Mechanizm sesji

Ciasteczka (*ang. cookies*) omówione przez nas w kursie pozwalają przechowywać dane na dysku klienta, dzięki bazie SQLite obsługiwanej przez przeglądarkę na komputerze internauty. Lecz co jeśli chcemy zapamiętać dane użytkownika na czas jego wizyty, nawet jeśli przechodzi między różnymi podstronami naszego serwisu? (np. sklepu internetowego lub innej aplikacji webowej umożliwiającej nam zalogowanie się). Wtedy do akcji wkracza tzw. sesja, realizowana już po stronie serwera, a nie klienta. Co przede wszystkim umożliwia nam sesja:

- Przechowywanie danych użytkownika (np. login, imię, wartości liczbowe, preferencje konfiguracyjne, inne zmienne potrzebne do działania skryptów).
- Tworzenie systemu logowania – czyli aplikacji internetowych, które już nie tylko statycznie prezentują taką samą zawartość dla wszystkich internautów, tylko pozwalają nam na posiadanie spersonalizowanego, osobistego konta w usłudze.
- Pamiętanie zawartości koszyka zakupowego w sklepie internetowym.

Sesja to zatem **mechanizm przechowywania danych po stronie serwera**, który działa do momentu wygaśnięcia sesji (jak długo to trwa zależy od konfiguracji po stronie serwera) albo do momentu jej zniszczenia (np. skryptem realizującym proces wylogowania):

- Użytkownik odwiedza stronę – PHP tworzy identyfikator sesji, tzw. `session ID`.
- Dane sesji są przechowywane na serwerze, a przeglądarka dostaje tylko identyfikator `PHPSESSID`.
- Gdy użytkownik odwiedza kolejne strony, PHP używa identyfikatora, aby w razie potrzeby uzyskać dostęp do danych zgromadzonych w sesji.

Jest to oczywiście mechanika z natury bezpieczniejsza niż ciasteczka, ponieważ dane nie są przechowywane po stronie klienta, a po stronie serwera.

Jak rozpoczynamy używanie sesji w skrypcie PHP?

Aby z sesji skorzystać trzeba nam rozpocząć od uruchomienia użycia sesji w skrypcie za pomocą prostej, gotowej funkcji:

```
session_start();
```

Musimy to zrobić **na samym początku pliku**, jeszcze przed jakimkolwiek kodem HTML (nawet przed pustym znakiem nowej linii), ponieważ PHP musi wysłać nagłówki HTTP. Czyli przykładowo:

```
<?php
  session_start();
?>
<!DOCTYPE html>
<html lang="pl">
<head>
    <meta charset="UTF-8">
    <title>Sesja w PHP</title>
</head>
<body>
    Tutaj właściwa zawartość podstrony...
</body>
</html>
```

Dopiero teraz możemy w ogóle skorzystać z sesji w skrypcie PHP – ponieważ nie wszystkie skrypty w witrynie będą z niej korzystać (a może nawet większość nie będzie), to właśnie w taki właśnie sposób niejako “oznaczamy” te pliki PHP, które będą sesji używać.

Tworzenie zmiennych sesyjnych

Zmienną sesyjną tworzymy, przypisując jej wartość do globalnej tablicy

`$_SESSION`:

```
<?php
session_start();
$_SESSION["imie"] = "Jolka";
?>
<p>Zmienna sesyjna została zapisana!</p>
```

Zmienna

`$_SESSION["imie"]` zostaje stworzona w aktualnej sesji użytkownika. Dane będą przechowywane na dysku serwera – lecz właśnie! – opowiedzmy teraz gdzie dokładnie można je podczas egzaminu INF.03 odnaleźć w XAMPP.

Gdzie XAMPP przechowuje dane sesji?

Gdy tworzymy zmienne sesyjne w PHP, to wartości te zostaną zapisane w specjalnym pliku tymczasowym na dysku. W przypadku lokalnego serwera XAMPP w Windows, pliki te są zapisywane w katalogu

`C:\xampp\tmp` (*ang. temporary – pliki tymczasowe*). Każda nowa sesja generuje osobny plik, który rozpoczyna się od przedrostka `sess_`, np.:

```
sess_4vgr81bojd5kjdqclcth7n52q5
```

Wewnątrz tego pliku znajdują się zakodowane (jak to mówimy: serializowane) dane sesyjne, które możemy odczytać np. za pomocą funkcji

`session_decode()` lub ręcznie z pomocą edytora tekstu – choć najczęściej nie ma potrzeby tego robić.

Ważne: aby sesje działały poprawnie, folder

`C:\xampp\tmp` musi istnieć i mieć ustawione prawa do zapisu. Jeśli sesje nie działają prawidłowo (np. nie są zapamiętywane dane), warto sprawdzić ten folder oraz konfigurację w pliku `php.ini` – linia:

```
session.save_path = "C:\xampp\tmp"
```

Dzięki temu wiemy, gdzie fizycznie PHP “chowa” nasze dane sesyjne na serwerze w czasie działania aplikacji – a to przydaje się podczas debugowania problemów z sesjami.

Odczytywanie zmiennych sesyjnych

Aby odczytać dane sesyjne, wystarczy użyć zmiennej

`$_SESSION`, ale trzeba nam sprawdzić najpierw funkcją `isset()` w ramach instrukcji warunkowej `if` czy taka wartość w tablicy sesyjnej istnieje:

```
<?php
session_start();
if (isset($_SESSION["imie"])) {
```

```
    echo "<p>Witaj ponownie, " . $_SESSION["imie"] . "!</p>";
} else {
    echo "<p>Nie znaleziono zmiennej sesyjnej.</p>";
}
?>
```

Usuwanie zmiennych sesyjnych lub zakończenie sesji

Różnica jakościowa obu tych działań jest oczywista – albo kasujemy wybraną zmienną z sesji, albo niszczymy całą sesję.

Usunięcie konkretnej zmiennej sesyjnej:

```
<?php
session_start();
unset($_SESSION["imie"]);
?>
<p>Zmienna sesyjna została usunięta!</p>
```

Usunięcie wszystkich zmiennych i zakończenie sesji

```
<?php
session_start();
session_destroy();
?>
<p>Sesja została zakończona!</p>
```

Warto wiedzieć, że

`session_destroy()` nie usuwa natychmiast danych z `$_SESSION`. Aby w pełni “wyczyścić” dane sesyjne z pamięci bieżącego skryptu, warto dodatkowo użyć `$_SESSION = [];`. Takie zniszczenie sesji przydaje się np. w skrypcie wylogowania użytkownika.

Praktyczny przykład użycia sesji – w jednym pliku

Zrealizujmy teraz “zbiorczy” przykład użycia sesji, czyli jeden plik PHP, który:

- uruchamia użycie sesji,
- pozwala zapisać zmienną sesyjną na podstawie danych z formularza,
- wyświetla spersonalizowany komunikat powitalny,
- umożliwia zakończenie sesji poprzez przycisk „Wyloguj”.

Cały kod podstrony prezentuje się następująco:

```
<?php
session_start();
if (isset($_POST["zapisz"])) {
    $_SESSION["imie"] = $_POST["imie"];
}
if (isset($_POST["wyloguj"])) {
    session_destroy();
    header("Location: " . $_SERVER["PHP_SELF"]);
```

```
    exit();
}
?>
<!DOCTYPE html>
<html lang="pl">
<head>
    <meta charset="UTF-8">
    <title>Obsługa sesji w PHP</title>
</head>
<body>
    <h2>System powitań</h2>
    <?php
    if (isset($_SESSION["imie"])) {
        echo "<p>Witaj ponownie, " . $_SESSION["imie"] . "!</p>";
        echo '<form method="post"><button type="submit"
name="wyloguj">Wyloguj</button></form>';
    } else {
    ?>
        <form method="post">
            <input type="text" name="imie" required>
            <button type="submit" name="zapisz">Zapisz imię</button>
        </form>
    <?php
    }
    ?>
</body>
</html>
```

Omówmy teraz poszczególne fragmenty tego kodu źródłowego.

1. Rozpoczęcie sesji

```
<?php
session_start();
?>
```

To polecenie **obowiązkowo** umieszczamy na samym początku pliku – przed jakimkolwiek kodem HTML czy

`echo`. Dzięki niemu mechanizm sesji jest aktywowany i możemy już używać globalnej tablicy `$_SESSION`.

2. Obsługa formularza zapisu imienia

```
if (isset($_POST["zapisz"])) {
    $_SESSION["imie"] = $_POST["imie"];
}
```

Jeśli użytkownik przesłał formularz z przyciskiem

`name="zapisz"`, to zapisujemy do sesji wartość z pola tekstowego. Zmienna `$_SESSION["imie"]` zapamiętuje imię użytkownika aż do momentu jego “wylogowania” lub zamknięcia przeglądarki (chyba że sesja wygaśnie wcześniej).

3. Obsługa formularza wylogowania

```php
if (isset($_POST["wyloguj"])) {
    session_destroy();
    header("Location: " . $_SERVER["PHP_SELF"]);
    exit();
}
```

Po kliknięciu przycisku “Wyloguj” zniszczymy całą sesję – wszystkie dane zapisane wcześniej w

`$_SESSION` zostają skasowane!

Użycie funkcji

`header()` przekierowuje nas od razu do tego samego pliku, dzięki czemu nie zobaczymy już danych sesyjnych (a zamiast tego pojawi się formularz).

Uwaga techniczna: wywołanie

`exit()` po `header()` jest bardzo dobrą praktyką – zapobiega dalszemu wykonywaniu kodu poniżej.

4. Logika wyświetlania treści HTML

```php
if (isset($_SESSION["imie"])) {
    echo "<p>Witaj ponownie, " . $_SESSION["imie"] . "!</p>";
    echo '<form method="post"><button type="submit" name="wyloguj">Wyloguj</button>
</form>';
} else {
    // formularz logowania
}
```

Tutaj widzimy tzw. warunkowe renderowanie zawartości strony – w zależności od tego, czy zmienna sesyjna

`$_SESSION["imie"]` istnieje. Jeśli tak, pokazujemy spersonalizowane powitanie oraz przycisk “Wyloguj”. Jeśli nie – formularz logowania.

Ten kod dobrze obrazuje też zasadę działania HTTP: każdorazowe odświeżenie strony to osobne żądanie – a mechanizm sesji pozwala mimo tego “zapamiętać” dane między tymi żądaniami (a nawet ewentualnie przekierowaniami), dzięki tablicy globalnej

`$_SESSION`.

Podsumowanie

- Sesja to mechanizm przechowywania danych po stronie serwera – działa do momentu wygaśnięcia lub celowego zakończenia.
- Użytkownik otrzymuje tylko identyfikator sesji (PHPSESSID), natomiast dane sesyjne przechowywane są na serwerze – w XAMPP jest to domyślnie lokalizacja: `C:\xampp\tmp`.
- Aby rozpocząć sesję, należy użyć `session_start();` – zawsze przed jakimkolwiek kodem HTML.
- Do zapisu danych sesyjnych służy superglobalna tablica `$_SESSION` – np. `$_SESSION["klucz"] = "wartość";`.
- Aby odczytać zmienną sesyjną, używamy instrukcji warunkowej `isset()`, np. `isset($_SESSION["imie"])`.
- Usunięcie zmiennej sesyjnej wykonuje się za pomocą `unset($_SESSION["klucz"]);`, natomiast zakończenie całej sesji – poprzez `session_destroy();`.

- Najlepszą praktyką jest odświeżenie strony po zapisaniu lub zniszczeniu sesji, przy użyciu `header("Location: ...");` oraz `exit();`.
- Sesje świetnie nadają się do logowania użytkowników, personalizacji interfejsu czy zapamiętywania stanu koszyka w sklepie internetowym.