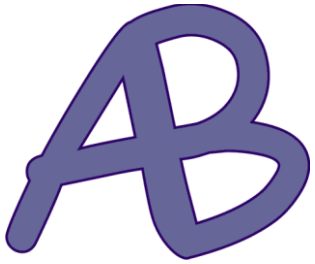


PHP

- ciasteczka

dr Artur Bartoszewski
UTH Radom

Ciasteczka

Ciasteczka

Cookies, czyli tzw. ciasteczka, to małe pliki tekstowe przechowywane na komputerze odbiorcy, w których przetrzymywane są informacje.

Gdy przeglądarka łączy się ze stroną, w pierwszej kolejności szuka lokalnych cookies, odpowiednich dla danej witryny. Jeżeli znajdzie – zostaną one przekazane serwerowi.

Ciasteczka zapisywane są w tablicy superglobalnej $_COOKIES

```php
<?php
  foreach($_COOKIE as $nazwa => $tresc){
      //echo "$nazwa - $tresc <br>";
  }
?>
```

Ciasteczka

PHP ma możliwość zapisu ciasteczek dzięki funkcji setcookie().

```
setcookie($nazwa, $wartosc, $koniec, $sciezka, $domena, $bezpieczne);
```

Ma ono 6 argumentów:

- $nazwa – nazwa pliku ciasteczka
- $wartosc - zawartość ciastka,
- $koniec - data wygaśnięcia,
- $sciezka - ścieżka do folderu na serwerze, z poziomu którego dane ciasteczko będzie dostępne
- $domena domenę lub poddomenę z poziomu której dane ciasteczko będzie dostępne
- $bezpieczne, gdy ustawione na TRUE to pliki cookie będą tworzone jedynie, gdy do połączenia użyto protokołu HTTPS (połączenie szyfrowane);

Wymagane jest podanie tylko pierwszego, którym jest nazwa ciasteczka.
W praktyce często podaje się pierwsze trzy.

Ciasteczka

Sprawdzanie obecności ciasteczka,

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

Odczytanie zawartości ciasteczka,

```
if (isset($_COOKIE['nazwa']))
  echo $_COOKIE['nazwa'];
else
  echo "Brak ciastecza";
```

Ciasteczka

Usuwanie ciasteczka,

```
setcookie("nazwa");
```

Polecenie `setcookie()` z jednym parametrem - tylko z nazwą, wbrew pozorom nie ustawia pustego ciasteczka lecz usuwa ciasteczko o wskazanej nazwie

Ciasteczka – czas życia

Czas życia ciasteczka, możemy określić datę ważności ciasteczka czyli czas po którym zostanie usunięte podajemy ją w czasie linuksowym.

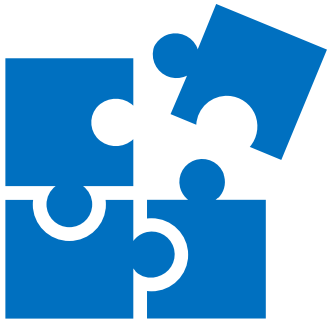
Ciasteczko które będzie istniało przez godzinę

```
setcookie('odwiedziny', "tresc", time() + 60 * 60);
```

sprawdzamy kiedy Użytkownik ostatnio odwiedził stronę. Mechanizm działa przez miesiąc

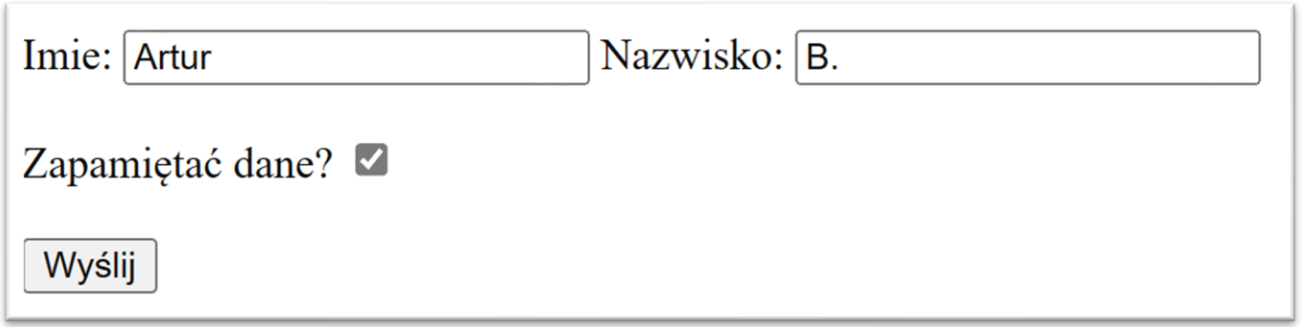
```
<?php
setcookie('odwiedziny', time(), time() + 30 * 24 * 60 * 60);
// time() - bieżąca data + 30 [dni] * 24 [godziny] * 60 [minuty] * 60 [sekundy] = czas wygaśnięcia ciasteczka
w sekundach
if (isset($_COOKIE['odwiedziny'])) {
  echo ("<p>Tę stronę odwiedziłeś:");
  echo (date('d.m.Y, H:i:s', $_COOKIE['odwiedziny']) . "</p>");
} else {
  echo ("<p>Jesteś tu pierwszy raz (od miesiąca)</p>");
}
?>
```

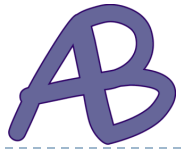
Przykład do wykonania



Program, który umożliwia użytkownikowi wprowadzenie swojego imienia i nazwiska za pomocą formularza.

Program korzysta z ciasteczek do zapamiętania wprowadzonych danych nawet po odświeżeniu strony czy ponownym wejściu na nią w przyszłości.



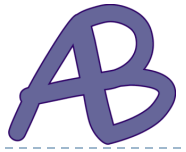


Przykład do wykonania

```php
<?php
if (isset($_POST['imie'])) $imie = htmlspecialchars($_POST['imie']);
else $imie = '';
if (isset($_POST['nazwisko']))
    $nazwisko = htmlspecialchars($_POST['nazwisko']);
else $nazwisko = '';
```

Sprawdzanie przesłanych danych:

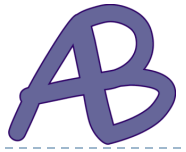
- Program sprawdza, czy dane zostały przesłane za pomocą formularza za pomocą funkcji `isset($_POST['imie'])` i `isset($_POST['nazwisko'])`.
- Jeśli dane zostały przesłane, są one zabezpieczane przed atakami XSS poprzez funkcję `htmlspecialchars()` i przypisywane do zmiennych `$imie` i `$nazwisko`.
- Jeśli dane nie zostały przesłane, zmienne `$imie` i `$nazwisko` są ustawiane na puste ciągi znaków.



```php
if (isset($_COOKIE['nazwiskocookie']))
    $nazwiskocookie = htmlspecialchars($_COOKIE['nazwiskocookie']);
else
    $nazwiskocookie = '';
if (($nazwisko) && ($imie)) { // są wpisane wartości w formularzu
    if (isset($_POST['pamietac'])) {
        setcookie("imiecookie", "$imie", time() + 25920000);
        setcookie("nazwiskocookie", "$nazwisko", time() + 25920000);
    } else {
        setcookie("imiecookie", "");
        setcookie("nazwiskocookie", "");
    }
}
?>
```

Sprawdzanie ciasteczek:

- Program sprawdza, czy ciasteczka z imieniem i nazwiskiem są już ustawione za pomocą `isset($_COOKIE['imiecookie'])` i `isset($_COOKIE['nazwiskocookie'])`.
- Jeśli ciasteczka istnieją, ich wartości są odczytywane i przypisywane do zmiennych `$imiecookie` i `$nazwiskocookie`.
- Jeśli ciasteczka nie istnieją, zmienne `$imiecookie` i `$nazwiskocookie` są ustawiane na puste ciągi znaków.
- Jeśli użytkownik wprowadził zarówno imię, jak i nazwisko, program sprawdza, czy została zaznaczona opcja „Zapamiętać dane?" (checkbox o nazwie "pamietac").
- Jeśli opcja została zaznaczona, program ustawia ciasteczka na podstawie wprowadzonych danych za pomocą funkcji `setcookie()`. Ciasteczka mają ważność 25920000 sekund (około 300 dni).
- Jeśli opcja nie została zaznaczona, program usuwa ciasteczka ustawiając je z pustymi wartościami.



```
<body>
        <?php
        if (($nazwisko) && ($imie)) { // są wpisane wartości w formularzu
            echo "Wpisana wartość to <b>$imie $nazwisko</b>.<br />";
            echo '<a href="w06p01.php">Powrót do formularza</a>';
        } else { // nie ma wpisanych danych, wyświetlasz formularz
            echo '<form action="w06p01.php" method="post">';
            echo "<label for=\"imie\">Imie: </label>";
            echo "<input type=\"text\" name=\"imie\" value=\"$imiecookie\" />";
            echo "<label for=\"nazwisko\"> Nazwisko: </label>";
            echo "<input type=\"text\" name=\"nazwisko\" value=\"$nazwiskocookie\" />";
            echo '<br><br>';
            echo 'Zapamiętać dane? <input type="checkbox"';
            echo 'name="pamietac" /><br />';
            echo '<br /><input type="submit" value="Wyślij" />';
            echo '</form>';
        }
        ?>
</body>
```

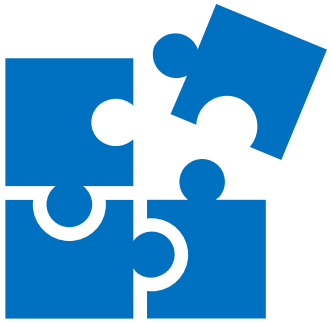
Przykład do wykonania

Wyświetlanie wyników:

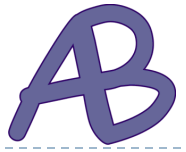
- Jeśli użytkownik wprowadził dane, program wyświetla komunikat z wprowadzonym imieniem i nazwiskiem oraz udostępnia link do powrotu do formularza.
- Jeśli użytkownik jeszcze nie wprowadził danych, program wyświetla formularz z polami na nazwisko i imię. Wartości pól są uzupełniane danymi z ciasteczek, jeśli istnieją.
- Dodatkowo, program umożliwia użytkownikowi zaznaczenie opcji ***„Zapamiętać dane?"*** za pomocą checkboxa.

Ciasteczka są używane tu do zapamiętania imienia i nazwiska użytkownika, co pozwala na personalizację strony nawet po odświeżeniu lub ponownym wejściu na nią.

Przykład do wykonania

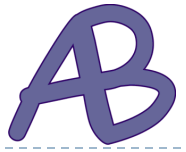


Program jest systemem wyświetlania banerów reklamowych. System ten stara się unikać sytuacji, w której użytkownikowi wyświetlany jest ten sam baner reklamowy, dopóki nie obejrzy przynajmniej raz wszystkich dostępnych banerów.



Przykład do wykonania

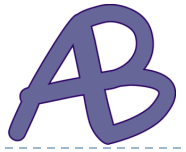
```php
<?php
   $znaleziono = false;
   for ($i=1; $i<=4; $i++) { // sprawdzasz, które bannery już wyświetlano
      if(!isSet($_COOKIE['banner'.$i])){
        $tab[] = $i; $znaleziono = true;
      }
   }
   // jeżeli nie wyświetlano, włączasz do tabeli wszystkie.
   if (!($znaleziono)) { for ($i=1; $i<=4; $i++) { $tab[$i-1] = $i; }}
   $nr= $tab[rand()%count($tab)];
   setcookie ("banner".$nr, "1", time()+86400);
?>
```



Przykład do wykonania

```
<body>
    <div>
        <?php
        echo "<img src=\"banery/$nr.jpg\" alt=\"baner reklamowy\" />";
        ?>
    </div>
</body>
```

Należy przygotować folder „banery” zawierający 4 pliki jpeg o nazwach: „1.jpg” – „4.jpg”



Przykład do wykonania

1. Sprawdzanie, które bannery już były wyświetlane:

- Program używa pętli `for`, aby przejrzeć numery od 1 do 4 (założenie, że istnieje pięć różnych banerów).
- Dla każdego numeru banneru sprawdza, czy ciasteczko o nazwie `'banner'.$i` istnieje.
- Jeśli ciasteczko nie istnieje, dodaje numer banneru do tablicy `$tab` i ustawia zmienną `$znaleziono` na `true`.

2. Jeżeli nie wyświetlano jeszcze żadnego bannera:

- Jeżeli zmienna `$znaleziono` jest `false`, oznacza to, że wszystkie bannery zostały już wyświetlone przynajmniej raz.
- W takim przypadku program dodaje wszystkie numery bannerów do tablicy `$tab`.

3. Losowe wybranie numeru banneru:

- Program losuje numer banneru z tablicy `$tab` przy użyciu funkcji `rand()`.
- Numer ten jest przypisywany do zmiennej `$nr`.

Przykład do wykonania

4. Ustawienie ciasteczka dla wybranego banneru:

- Program ustawia ciasteczko o nazwie `` 'banner'.$nr` `` na wartość `` '1'` `` (oznaczającą, że banner o numerze `$nr` został już wyświetlony).

- Ciasteczko ma czas wygaśnięcia ustawiony na 86400 sekund (1 dzień) od aktualnego czasu.

5. Wyświetlenie losowego bannera:

- Program używa tagu `<img>` do wyświetlenia obrazu (bannera) z katalogu "banery" o nazwie `$nr.gif`.

Bezpieczeństwo danych

Ciasteczka - bezpieczeństwo danych

Częste ataki wykorzystujące ciasteczka:

1. **Session Hijacking:**
   - **Atak:** Atakujący przechwytuje sesję użytkownika, kradnąc ciasteczko sesji.
   - **Obrona:** Używaj sesji zabezpieczonych, stosuj SSL/TLS, unikaj przesyłania wrażliwych danych w ciasteczkach.

2. **Cross-Site Scripting (XSS):**
   - **Atak:** Atakujący wstrzykuje złośliwy skrypt do treści ciasteczka, który jest później wykonany przez przeglądarkę innego użytkownika.
   - **Obrona**: Używaj HttpOnly flag w ciasteczkach, stosuj `htmlspecialchars()` podczas wyświetlania danych od użytkownika.

3. **Cross-Site Request Forgery (CSRF):**
   - **Atak:** Atakujący wykorzystuje sesję użytkownika do złośliwego wykonania akcji na innej stronie.
   - **Obrona:** Używaj unikalnych tokenów CSRF w formularzach, sprawdzaj referencje pochodzenia (Origin) zapytań HTTP.

Ciasteczka - bezpieczeństwo danych

1. **Cookie Poisoning:**
   - **Atak**: Atakujący modyfikuje zawartość ciasteczek, próbując wprowadzić błędne lub złośliwe dane.
   - **Obrona**: Stosuj sprawdzenie integralności danych w ciasteczkach, np. podpis cyfrowy.

2. **Cookie Theft:**
   - **Atak:** Atakujący kradnie ciasteczka z danymi sesji.
   - **Obrona:** Stosuj zabezpieczony protokół komunikacyjny (SSL/TLS), unikaj przesyłania ciasteczek przez niezabezpieczone połączenia.

3. **Session Fixation:**
   - **Atak:** Atakujący ustawia własne ciasteczko sesji na urządzeniu użytkownika.
   - **Obrona:** Generuj nową sesję po zalogowaniu, używaj unikalnych identyfikatorów sesji.

4. **Side Channel Attacks:**
   - **Atak:** Atakujący wykorzystuje informacje zawarte w ciasteczkach do uzyskania wrażliwych danych.
   - **Obrona:** Nie przechowuj wrażliwych danych w ciasteczkach, używaj bezpiecznych algorytmów kryptograficznych.

Ciasteczka - bezpieczeństwo danych

Podstawowe środki obronne obejmują:

- **SSL/TLS:** Używaj zabezpieczonego połączenia, szczególnie gdy przesyłasz wrażliwe dane.
- **HttpOnly i Secure Flags:** Używaj HttpOnly, aby zablokować dostęp do ciasteczek z poziomu skryptów JavaScript, oraz Secure, aby ograniczyć przesyłanie ciasteczek jedynie przez bezpieczne połączenia (HTTPS).
- **Token CSRF:** Dodawaj do formularzy unikalne tokeny CSRF, aby zabezpieczyć się przed atakami CSRF.
- **Sprawdzanie referencji pochodzenia (Origin):** Weryfikuj referencje pochodzenia w zapytaniach HTTP, aby chronić się przed CSRF.
- **Szyfrowanie Ciasteczek**: W przypadku przechowywania wrażliwych danych w ciasteczkach, stosuj odpowiednie szyfrowanie i podpisy cyfrowe.
- **Krótki Czas Wygaśnięcia:** Ustaw krótki czas wygaśnięcia dla ciasteczek, zwłaszcza tych przechowujących dane sesji.

Warto pamiętać, że bezpieczeństwo aplikacji internetowej to kompleksowe zadanie, które wymaga uwzględnienia wielu aspektów, takich jak konfiguracja serwera, obsługa sesji, bezpieczne praktyki kodowania, oraz monitoring i reagowanie na ewentualne incydenty.

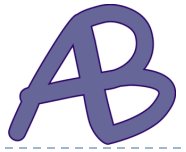
Ciasteczka - bezpieczeństwo danych

Funkcja **`htmlspecialchars()`** w PHP jest używana do zabezpieczania treści przed atakami typu **Cross-Site Scripting (XSS),** który polega na wstrzykiwaniu złośliwego kodu (najczęściej JavaScript) do treści strony, co może prowadzić do różnych zagrożeń, takich jak kradzież sesji użytkownika czy manipulacja treścią strony.

Działanie `htmlspecialchars()` polega na zamianie pewnych znaków specjalnych na ich encje HTML, co powoduje, że przeglądarka traktuje te znaki dosłownie jako tekst, a nie jako znaczniki HTML lub instrukcje JavaScript.

1. Znak & zamieniany jest na &.
2. Znak < zamieniany jest na <.
3. Znak > zamieniany jest na >.
4. Znak " zamieniany jest na ".
5. Znak ' zamieniany jest na '.

Dzięki tej zamianie, nawet jeśli dane pochodzące od użytkownika zawierają znaki specjalne, zostaną one zamienione na encje HTML i wyświetlone jako zwykły tekst, a nie jako kody lub instrukcje do wykonania. To zabezpiecza stronę przed potencjalnymi atakami XSS.



Literatura

W prezentacji użyto przykładów z książki:
- Żygłowicz Jerzy - PHP - Kompendium wiedzy, Helion

- https://www.php.net