Technologie internetowe

CZĘŚĆ 7

PHP
Skryptowy język aplikacji WWW
Ciąg dalszy

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/
http://coogi.cba.pl/

Dr inż. Arkadiusz Rzucidło

PHP - programowanie

- Funkcje
- System plików
- Sesje
- Cookie
- Obsługa formularzy

Funkcje

- Jest blokiem instrukcji posiadającym nazwę oraz opcjonalnie posiadającym parametry wejściowe
- Funkcja zwraca po swoim działaniu wynik
- PHP posiada wbudowane funkcje a także umożliwia tworzenie własnych
- Podstawowe ramy funkcji

```
function pomnoz($czynnik1, $czynnik2)
{
$wynik = $czynnik1 * $czynnik2;
echo $wynik;
}
```

- Wywołanie funkcji w kodzie skryptu

```
pomnoz(4, 5);
```

- Wynikiem funkcji jest wyświetlona na ekranie liczba – w przykładzie: 20
- Aby funkcja zwróciła wartość lecz nie wyświetlała jej na ekranie należy użyć w kodzie:

```
return $wynik;
```

- A wywołanie funkcji to:

```
$wynik=pomnoz(4,5);
```

Funkcje

- Zmienne użyte w funkcjach są niewidoczne dla skryptu – są to zmienne lokalne
- Są przypadki potrzeby wykorzystania zmiennych z zewnątrz funkcji w skrypcie – zmienne globalne
- Zmienne globalne są widoczne przez cały czas działania skryptu
- Dostęp do zmiennych globalnych w funkcji realizuje się za pomocą polecenia *global*

```
$a = 5;

function globalna()
{
global $a;
// import zmiennej globalnej
$b = 2;
$c = $a + $b;
return $c;
}
echo globalna(); // wypisze liczbę 7
```

Obsługa systemu plików

- Wszystkie operacje PHP związane z systemem plików związane są z plikami na serwerze
- PHP nie realizuje bezpośrednich działań związanych z plikami systemu klienta
- Plik musi spełniać wymagania:
  - być własnością użytkownika obsługującego serwer WWW
  - posiadać stosowne prawa do zapisu informacji
- Operacje w systemie plików dotyczą wszystkich operacji, które są wykonywane lokalnie ! UWAGA !
- Funkcje dostępu do plików mogą stanowić potencjalne niebezpieczeństwo
- Sugerowana jest rozwaga w stosowaniu operacji uruchamiania poleceń systemu operacyjnego

Dostęp do pliku

- Do otwierania plików służy funkcja
  fopen`(nazwa_pliku,`*`mode`*`);`
  - **Nazwa** pliku to pełna ścieżka do pliku, który tworzymy lub otwieramy,
  - **mode** określa tryb działania funkcji
  - Funkcja zwraca wskaźnik pliku (służy jako odnośnik i dostarcza informacji o pliku)
- Parametr mode przyjmuje jedną z wartości:

Mode	Opis
r	Otwiera istniejący plik i odczytuje zawarte w nim dane. Ustawia wskaźnik na początku pliku
r+	Otwiera istniejący plik do odczytu i zapisu i ustawia wskaźnik na początku pliku
w	Otwiera plik tylko do zapisu i jeżeli plik nie jest pusty, opróżnia go, a jeżeli taki plik nie istnieje zostanie utworzony
w+	Otwiera plik do odczytu i zapisu. Jeżeli plik nie jest pusty, opróżnia go, a jeżeli taki plik nie istnieje zostanie utworzony
a	Otwiera istniejący plik do zapisu i ustawia wskaźnik na końcu pliku. Jeśli plik nie istnieje f. tworzy go.
a+	Otwiera plik do odczytu i zapisu i ustawia wskaźnik na końcu pliku. Jeśli plik nie istnieje f. tworzy go.

Dostęp do pliku

- Pliki do których PHP ma dostęp mogą być dowolnego typu (mogą mieć dowolne rozszerzenie)
- Funkcja *fopen* zwraca tzw. uchwyt do pliku (ang. file handle), którym w przykładzie poniżej jest zmienna *$plik*

```
$plik = fopen('plik.txt', 'r'); // do odczytu
$plik = fopen('plik.txt', 'w'); // do skasowania całej zawartości
i zapisu
$plik = fopen('plik.txt', 'a'); // do dopisywania nowej treści
$plik = fopen('plik.txt', 'r+'); // do odczytu i zapisu
$plik = fopen('plik.txt', 'w+'); // do skasowania wszystkiego,
zapisu i odczytu
$plik = fopen('plik.txt', 'a+'); // do dopisywania i odczytu
```

Pobieranie danych z pliku

- Odczytanie z pliku jednego znaku

```
$znak = fgetc($plik);
```

  - Funkcja odczytuje *fgetc* odczytuje za każdym razem jeden (kolejny) znak a kiedy dojdzie do końca pliku – zwróci wartość *false*
- Odczytanie z pliku całej linii danych

```
$linia = fgets($plik, 100);
```

  - Funkcja gets odczytuje z pliku określonego w zmiennej $plik określoną liczbowo (opcjonalnie) liczbę znaków
  - Jeśli linia będzie dłuższa funkcja odczyta tylko tyle znaków ile wskazuje wartość
  - Jeśli nie będzie podanej wartości – domyślnie funkcja odczyta 1024 znaki
- Odczytanie całej zawartości pliku

```
$dlugosc = filesize($nazwa_pliku);
$zawartosc = fread($plik, $dlugosc);
```

  - brak argumentu *$dlugosc* skutkuje błędną odpowiedzią funkcji
  - Do odczytania całego pliku wykorzystuje się zatem funkcje zwracającą długość pliku *filesize*

Pobieranie danych z pliku - alternatywa

- Funkcja file tworzy tablicę, której elementami są kolejne wiersze pliku
- Funkcja potrzebuje uchwytu do pliku jako parametru

```
$plik = 'przykladowy.txt';
$linia = file($plik);
echo $linia[1];
```

- Nowsza funkcja PHP

```
echo file_get_contents($plik);
```

- Obie funkcje wprowadzają cały plik do pamięci dlatego nadają się do obsługi małych rozmiarowo plików
- Można unieruchomić w ten sposób serwer zapychając jego pamięć
- Lepiej używać funkcji fgets z dostępem do kolejnych linii

Zapis danych do pliku

- Uchwyt pliku należy zmodyfikować dodając parametr „w”
- Funkcja zapisu ma trzy parametry:
  - uchwyt do pliku,
  - treść do zapisania
  - *opcjonalnie* maksymalna długość wpisu

```
$plik = fopen('plik.txt', 'w');
fwrite($plik, $tresc);
fclose($plik);
```

```
$plik = fopen('plik.txt', 'r');
$tresc = fread($plik, filesize('plik.txt'));
fclose($plik);
```

```
echo $tekst_z_pliku;
```

// rozwiązanie alternatywne, o ile krótsze (tylko w PHP >= 4.3.0)

```
echo file_get_contents('plik.txt');
```

Blokowanie pliku

- Przydatne w przypadkach równoczesnej próby dostępu do pliku np. licznika
- Plik blokuje funkcja *fblock*, Potrzebuje dwóch argumentów: zmiennej z plikiem i tryb blokowania
  - LOCK_SH - blokowanie tylko do odczytu (skrypt może odczytywać z pliku, plik może być jednocześnie czytany przez wiele skryptów)
  - LOCK_EX - blokowanie tylko do zapisu (skrypt uniemożliwia w tym momencie zapis do pliku innym skryptom/programom niż tylko on sam, może wtedy bezpiecznie zapisywać)
  - LOCK_UN - całkowite odblokowanie pliku

```
// odczytujemy stan licznika
zapisany w pliku licznik.dat

function odczytaj()
{
global $ile_wejsc;
// otwieram do odczytu
$plik_licznika =
fopen('licznik.dat', 'r');
// blok do odczytu
flock($plik_licznika, LOCK_SH);
// odczyt
$ile_wejsc = fread($plik_licznika,
100);
// odblokowanie
flock($plik_licznika, LOCK_UN);
//zamknięcie
fclose($plik_licznika);
// inkrementacja licznika
$ile_wejsc++;
}
```

Blokowanie pliku

```
// funkcja zapisuje stan licznika
function zapisz($ile_wejsc)
{
// otwieram do zapisu
$plik_licznika =
fopen('licznik.dat','w');
// blok na wyłączność do zapisu
flock($plik_licznika, LOCK_EX);
// zapis
fwrite($plik_licznika, $ile_wejsc);
// odblokowuję
flock($plik_licznika, LOCK_UN);
// zamykam
fclose($plik_licznika); }
```

```
// wlasciwy skrypt
// jesli istnieje plik licznika, to
// odczytujemy stan z pliku

if (file_exists('licznik.dat'))
odczytaj();


// nie ma pliku -
// czyli stan licznika = 1,
// a plik zostanie utworzony
// przy otwarciu go do zapisu
// w funkcji zapisz()

else $ile_wejsc = 1;


// zapisujemy stan licznika


zapisz($ile_wejsc);

// wyswietlam liczbe wejsc

 echo($ile_wejsc);
```

Podsumowanie operacji na plikach

```
$file = fopen($plik, "r");
$zawartosc = fread($plik, filesize($nazwa-pliku));
```

czyta od początku do końca pliku, *lub*

```
$zawartosc = fgets($plik, filesize($nazwa-pliku));
```

lub

```
$zawartosc = file($plik);
fclose($file);
echo $zawartosc;
```

Można podawać liczbę znaków do odczytania

Sesje w PHP

- HTTP – jest protokołem bezstanowym co oznacza, że nie można ustalić powiązań danych wysłanych przez konkretnego (jednego) użytkownika
- PHP posiada metody takiej kontroli
  - Pierwsza wizyta w serwisie związana jest z utworzeniem unikatowego identyfikatora przesyłanego z kolejnymi żądaniami za pomocą *„ciastek”* lub *PHPSESID*
  - Na podstawie tego odczytywany jest odpowiedni plik sesji zapisany na serwerze
  - W pliku tym znajdują się informacje o działalności klienta

Sesje w PHP

- Inicjowanie sesji w PHP to wywołanie funkcji

```
Session_start()
```

- Od tego wywołania zadeklarowane zmienne sesyjne zostają zapisane do tablicy *$_SESSION*

```
session_start();
if(!isset($_SESSION['licznik']))
{
$_SESSION['licznik'] = 0;
}
$_SESSION['licznik']++;
echo 'Odwiedziłeś już '.$_SESSION['licznik'].' podstron!';
```

- Każdy część serwisu po włączeniu sesji musi posiadać włączoną obsługę zmiennych sesyjnych

Sesje w PHP

- Autoryzacja klientów
  - Otwarcie formularza logowania w bezpiecznym połączeniu https
  - Uzupełnienie danych logowania
  - Wysłanie danych do autoryzacji z formularza
  - Włączenie obsługi sesji
  - Sprawdzenie w skrypcie logowania czy dane logowania są poprawne z zapisanymi w serwisie
  - Powołanie do życia zmiennych sesyjnych i ich rejestracja:

```
$_SESSION['abc']='abc';
```

  - Wylogowanie i likwidacja sesji

```
session_start(); session_destroy()
```

Cookies

- Cookie – jest małym plikiem tekstowym, który serwer przesyła do przeglądarki klienta
- Plik ten zawiera informacje tekstowe ułatwiające wyświetlanie serwisu zarówno klientowi jak i dostosowanie serwisu do potrzeb klienta (hasła, identyfikatory, ustawienia linków,…)
- Aby ustawić cookie w systemie klienta korzysta się z funkcji *setcookie*
  `setcookie('ciacho', 'wartosc');`
- Funkcja posiada kilka argumentów, oto trzy najważniejsze:
  - nazwa ciasteczka
  - wartość nadana przez skrypt
  - czas, przez jaki „ciacho” może być wykorzystywane [s] - liczy czas od 01-01-1970 r. i dla wygody zastępowany jest zwykle kombinacją: time() +czas

```
setcookie('ciacho', 'wartosc', time() + 3600);
```

- Taki zapis oznacza, że „ciacho” przechowuje wartości „wartość” przez czas godziny.

Cookies

- Odczytanie zawartości ciastka realizowane jest przez odczyt tablicy $_COOKIE

```
$_COOKIE['ciacho'];
```

- Funkcja setcookie musi być wykonana przed wysłaniem kodu HTML do przeglądarki, inaczej cookie nie będzie zapisane
- Cookie zadziała dopiero po kolejnym wejściu na stronę

- Klika przykładów zastosowania cookie

Obsługa formularza

- Metoda POST – wysłanie danych z formularza jako część nagłówka pakietu HTTP
- Metoda GET – wysłanie danych z formularza za pomocą adresu

```
<form action="obsluga.php" method="post">
pola formularza i opisy
</form>
```

- Jak wysyłane są dane z formularza odczytywane potem w PHP
  - Wysyłając wartości z pól formularza tworzone są zmienne tablicowe $_POST lub/i $_GET
  - Pola formularza oznaczone atrybutem *name* zostają zapisane w tablicach wraz z wysyłaną zawartością jako elementy tablicy
- Dostęp do poszczególnych pól i wartości przekazanych skryptowi:

```
$nazwa=$_POST['nazwa'];
$nazwa=$_GET['nazwa'];
```

- Obsługa zmiennych z formularza

```
echo "Nazywam się".$_POST['nazwa'];
```

Obsługa formularzy

```
<form action=„obsluga.php" method="post">
<p>Podaj login: <input type="text" name="login" /></p>
<p>Podaj hasło: <input type="password" name="password"/></p>
<p><input type="submit" value="zatwierdź" /></p> </form>
```

```
$login = 'Janek';
$password = 'Kowalski';
if ($login == $_POST['form_login'] && $password ==
$_POST['form_password'])
{
header('Location: tajne.php');
}
else
{
echo('ACCESS DENIED');
}
```

Dostęp do bazy danych

- Deklaracja połączenia się z bazą danych

```
$connection= @mysql_connect('localhost',’user’, ‘password’) or die ("Nie udało się");
if ($connection) echo "Połączyłeś się z bazą=";
```

- Wybór konkretnej bazy danych

```
mysql_select_db('ksiegarnia');
```

- Wykonywanie działań na bazie danych za pomocą poleceń SQL
  - mysql_query (*SQLstring, connection*) funkcja - wysyła zapytanie do aktywnej bazy na serwerze skojarzonym z podanym identyfikatorem połączenia - parametrem jest tekst zapytania MySQL (podany dosłownie lub wcześniej zdefiniowana zmienna tekstowa)
  - **mysql_query()** zwraca identyfikator wyniku (lub **FALSE** w przypadku niepowodzenia) jedynie dla zapytań typu SELECT, SHOW, EXPLAIN i DESCRIBE

```
$wynik =mysql_query($query);
```

  - Każdą kwerendę MySQL można wykonać w PHP
  - W przypadku kwerendy SELECT zmienna $wynik jest tzw. uchwytem do zasobu - *handle*

Prezentacja danych z bazy w serwisie

- mysql_num_rows($zapytanie)
  - zwraca liczbę znalezionych rekordów (SELECT)
  - Aby pobrać ilość wierszy przetworzonych w operacjach INSERT, UPDATE lub DELETE należy użyć funkcji mysql_affected_rows()

```
$ile = mysql_num_rows($wynik);
echo "Rekordów:". $ile."<BR>";
```

- mysql_fetch_array($zapytanie [,TYP])
  - zwraca tablicę (wiersz) składającą się z pól znalezionego rekordu – sekwencyjnie uruchamiana zwraca kolejne rekordy, parametrem jest zmienna rezultatu zapytania SQL
  - typy – MYSQL_NUM, MYSQL_ASSOC, MYSQL_BOTH

```
mysql_fetch_array($wynik,MYSQL_NUM);
```

Prezentacja danych z bazy w serwisie

- mysql_fetch_row ($zapytanie)
  - Zwraca tablicę zawierającą wiersz rekordu lub **FALSE** jeżeli nie ma więcej wierszy w *wynik*
  - pobiera jeden wiersz danych z wyniku skojarzonego z podanym identyfikatorem wyniku
  - Wiesz zwracany jest jako tablica
  - Komórki są umieszczone pod oddzielnymi "ofsetami", zaczynając od 0
  - Kolejne wywołanie MYSQL_FETCH_ROW () zwróci następny wiersz z wyniku, lub **FALSE** jeżeli nie ma więcej wierszy
- mysql_close ($connection) zamknięcie połączenia z bazą danych

Baza danych - podsumowanie

- połączenie z Mysql – mysql_connect
- otwarcie (wybór) bazy - mysql_select_db
- sformułowanie zapytania $zmienna='select..
- wykonanie zapytania mysql_query
- ewentualne wyświetlenie wyników zapytania z wykorzystaniem uchwytu do zasobu funkcją mysql_fetch_array
- zamknięcie połączenia mysql_close
  - (niekonieczne – po zakończeniu skryptu zamyka się automatycznie)

Baza danych – przykład 1

```
<?php
$polaczeniedb=mysql_connect('localhost', 'jozio', 'kowal') or
    die('Nie można się połączyć: ' . mysql_error());
mysql_select_db('hurtownia');
$zapytanie = mysql_query("SELECT kod_towaru, opis FROM
  kategorie");
while ($wynik = mysql_fetch_array($zapytanie, MYSQL_NUM))
{
    printf ("ID: %s  Nazwa: %s ", $wynik[0], $wynik[1]);
    echo "<BR>";
}
mysql_close($polaczeniedb);
?>
```

Baza danych – przykład 2

```
<?php
$polaczenie=mysql_connect('localhost', 'jozio', 'kowal') or
    die('Nie można się połączyć: ' . mysql_error());
mysql_select_db('hurtownia');
$zapytanie = mysql_query("SELECT kod_towaru, opis FROM
  kategorie");
while ($wynik = mysql_fetch_array($zapytanie, MYSQL_ASSOC))
{
    echo  "ID:". $wynik['kod_towaru']." Nazwa:".
  $wynik['opis']."<BR>";
    echo "<BR>";
}
mysql_close($polaczenie);
?>
```

Baza danych – przykład 3

```
<?php
$polaczenie=mysql_connect('localhost', 'jozio', 'kowal') or
    die('Nie można się połączyć: ' . mysql_error());
mysql_select_db('hurtownia');
$zapytanie = mysql_query("SELECT kod_towaru, opis FROM
  kategorie");
while ($wynik = mysql_fetch_array($zapytanie, MYSQL_BOTH))
{
printf ("ID: %s  Nazwa: %s", $wynik["kod_towaru"],
  $wynik[1]);
 echo "<BR>";
}
mysql_close($polaczenie);
?>
```

Baza danych – przykład for

```
//wypisujemy w iteracji rekordy:
...
$ile=mysql_num_rows($wynik);
for ($i=0; $i<$ile; $i++)
{
$wiersz=mysql_fetch_array($wynik);
  $wiersz['id_ks'] .
  stripslashes($wiersz['autor']) .
  stripslashes($wiersz['tytul']) .
  $wiersz['cena'] ."  zł<BR>";
};
```

Stripslashes usuwa znaki \ z linijek tekstu

Dostęp do zmiennych środowiskowych

- PHP posiada wiele zmiennych informujących o parametrach serwera lub klienta
- Wartości przechowywane są w zmiennych środowiskowych
- Zamieszczone są one w tablicy *$_SERVER*
- Dostęp do wartości środowiskowych realizowany jest: `echo $_SERVER['NAZWA_ZMIENNEJ'];`
- Lub przez specjalną funkcję `getenv('NAZWA_ZMIENNEJ');` , która pobiera wartość zmiennej i zwraca jej wartość lub *false* gdy takiej zmiennej nie ma
- PHP posiada jeszcze jedną funkcje globalnego przeglądania zmiennych środowiskowych `phpinfo();`

Dostęp do zmiennych środowiskowych

- Niektóre ze zmiennych środowiskowych:
  - $_SERVER['SERVER_NAME'] - Nazwa hosta (np. localhost)
  - $_SERVER['SERVER_SOFTWARE'] - Oprogramowanie serwera, np. Apache
  - $_SERVER['SERVER_PROTOCOL'] - Protokół, jakim jest przesyłana strona
  - $_SERVER['DOCUMENT_ROOT'] - Katalog główny, w którym działa skrypt
  - $_SERVER['HTTP_REFERER'] - Strona oglądana jako poprzednia, z której klient przeniósł się a aktualnie kwyświetlaną
  - $_SERVER['HTTP_USER_AGENT'] – Przeglądarka klienta
  - $_SERVER['REMOTE_ADDR'] – Numer IP klienta (Internet Protocol)
  - $_SERVER['SERVER_ADMIN'] - Adres e-mail administratora serwera
  - $_SERVER['SERVER_SIGNATURE'] - "Podpis" serwera, który jest wyświetlany na stronach generowanych przez serwer (np. niektóre komunikaty błędu)

Kilka pożytecznych funkcji

- Działanie na tekstach jest dość powszechnym zajęciem przy tworzeniu aplikacji dla www
- Operator konkatencji „.”

```
$tekst1 = 'Ala ma kota';
$tekst2 = ', a ja';
$tekst3 = ' mam kota na punkcie Ali :))';
$tekst = $tekst1.$tekst2.$tekst3; echo $tekst;
```

- Pozbawianie „białych znaków”

```
$przed = '      Tekst do obcięcia funkcją trim ';
$po = trim($przed); echo $po;
```

- Automatyczne zamienianie znaków specjalnych (encji) np. < *na &lt, & na &amp*

```
$przed = '<a
  href="costam.php?cos=2&cos_innego=3">odsylacz</a>';
$po = htmlspecialchars($przed);
echo $po;
```

Kilka pożytecznych funkcji

- Opuszczanie tagów np. zamiana przesłanego kodu przez klienta:* Fajna strona!!! *

```
$wpis= strip_tags($wpis);
```

- Zamiana małych liter na wielkie i odwrotnie. Przydaje się gdy zapisujemy dane z formularza do bazy danych w okreslonym formacie

```
$fraza = strtolower(NAPISANE DUŻYMI LITERAMI!'); // Otrzymujemy „napisane dużymi literami!";
$fraza = strtoupper($fraza); // i znów to co na początku
```

- Wyciąganie fragmentów testu z ciągu znaków:

```
$tekst = 'Przychodzi baba do lekarza';
$fragment = substr($tekst, 11);
echo $fragment; // wypisze "baba do lekarza"
echo substr($tekst, 16, 2); // wypisze "do"
echo substr($tekst, -7); // wypisze ostatnich 7 znaków, czyli "lekarza„
echo substr($tekst, 0, -8); // pomijamy lekarza i spację, więc echo wypisze "Przychodzi baba do"
```

Kilka pożytecznych funkcji

- Wyszukiwanie znaków w ciągu
  - Czasem w połączeniu z *substr* stosuje się funkcję *strpos*
  - Wyszukiwanie w tekście oznaczonym jako pierwsza zmienna znaku, który jest określony jako druga zmienna
  - Funkcja zwraca pozycje znaku na którym został znaleziony lub false
```
$email = 'kochamy_spam@spamspam.pl';
if (($poz = strpos($email, '@')) !== false)
echo 'To jest e-mail';
echo substr($email, 0, $poz);
```
  - (!== sprawdza nie tylko wartości ale i typ wyrażenia) jeśli „!=„ i @ byłaby na 1 miejscu to funkcja zwróciłaby 0 a nie false.

Netografia

- PHP i MySQL. Tworzenie stron internetowych. Helion
- http://pl.docs.pld-linux.org/
- http://coogi.cba.pl
- 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 funkcja w PHP?
- Jak wygląda konstrukcja funkcji w PHP?
- Co to są zmienne globalne i lokalne?
- Na czym polega obsługa plików w PHP?
- Jak wygląda procedura dostępu do pliku w PHP?
- Pobieranie danych z plików danych ?
- Co to są sesje PHP?
- Jak obsługiwać sesje PHP?
- Co to jest cookie ?
- Co moze zwierać cookie?
- Na czym polega obsługa formularzy?
- Zmienne środowiskowe w PHP?
- Jak wygląda procedura łączenia siez bazą danych?
- Jak prezentować dane z bazy danych w serwisie?

Koniec

Temat następnego wykładu to:

Java Script