (ważne duże i małe litery)
- Zmiennych nie trzeba deklarować i nie mają one typu, dopóki nie przypisze im się wartości
- Ustalenie typu i przypisanie wartości zmiennej:

```
$zmn = 15;
```

- Zmiana typu zmiennej przez przypisanie nowej wartości:

```
$zmn = "i tekst gotowy";
```

- $TWOJAZM ≠ $twojazm (nie te same zmienne !!!)
- Wartości zmiennych można włączyć do literałów:

```
$pojazd = "autobus czerwony";
$stan = 'mknie';
$spiew = "$pojazd ulicami mego miasta $stan";
$ile = 45;
$info = "$pojazd zabiera $ile osób";
```

Zmienne - typy

- PHP posiada cztery typy skalarne:
  - boole'owski, zmiennoprzecinkowy, całkowity i łańcuchowy,
  - oraz typ złożony: tablicowy.
- Zmienne skalarne zawierają w danej chwili tylko jedną wartość
- Tablice mogą zawierać wiele wartości skalarnych lub inne wartości złożone.

```
$tak = true;
$test = false;
$intvar = 65;
$floatvar = 6.5;
$floatvar1 = 1.12e3;
$floatvar2 = 2e-2
$suma = $intvar + $floatvar;
$ciekawe = $tak + $intvar
```

Stałe

- Stałe wiążą nazwę z prostą wartością skalarną (np. *true* i *false* są stałymi skojarzonymi z wartościami 1 i 0)
- Stałych nie poprzedza się znakiem $, nie można ich zmienić po zdefiniowaniu, można z nich korzystać w dowolnym miejscu skryptu (po definicji), mogą przyjmować wartości skalarne.

```
define( "pi" , 3.14159);
$alfa = pi * pi;
echo $alfa;
```

Operatory w PHP

- Zrobienie czegoś z wartością zmiennej (przypisanie wartości, zmiana wartości, porównanie kilku wartości)

Główne typy operatorów:

- **Operatory przypisania:** przypisują wartość do zmiennej mogą również dodawać do bieżącej wartości zmiennej lub od niej odejmować
- **Operatory arytmetyczne:** służą do dodawania, odejmowania, dzielenia i mnożenia
- **Operatory porównania:** porównują dwie wartości i zwracają prawdę lub fałsz; na podstawie zwróconej wartości można wykonywać dalsze działania;
- **Operatory logiczne:** określają status warunków

Operatory przypisania

Operator	Przykład	Czynność
=	$a = 8;	Przypisuje zmiennej wartość po prawej
+=	$a += 3;	Dodaje do zmiennej wartość po prawej
-=	$a -= 5;	Odejmuje od zmiennej wartość po prawej
.=	$a .= "drzew";	Łączy wartość po prawej z bieżącą

Operatory arytmetyczne

Operator	Przykład	Czynność
+	$b = $a + 3;	Dodaje wartości
-	$b = $a - 3;	Odejmuje wartości
*	$b = $a * 3;	Mnoży wartości
/	$b = $a / 3;	Dzieli wartości
%	$b = $a % 3;	Zwraca resztę
.	$t= 'Napis' . $a;	Konkatenacja tekstów (łączenie)

Operatory porównania

Operator	Definicja
==	Równe
!=	Nie równe
>	Większe niż
<	Mniejsze niż
>=	Większe lub równe
<=	Mniejsze lub równe

Operatory logiczne
pozwalają skryptowi określić status warunków (takich jak porównania)

W kontekście instrukcji `if…else` lub `while` operatory logiczne wykonują określony kod według tego, które warunki są prawdziwe, a które fałszywe

- `&&` - oznacza koniunkcję warunków (i)
- `||` - oznacza alternatywę warunków (lub)

Operatory składania

- Operatory te są stosowane podczas jednoczesnego wykonywania operacji arytmetycznych i przypisywania
- Pozwalają na przyspieszenie kodowania, ale mogą uczynić wyrażenia trudniejszymi do odczytania

Operator	Przykład	Równoważnik
++	$a++ ++$a	$a = $a + 1
--	$a -- --$a	$a = $a - 1
+=	$a += $b	$a = $a + $b
-=	$a -= $b	$a = $a - $b
/=	$a /= $b	$a = $a / $b
*=	$a *= $b	$a = $a * $b
.=	$a .= $b	$a = $a . $b
%=	$a%= $b	$a = $a%$b

Postinkrementacja i preinkrementacja

```
echo "Postinkrementacja";
$a = 5;
echo "Powinno być 5: " . $a++ . "<BR>";
echo "Powinno być 6: " . $a   . "<BR>";

echo "Preinkrementacja";
$a = 5;
echo "Powinno być 6: " . ++$a . "<BR>";
echo "Powinno być 6: " . $a   . "<BR>";
```

Instrukcje warunkowe

```
if (warunek)
instrukcje;
```

```
if (warunek)
Instrukcja1;
else
Instrukcja2;
```

```
if (warunek1)
{
instrukcje1;
}
elseif (warunek2)
instrukcja2;
(…)
else
instrukcja3;
```

- Stosuje się gdy chcemy by fragment kodu został wykonany po spełnieniu warunku
- Przydatne uzupełnienie instrukcji warunkowej `if` stanowi funkcja `isset()`, sprawdzająca czy zmienna została ustawiona:

```
if (isset ($submit))

// operacje po naciśnięciu przycisku
   submit;


else

// ponowne wyświetlenie formularza;
```

Pętle - while do

- Pętle – stosuje się aby fragment kodu wykonać wiele razy
- Pętla while do

```
While (warunek)
{
Instrukcje;
}
```

- Przed każdą iteracją bloku kodu wewnątrz instrukcji while, sprawdzany jest warunek
- Jeżeli wyrażenie warunkowe jest prawdziwe - blok kodu umieszczony wewnątrz instrukcji zostanie wykonany
- Jeśli natomiast wyrażenie przyjmie wartość false, pętla zakończy się, a wykonanie skryptu zostanie kontynuowane od pierwszej instrukcji umieszczonej po pętli while
- ***Sekwencja kroków pętli while:***
  1. *Sprawdzenie wyrażenia warunkowego*
  2. *Jeśli wyrażenie = false → krok 5*
  3. *Wykonanie bloku instrukcji*
  4. *Krok 1*
  5. *Zakończenie pętli*

Petla – do while

- Pętla do … while

```
do
{
instrukcje;
}
while (warunek)
```

- Wyrażenie warunkowe sprawdzane jest na końcu pętli – po wykonaniu bloku instrukcji.
- Sekwencja kroków pętli do … while:
  1. *Wykonanie bloku instrukcji*
  2. *Sprawdzenie wyrażenia warunkowego*
  3. *Jeżeli warunek prawdziwy → krok 1*
  4. *Zakończenie pętli, wykonanie kolejnej instrukcji programu*

Pętla -for

- Pętla FOR

```
for (wyr_startowe; wyr_warunkowe; wyr_iteracyjne)
{
instrukcje
}
```

- wyr_startowe – jest wykonane raz gdy program dotrze do instrukcji for
- wyr_warunkowe – sprawdzane przed każdą iteracją pętli, jeśli warunek = false – pętla zostaje przerwana
- wyr_iteracyjne – jest wykonywane po każdej iteracji pętli, służy do modyfikacji zmiennych zdefiniowanych w wyrażeniu warunkowym

```
for ($x = 0, $x<=100; $x++)
```

Uwaga: wyrażenia mogą być dowolnie skomplikowane, można użyć kilku instrukcji startowych, warunków i wyr. iteracyjnych, oddzielając je przecinkami

```
for ($x = 0, $y = -5; $x < 10 && $y < $z; $x++, $y+=3)
```

Pętla - for

- ***Sekwencja kroków pętli for**:*
  ***1.** Wykonanie wyrażenia startowego*
  ***2.** Sprawdzenie wyrażenia warunkowego*
  ***3.** Jeżeli warunek jest fałszywy → krok 7*
  ***4.** Wykonanie bloku instrukcji pętli*
  ***5.** Wykonanie wyrażenia iteracyjnego*
  ***6.** → krok 2*
  ***7.** Zakończenie instrukcji for; wykonanie kolejnej instrukcji programu*

Instrukcje break i continue

- Wykonanie pętli można w każdym momencie zakończyć
- Służy do tego instrukcja break:

```
$a=0;
while($a<10)
{
    $a++;
     echo $a;
    if($a==3)
       break;
}
```

- Ta pętla nie wykona się 10 razy - gdy $a osiągnie wartość 3, wykonanie pętli zostanie przerwane.

Instrukcje break i continue

- Można też przejść do następnego powtórzenia pomijając pozostałe instrukcje ciała pętli - służy do tego instrukcja **continue**

```
$a=0;
while($a<10)
{
   $a++;
   if($a==3)
      continue;
   echo("aaa"); /* ta instrukcja wykona
      się tylko gdy $a nie jest równe 3 */
}
```

Instrukcja switch

- Wygodniejsze jest zastosowanie instrukcji switch:

```
switch($a)
    {
        case 1:
        echo("a jest równe 1");
        break;
        case 3:
        echo("a jest równe 3");
        break;
        case 11:
        echo("a jest równe 11");
        break;
    }
```

Wybrane funkcje matematyczne

Funkcja	Znaczenie
abs (x)	zwraca wartość bezwzględną x
ceil (x)	zwraca wartość x zaokrągloną w górę do najbliższej liczby całkowitej
floor (x)	zwraca wartość x zaokrągloną w dół do najbliższej liczby całkowitej
max (x,y,...)	zwraca największą wartość listy wartości
min (x,y,...)	zwraca najmniejszą wartość listy wartości
pow (x,n)	zwraca liczbę x podniesioną do potęgi n
rand(min, max)	generuje wartość losową z zakresu liczb
sqrt (x)	zwraca pierwiastek kwadratowy

Wybrane funkcje obsługi łańcuchów znakowych

Funkcja	Znaczenie
strlen(s)	Zwraca długość łańcucha znaków
strtoupper(s)	Zamienia znaki łańcucha na duże litery
strtolower(s)	Zamienia znaki łańcucha na małe
chop(s) rtrim(s)	zwraca wartość s usuwając białe znaki z prawej strony.
ltrim(s)	zwraca wartość s usuwając białe znaki z lewej strony
trim(s)	zwraca wartość s usuwając białe znaki z obu stron
str_repeat(s, n)	powiela tekst s, n razy.
strrev(s)	odwraca tekst

Kontrola błędów – znak @

- Jeśli znak ten zostanie postawiony przed dowolnym wyrażeniem w PHP, jakiekolwiek powiadomienia o błędach wygenerowane przez to wyrażenie zostaną pominięte (nie będą wyświetlone)
- Jeśli mechanizm track_errors (php.ini) został włączony, jakiekolwiek powiadomienie o błędzie zostanie zapisane do zmiennej globalnej $php_errormsg
- Zawartość tej zmiennej jest nadpisywana przy każdym błędzie, więc po wystąpieniu kolejnego błędu w skrypcie, informacja o poprzednim błędzie jest tracona
- Operator @ działa tylko na wyrażeniach
- Jeśli da się pobrać wartość czegoś, można postawić operator @ przed tym czymś
- Można postawić @ przed zmiennymi, wywołaniami funkcji, instrukcjami include(), stałymi, itp.
- Nie można stawiać @ przed definicjami funkcji bądź klasy, lub strukturami warunkowymi, takimi jak if lub foreach, itd.

```
$my_file = @file ('nieistniejący_plik') or die ("Błąd przy
otwieraniu pliku: treść błędu: '$php_errormsg'");
```

Składnia alternatywna

- Dotyczy instukcji: if, while, for, foreach i switch
- Podstawowa forma składni alternatywnej polega na zamianie nawiasu otwierającego na dwukropek (:), a nawiasu zamykającego na odpowiednie słowo: endif;, endwhile;, endfor;, endforeach; lub endswitch;.

```
<?php if ($a == 5): ?>
   A jest równe 5
 <?php endif; ?>
```

Netografia

- PHP i MySQL. Tworzenie stron internetowych. Helion
- http://pl.docs.pld-linux.org/
- Kurs HTML http://www.kurshtml.edu.pl
- Za zgodą fragmenty materiałów dydaktycznych dr inż. Tomasza Bajorka
- http://www.a4.pl
- manual na stronie z instrukcjami http://pl2.php.net/FAQ.php
- PHP-owa witryna: PHP-Nuke http://www.phpnuke.org

Pytania sprawdzające

- Czym jest PHP?
- Zalety i funkcjonalność PHP?
- Przykłady zastosowania PHP?
- Jak interpretowane są skrypty?
- Co to jest parser?
- Co czyni plik skryptem PHP?
- Cechy zmiennych?
- Operatory w PHP, przykład?
- Różne postacie instrukcji warunkowej w PHP?
- Pętle w skrypcie, przykłady?
- Inkrementacja i dekrementacja na czym polega?
- Obsługa błędów w PHP?

Koniec

Temat następnego wykładu to:

PHP cd.