COOKIE

Czym są ciasteczka (z ang. cookies)? To niewielki fragment informacji, który aplikacja internetowa może zapisać na komputerze użytkownika. Gdy przeglądarka łączy się ze stroną, w pierwszej kolejności szuka lokalnych cookies, odpowiednich dla danej witryny. Jeżeli znajdzie – zostaną one przekazane serwerowi.

W PHP możemy ustawić cookie za pomocą funkcji setcookie().

```
<?php
// utworzenie ciasteczka
setcookie($nazwa, $wartosc, $koniec, $sciezka, $domena, $bezpieczne);
?>
```

Setcookie() może przyjąć 6 argumentów. Jednak wymagane jest podanie tylko pierwszego, którym jest nazwa ciasteczka.
$wartosc to wartość przypisana do ciastka, $koniec wyraża datę wygaśnięcia, $sciezka i $domena mogą być stosowane do określenia adresów, dla których cookie jest ważne. Ostatni argument, $bezpieczne, oznacza, że cookie nie będzie wysyłane przez zwykłe połączenie HTTP.

COOKIE

Ciasteczko (formalnie HTTP Cookie, w skrócie ang. cookie, tłumaczone czasem niepoprawnie jako plik cookie– mały fragment tekstu, który serwis internetowy wysyła do przeglądarki i który przeglądarka wysyła z powrotem przy następnych wejściach na witrynę. Używane jest głównie do utrzymywania sesji np. poprzez wygenerowanie i odesłanie tymczasowego identyfikatora po logowaniu. Może być jednak wykorzystywane szerzej poprzez zapamiętanie dowolnych danych, które można zakodować jako ciąg znaków. Dzięki temu użytkownik nie musi wpisywać tych samych informacji za każdym razem, gdy powróci na tę stronę lub przejdzie z jednej strony na inną.
Zabezpieczenia przeglądarek pozwalają na odczyt ciasteczek jedynie z domeny, na której zostały utworzone, lub domen niższego poziomu.

Zastosowanie cookies

Ciasteczka różnych rodzajów są stosowane najczęściej po logowaniu do utrzymywania sesji. Mogą jednak przechowywać inne tymczasowe dane jak stan elementów na stronie, czy historię odwiedzanych poprzednio stron (na danej witrynie). Umożliwia to tworzenie spersonalizowanych serwisów WWW (np. zapamiętanie stanu menu), obsługi logowania, prostych sond i liczników, „koszyków zakupowych” w internetowych sklepach, a także tworzenie statystyk użyteczności witryny oraz badanie preferencji użytkowników.Wszelkie zapamiętywanie zalogowanego użytkownika, oddania głosu w ankiecie itp. są obsługiwane właśnie z wykorzystaniem cookies. Zasada jest prosta – jeśli cookie istnieje, to znaczy, że głos został oddany. Jeżeli nie, można oddać głos, po czym cookie jest tworzone.

```
<?php

// zapisanie oddania głosu jednorazowego
setcookie('oddano_glos', '1');

// w przypadku gdy głosować można raz dziennie
setcookie('oddano_glos', '1', time()+3600*24);

?>
```

Sprawdzanie obecności cookies

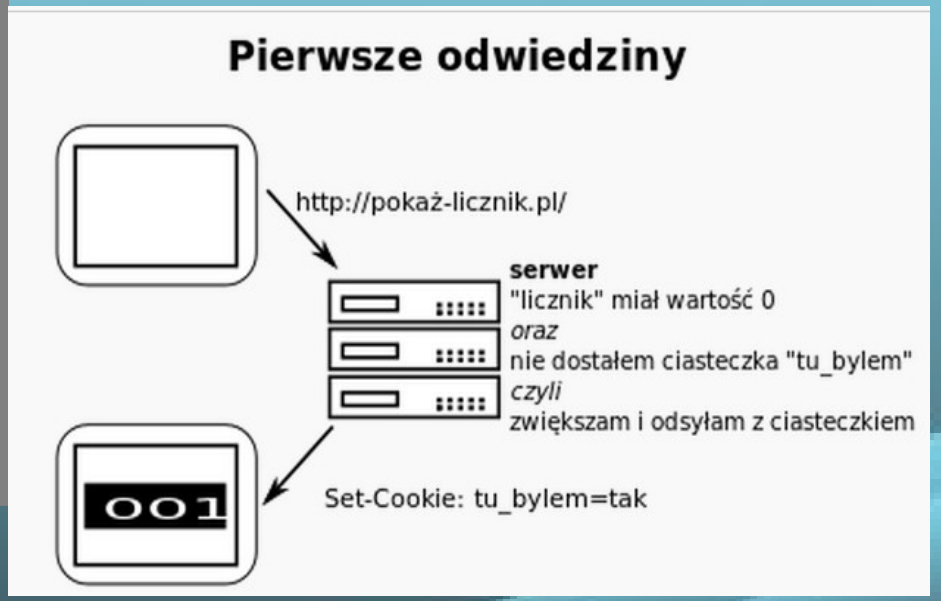
Każde ciasto jest przechowywane w tablicy globalnej $_COOKIES

```
<?php

if(isset($_COOKIE['aktywacja']))
   echo "Ciasteczko istnieje";
else
   echo "Brak ciastecza o nazwie aktywacja";
?>
```

Schemat działania cookies

Zastosowanie cookies do sond i liczników internetowych może wyglądać następująco – serwer ustawia ciasteczko informujące, że z danego komputera oddano już głos lub też odwiedzono daną stronę. Na tej podstawie może wykonać odpowiednie operacje i wygenerować dla użytkownika zindywidualizowaną treść strony.



Ilustracje obok - schematyczny sposób wykorzystywania ciasteczek przy obsłudze licznika internetowego, wykluczającego przeładowania (zwiększanie liczby odwiedzin przy odświeżeniu strony)

Cookies-licznik odwiedzin

```php
<?php
if(!isset($_COOKIE['tu_bylem'])){
    setcookie('tu_bylem','tak',time()+10);
    $plik=fopen('licznik.txt','r+') ;
    if(isset($plik))
    {
   // blokada pliku
     flock($plik, 2);
     $a=intval(fgets($plik));
      $a++;
      rewind($plik);
      fwrite($plik,$a);
      // odblokowanie pliku
flock($plik, 3);

fclose($plik);
    }
}
?>
<!DOCTYPE html>
<html>
<head>
<title>licznik odwiedzin</title>
<meta charset="utf-8">
</head>
<body>
<h2>witam na stronie</h2>
<?php
$plik1=fopen('licznik.txt','r') ;
    if(isset($plik1))
    { $b=intval(fgets($plik1));
fclose($plik1);
    }
echo '<h2>strone odwiedzono:'.$b.'razy </h2>';
?>
</body>
</html>
```

SESJE

Sesje służą do przekazywania danych między różnymi stronami WWW.

W odróżnieniu od zmiennych $_GET i $_POST nie są za każdym razem przesyłane od użytkownika do formularza po stronie serwera, lecz po jednorazowym ustaleniu ich wartości serwer będzie przechowywał ich stan po swojej stronie.

W momencie pierwszego trafienia na stronę interpreter tworzy specjalny, losowy oraz unikalny identyfikator przesyłany między żądaniami za pomocą ciastek lub parametru PHPSESSID doklejanego automatycznie do adresów URL. Na jego podstawie odczytywany jest później odpowiedni plik z danymi sesji zapisany gdzieś na serwerze. Pod koniec przetwarzania żądania wszystkie wprowadzone przez skrypt zmiany są z powrotem zapisywane do wspomnianego pliku tak, aby były widoczne przy wejściu na kolejną podstronę.

Programista w celu stworzenia sesji, wykonuje session_start(), co spowoduje albo utworzenie sesji jeśli nie istniała, ale odwołanie się do już istniejącej.

SESJE

Obsługa sesji w PHP ma na celu zapewnienie sposobu na zachowanie pewnych danych w trakcie następujących po sobie wywołań strony. Pozwala to na budowanie bardziej spersonalizowanych aplikacji i zwiększenie atrakcyjności strony internetowej.

Odwiedzający stronę WWW otrzymuje unikalny identyfikator, tzw. id sesji. Jest ono przechowywane albo jako ciasteczko po stronie użytkownika lub propagowane w URL'u.

Obsługa sesji pozwala na rejestrowanie dowolnej ilości zmiennych, które mają być przekazywane pomiędzy stronami. Otwierając stronę, PHP automatycznie sprawdzi (jeśli session.auto_start jest ustawione na 1) lub na życzenie (jawnie przez wywołanie session_start() lub niejawnie przez wywołanie session_register()) czy specyficzne id sesji zostało przypisane.

SESJE

Mechanizm sesji był najważniejszą ze zmian oczekiwanych w PHP 4. Umożliwia on przekazywanie parametrów między stronami w łatwy sposób. Zmienne są przechowywane po stronie serwera, a u klienta trzymane jest tylko ID sesji. Te ID jest zapisane w cookie lub przekazywane przez URL. PHP jest w stanie sam rozpoznać czy na komputerze klienta włączony jest mechanizm cookies i w razie potrzeby dodać identyfikator sesji do każdego nagłówka URL i formularza. Wymaga to jednak posiadania PHP skompilowanego z opcją –enable-trans-sid.

Ponieważ sesje mogą bazować na ciasteczkach, także i w tym przypadku przed rozpoczęciem sesji do przeglądarki nie mogą być wysłane żadne inne dane.

Po otrzymaniu żądania klienta PHP automatycznie (jeśli w konfiguracji PHP włączona została opcja auto_start) lub „ręcznie” przez programistę (za pomocą funkcji session_start()) sprawdza, czy przypisano już ID sesji. Jeśli tak, to PHP odczytuje zmienne zarejestrowane w tej sesji. Jeśli nie, generowany jest nowy, unikalny identyfikator sesji.

SESJE-
lista podstawowych funkcji

session_start(); The session_start() function must be the very first thing in your document. Before any HTML tags.

session_destroy(); - Niszczy wszystkie dane zarejestrowane w sesji

session_register('nazwa_zmiennej');//nie powinno być używane z $_SESSION
session_unregister('nazwa_zmiennej');//nie powinno być używane z $_SESSION

session_id(); Pobierz i/lub ustaw identyfikator bieżącej sesji
string session_id ([string id])

$_SESSION['nazwa_zmiennej']

session.name()--określa nazwę sesji, która jest używana jako nazwa ciastka. Powinna zawierać tylko znaki alfanumeryczne. Domyślnie PHPSESSID.
Ogólnie:
string session_name ([string nazwa]) zwraca nazwę bieżącej sesji. Jeśli podano parametr name, nazwa bieżącej sesji zostanie zmieniona na tą wartość.

SESJE-przykład

```php
<?php
session_start();

var_dump($_SESSION);
?>
<!DOCTYPE html>
<html>
<body>

<?php
//
$_SESSION["kolor"] = "green";
$_SESSION["zwierze"] = "kot";
echo 'zmienne sesyjne są utworzone <br>';
echo 'identyfikator sesji '.session_id();
?>

</body>
</html>
```

SESJE-licznik odwiedzin podstron

```php
<?php

    session_start();

    if(!isset($_SESSION['licznik']))
    {
        $_SESSION['licznik'] = 0;
    }

    $_SESSION['licznik']++;

    echo 'Odwiedziłeś już '.$_SESSION['licznik'].' podstron!';
?>
```

Sesje- zadanie1

```html
<!DOCTYPE html>
<html>
<head>
<title>formularz</title>
<meta charset="utf-8">
</head>
<body>
nazwa pliku: "form.php" <br>
<form action="session1.php" method="post">
    login:<input type="text" name="login">
    hasło:<input type="password" name="password">
    <input type="submit" name="Wyslij">
</form>
</body>
</html>
```

```php
<?PHP
//nazwa pliku  "session1.php"
        session_start();
        echo "Ustalam login=".$_POST['login'];
        $_SESSION['login']=$_POST['login'];
        echo "Ustalam hasło=".$_POST['password'];
        $_SESSION['password']=$_POST['password'];
?>
```

```php
<?PHP
//nazwa pliku "session2.php"
session_start();

echo "Twój login to: ".$_SESSION['login']."<br>";
echo "Twoje haslo to: ". $_SESSION['password']."<br>";

?>
```

Tą stronę otwórzmy w nowej zakładce przeglądarki lub po prostu podajmy jej adres ręcznie (czyli dane nie zostaną przekazane przez POST/GET). Zauważymy że mogliśmy się odnieść do danych zapisanych w sesji.

Destroy a PHP Session

bool **session_destroy** (void) Niszczy wszystkie dane zarejestrowane w sesji

session_destroy() niszczy wszystkie dane skojarzone z bieżącą sesją. Nie usuwa żadnych globalnych zmiennych związanych z sesją. Nie usuwa też ciasteczka sesyjnego.

```php
<?php
session_start();
?>
<!DOCTYPE html>
<html>
<body>

<?php
// remove all session variables
session_unset();

// destroy the session
session_destroy();
?>

</body>
</html>
```

void **session_unset** (void) -- Zwolnij wszystkie zmienne sesyjne

Funkcja **session_unset()** zwalnia wszystkie zmienne sesyjne, które są aktualnie zarejestrowane.

Notatka: Jeśli użyta została tablica *$_SESSION* (lub *$HTTP_SESSION_VARS* dla PHP 4.0.6 i starszych), aby wyrejestrować zmienną z sesji należy użyć **unset()**, na przykład *unset ($_SESSION['varname']);*.