Bazy Danych i Usługi Sieciowe

Język PHP

Paweł Daniluk

Wydział Fizyki

Jesień 2013

Plan wykładu

1. Język PHP
2. Składnia PHP
3. Dostęp do bazy danych
4. Przetwarzanie danych z formularzy

Język PHP

1. **P**HP: **H**ypertext **P**reprocessor
2. Język skryptowy interpretowany na serwerze www
3. Kod może być zanurzany w plikach HTML
4. Służy to tworzenia interaktywnych aplikacji internetowych
5. Skrypty PHP generują kod HTML wysyłany do klienta przez serwer

Przykład

Witaj świecie - HTML + PHP

```
<html>
<head>
<title>PHP: Witaj świecie</title>
</head>
<body>
<p><?php echo "Witaj świecie!"; ?></p>
</body>
</html>
```

Przykład c.d.

Witaj świecie - wygenerowany HTML

```
<html>
<head>
<title>PHP: Witaj świecie</title>
</head>
<body>
<p>Witaj świecie!</p>
</body>
</html>
```

Widok w oknie przeglądarki

Witaj świecie!

Jak działa PHP?

1. Serwer www dostaje żądanie zasobu zawierającego kod PHP
2. Przekazuje zasób do interpretera PHP, który wykonuje instrukcje zawarte w znacznikach <?php i ?>
3. Interpreter PHP zastępuje fragmenty w znacznikach <?php i ?> przez kod HTML wygenerowany przez skrypty
4. Serwer www dostaje wygenerowany kod HTML od interpretera PHP
5. Kod HTML jest wysyłany do klienta

Przetwarzanie dokumentu PHP

1. Znaczniki HTML poza fragmentami ograniczonymi <?php i ?> są przepisywane bez zmian
2. Kod otoczony <?php i ?> jest interpretowany, wykonywany i zastępowany przez wypisywany tekst

Przetwarzanie dokumentu PHP

```
<h1>Tytuł</h2>
<?php echo '<h3>Podtytuł</h3>'; ?>
<p>Dalszy ciąg tekstu.</p>
<p><?php echo 'Zakończenie'; ?></p>
```

Co można zrobić przy pomocy PHP?

1. Używać wielokrotnie fragmentów kodu w różnych plikach HTML
2. Przetwarzać dane z formularzy
3. Generować raporty z bazy danych
4. Wysyłać wiadomości e-mail
5. Generować obrazy
6. Wykonywać operacje na plikach

Instrukcje, zmienne, komentarze

Koniec instrukcji - ;

```
echo 'Witaj świecie';
```

Zmienne - $

```
echo $napis;
```

Komentarze - // lub /*...*/

```
echo 'Witaj'; // Komentarz do końca linii
/* Komentarz
w kilku
liniach */
```

Instrukcja warunkowa

if-else

```
<?php
if ($zmienna1 == $zmienna2 ) {
   echo '<p>Zmienne są równe</p>';
}
else {
   echo '<p>Zmienne nie są równe</p>';
}
?>
```

Instrukcja warunkowa w kawałkach

if-else

```
<?php
if ($zmienna1 == $zmienna2) {
?>
<p>Zmienne są równe.</p>
<?php
} else {
?>
<p>Zmienne nie są równe.</p>
<?php
}
?>
```

Typy danych

1. Cztery typy skalarne
    1. `boolean`
    2. `integer`
    3. `float`
    4. `string`
2. Dwa typy złożone
    1. `array`
    2. `object`
3. Dwa typy specjalne
    1. `resource`
    2. `NULL`

Typy danych - przykłady

```
$logiczna = TRUE;
$napis1 = "Tekst";
$napis2 = 'Inny tekst';
$liczba1 = 12;
$liczbaNapis = "15";
$wynik = $liczba1 + $liczbaNapis;
echo $wynik; // Zostanie wypisane 27
```

Wartości logiczne

1. Słowa kluczowe `TRUE` i `FALSE`
2. Wielkość liter nie ma znaczenia
3. Liczby i napisy mogą być rzutowane na wartości logiczne
4. Jako `FALSE` uznawane są
   1. liczba całkowita 0
   2. liczba zmiennoprzecinkowa 0.0
   3. pusty łańcuch znaków ""
   4. łańcuch znaków "0"
   5. pusta tablica
   6. pusty obiekt
   7. NULL
5. Pozostałe wartości interpretowane są jako `TRUE` (w tym np. -1)

Łańcuchy znaków

Pojedynczy cudzysłów

```
echo 'Jakiś tekst';
echo 'Arnold once said: "I´ll be back"'
```

Podwójny cudzysłów - dodatkowe przetwarzanie

```
echo "Jakiś tekst \n i dalszy ciąg w nowej linii";
echo "Wartością zmiennej jest: $zmienna";
```

Operacje na łańcuchach znaków

echo

```
<?php echo 'Witaj świecie';?>
```

echo ze zmienną

```
<?php $imie = 'Tomasz'; echo "Witaj $imie"; ?>
```

Operatory

Operatory arytmetyczne

Operator	Opis	Wartość
5 - 3	odejmowanie	2
5 + 3	dodawanie	8
5 * 3	mnożenie	15
15 / 3	dzielenie	5
16 % 5	reszta z dzielenia	1

Operatory

Operatory porównania

Operator	Opis	Kiedy prawda
$a == $b	równe	$a i $b są równe bez kontroli typu
$a === $b	tożsame	$a i $b są równe i tego samego typu
$a < $b	mniejsze	$a jest mniejsze od $b
$a > $b	większe	$a jest większe od $b
$a <= $b	mniejsze równe	$a jest mniejsze lub równe $b
$a >= $b	większe równe	$a jest większe lub równe $b
$a != $b	nierówne	$a i $b są różne bez kontroli typu
$a !== $b	nietożsame	$a i $b są różne lub róznych typów
$a <> $b	nierówne	$a i $b są różne bez kontroli typu

Operatory

Operatory logiczne

Operator	Opis	Kiedy prawda
$a and $b	koniunkcja	$a i $b są prawdą
$a && $b	koniunkcja	$a i $b są prawdą
$a or $b	alternatywa	$a lub $b są prawdą
$a || $b	alternatywa	$a lub $b są prawdą
$a xor $b	alternatywa wykluczająca	$a albo $b są prawdą
!$a	negacja	$a nie jest prawdą

Operatory

Operatory przypisania

Wyrażenie	Wartości zmiennych po wykonaniu
`$a = 5`	a=5
`$b = $a = 5`	a=5, b=5
`$c = $a++`	a=6, c=5
`$d = ++$a`	a=7, c=6
`$a += 3`	a=10
`$a -= 5`	a=5
`$a *= 3`	a=15
`$a /= 5`	a=3
`$a .= ' słonie'`	a= 3 słonie

Dostęp do bazy danych

1. Istnieją rozszerzenia pozwalające na połączenia z różnymi bazami danych
   1. SQLite;
   2. MySQL
   3. PostgreSQL
   4. MSSQL
   5. Oracle
2. Do połączenia z bazą konieczne są dane użytkownika i hasło
3. PHP musi mieć uprawnienia do połączenia z bazą ze swojego serwera

Dostęp do bazy danych

Połączenie i zapytanie

```
<?php
$link = mysql_connect('mysql_host', 'mysql_user', 'mysql_password');
mysql_select_db('my_database');
$query = 'SELECT * FROM my_table';
$result = mysql_query($query);
```

Dostęp do bazy danych c.d.

Odczyt danych z tabeli

```
echo "<table>\n";
while ($line = mysql_fetch_array($result, MYSQL_ASSOC)) {
echo "\t<tr>\n";
foreach ($line as $col_value) {
echo "\t\t<td>$col_value</td>\n";
}
echo "\t</tr>\n";
}
echo "</table>\n";
```

Dostęp do bazy danych c.d.

Zakończenie połączenia

```
mysql_free_result($result);
mysql_close($link);
?>
```

Dostęp do bazy danych c.d.

Wygenerowany kod HTML

```
<table>
<tr>
<td>1</td>
<td>Anna</td>
<td>Babacka</td>
</tr>
<tr>
<td>2</td>
<td>Matylda</td>
<td>Babacka</td>
</tr>
<tr>
<td>3</td>
<td>Tomasz</td>
<td>Cabacki</td>
</tr>
```

Dostęp do bazy danych c.d.

Widok w przeglądarce

1	Anna	Abacka
2	Matylda	Babacka
3	Tomasz	Cabacki

Formularze w HTML

1. Znacznik `<form>`
2. Elementy formularza - znaczniki `<input>`
3. Typ pola określany przez atrybut `type`
   1. Pole tekstowe `<input type=''text''>`
   2. Pole jednokrotnego wyboru (radio) `<input type=''radio''>`
   3. Pole wielokrotnego zaznaczenia (checkbox) `<input type=''checkbox''>`
   4. Pole hasła (nie wyświetla wpisywanych znaków) `<input type=''password''>`
   5. Przycisk wysyłania `<input type=''submit''>`
   6. Przycisk czyszczenia `<input type=''reset''>`

Przykładowy formularz

Pytanie o imię, nazwisko i płeć

```
<form>
Imię: <input type="text" name="imie" /><br />
Nazwisko: <input type="text" name="nazwisko" /><br />
Płeć: <input type="radio" name="plec" value="M" /> Mężczyzna<br />
<input type="radio" name="sex" value="K" /> Kobieta
<input type="submit" value="Wyślij" />
<input type="reset" value="Wyczyść" />
</form>
```

Widok w przeglądarce

Imię:

Nazwisko:

Płeć:

- Mężczyzna
- Kobieta

Wyślij Wyczyść

Co się dzieje z danymi?

1. Dane z formularza wysyłane są do serwera przy po kliknięciu w przycisk `submit`
2. Dwie metody
   1. **GET** - dane zakodowane w adresie URL
   2. **POST** - dane zapisane treści żądania do serwera
3. Konieczne jest podanie skryptu, który zajmie się danymi otrzymanymi przez serwer
4. Atrybut **action**
5. Ścieżka dostępu i nazwa pliku, który przetwarza dane

Wysyłanie danych metodą POST

```
<form action="witaj.php" method="POST">
Imię: <input type="text" name="imie" /><br />
Nazwisko: <input type="text" name="nazwisko" /><br />
<input type="submit" value="Wyślij" />
</form>
```

Dostęp do danych POST

witaj.php

```
<html>
<head>
</head>
<body>
<p>Witaj <?php echo $_POST["imie"]; ?>!</p>
<p>Twoje nazwisko to <?php echo $_POST["nazwisko"]; ?>.</p>
</body>
</html>
```

Widok w przeglądarce po wysłaniu

Witaj Tomasz!
Twoje nazwisko to Cabacki.

Wysyłanie danych metodą GET

```
<form action="witaj.php" method="GET">
Imię: <input type="text" name="imie" /><br />
Wiek: <input type="text" name="wiek" /><br />
<input type="submit" value="Wyślij" />
</form>
```

Widok w przeglądarce

Imię: Tomasz

Wiek: 32

Wyślij

Adres URL wysłany do serwera

http://mojastrona.com/witaj.php?imie=Tomasz&wiek=32

Dostęp do danych GET

witaj.php

```
<html>
<head>
</head>
<body>
<p>Witaj <?php echo $_GET["imie"]; ?>!</p>
<p>Masz <?php echo $_GET["wiek"]; ?> lat.</p>
</body>
</html>
```

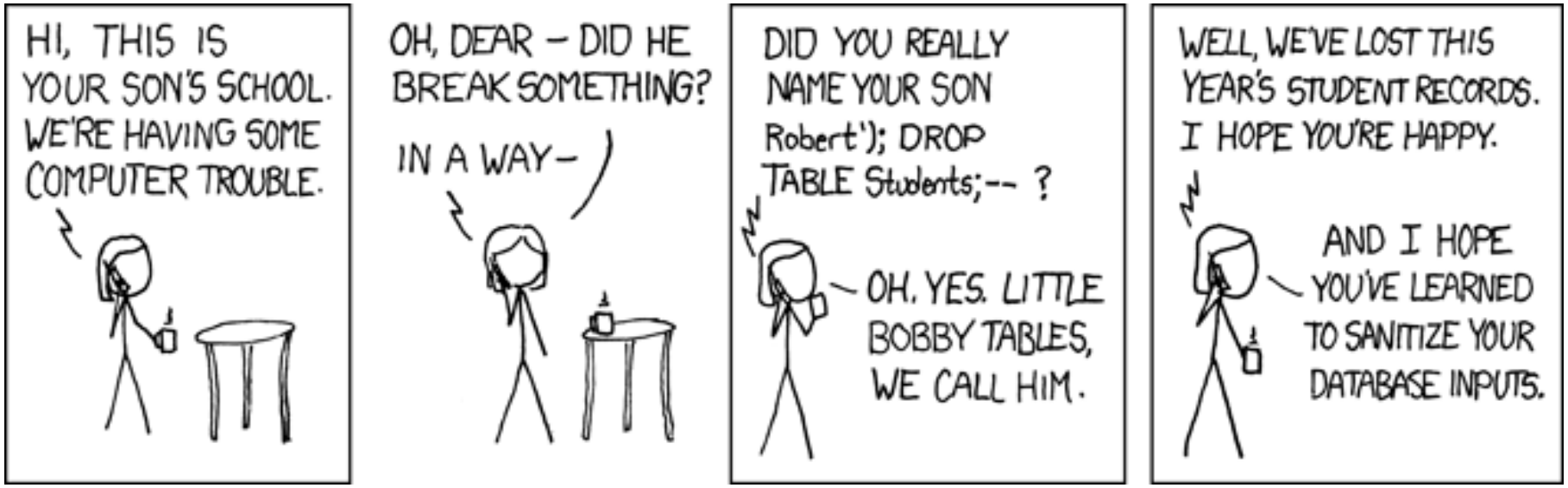
Widok w przeglądarce po wysłaniu

Witaj Tomasz!
Masz 32 lat.

Kiedy POST a kiedy GET?

1. GET
   1. Efekt wielokrotnego wysłania tych samych danych jest taki sam jak efekt jednokrotnego wysłania
   2. Brak efektów ubocznych, albo są one nieistotne
   3. Mała ilość danych
   4. Tylko przy pobieraniu danych z bazy
2. POST
   1. Wysłanie formularza powoduje dodatkowe efekty uboczne, które nie powinny być powtarzane
   2. Duża ilość danych
   3. Pobieranie, dodawanie, usuwanie danych z bazy

SQL injection ;)



Rysunek : http://xkcd.com/327/

Walidacja i zabezpieczenia

1. Po stronie klienta
   1. JavaScript
   2. Przed wysłaniem danych z formularza
   3. Wyrażenia regularne - poprawność danych
   4. Nie gwarantuje poprawności danych
   5. Dla wygody użytkownika
2. Po stronie serwera
   1. PHP,...
   2. Po wysłaniu danych z formularza
   3. Usuwanie niebezpiecznych znaków
   4. Dodawanie \ przed ' i "(*addslashes()*)
   5. Wykorzystanie funkcji dla używanej bazy (*mysql_real_escape_string()*)
   6. Wyrażenia regularne - poprawność danych
   7. Dla bezpieczeństwa

Alternatywy – CGI

- Common Gateway Interface
- Programy napisane w dowolnym języku (np. C, najczęściej Perl)
- Dane o żądaniu HTTP w zmiennych środowiskowych
- Dane z **POST** i **PUT** na standardowym wejściu
- Zawartość generowana przez program na standardowe wyjście

Przykład – Perl

```
#!/usr/bin/perl

print "Content-type: text/plain\n\n";

for my $var ( sort keys %ENV ) {
    my $value = $ENV$var;
    $value =~ s/n/\n/g;
    $value =~ s/"/\"/g;
    print qq[$var="$value"\n];
}
```

Alternatywy – CGI

Request

```
http://example.com/cgi-bin/printenv.pl/foo/bar?var1=value1&
var2=with%20percent%20encoding
```

Zmienne środowiskowe

```
GATEWAY_INTERFACE="CGI/1.1"
HTTP_ACCEPT="text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8"
HTTP_ACCEPT_CHARSET="ISO-8859-1,utf-8;q=0.7,*;q=0.7"
HTTP_ACCEPT_ENCODING="gzip, deflate"
HTTP_ACCEPT_LANGUAGE="en-us,en;q=0.5"
HTTP_CONNECTION="keep-alive"
HTTP_HOST="example.com"
HTTP_USER_AGENT="Mozilla/5.0 (Windows NT 6.1; WOW64; rv:5.0) Gecko/20100101 Firefox
PATH_INFO="/foo/bar"
PATH_TRANSLATED="/var/www/foo/bar"
QUERY_STRING="var1=value1&var2=with%20percent%20encoding"
REMOTE_ADDR="127.0.0.1"
REMOTE_PORT="63555"
REQUEST_METHOD="GET"
REQUEST_URI="/cgi-bin/printenv.pl/foo/bar?var1=value1&var2=with%20percent%20encodin
SCRIPT_FILENAME="/var/www/cgi-bin/printenv.pl"
SCRIPT_NAME="/cgi-bin/printenv.pl"
```

Alternatywy – Genshi

```
<!DOCTYPE html>
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:py="http://genshi.edgewall.org/">
  <head>
    <title>Geddit: News</title>
  </head>
  <body class="index">
    <div id="header">
      <h1>News</h1>
    </div>

    <ol py:if="links">
      <li py:for="link in reversed(links)">
        <a href="$link.url">$link.title</a>
        posted by $link.username at $link.time.strftime('%x %X')
      </li>
    </ol>

    <p><a class="action" href="/submit/">Submit new link</a></p>

    <div id="footer">
      <hr /><p class="legalese">© 2007 Edgewall Software</p>
    </div>
  </body>
</html>
```

Alternatywy – Genshi

Alternatywy – Django

```
{% extends "base.html" %}

{% block title %}Articles for {{ year }}{% endblock %}

{% block content %}
<h1>Articles for {{ year }}</h1>

{% for article in article_list %}
    <p>{{ article.headline }}</p>
    <p>By {{ article.reporter.full_name }}</p>
    <p>Published {{ article.pub_date|date:"F j, Y" }}</p>
{% endfor %}
{% endblock %}
```

```
<html>
<head>
    <title>{% block title %}{% endblock %}</title>
</head>
<body>
    <img src="sitelogo.gif" alt="Logo" />
    {% block content %}{% endblock %}
</body>
</html>
```