20. Session v PHP

Protokol HTTP, který slouží ke komunikaci mezi www serverem a prohlížečem je **bezstavový**. Tzn., že mezi jednotlivými přechody stránek **se neudržuje žádné spojení.** Když kliknete na odkaz, pouze se spojí klient se serverem, server pošle stránku a spojení se ukončí. Pokud ale potřebujete znát obsah hodnoty proměnné, kterou uživatel odeslal formulářem asi tak o 3 stránky dříve, bez session se neobejdeme.

Jaká je hlavní činnost session:

1. **identifikaci uživatele**
2. **uchovávání obsahu proměnných**

Pro identifikaci uživatele stačí správně identifikovat opětovné přístupy od téhož uživatele. Samotná data přiřazená ke konkrétnímu uživateli již není třeba mezi stránkami sdílet, ale lze je ukládat na server a odkazovat se na ně.
Na tomto principu jsou založeny sessions. Při jejich vytváření je uživateli vygenerován unikátní identifikátor sloužící k jeho rozpoznání a umožňující přistupovat ke konkrétnímu datovému souboru. Pro sdílení mezi jednotlivými stránkami tento identifikátor uložíme do cookie, kterou prohlížeč na server odesílá při každém načítání stránky. Data samotná jsou defaultně uložena v serializované podobě v souboru uloženém na serveru a pojmenovaném podle uvedeného identifikátoru.

Příklad http requestu (vyžádání nějaké URL adresy):

```
GET /wiki/Wikipedie HTTP/1.1
Host: cs.wikipedia.org
Accept-Charset: UTF-8,*
```

Zde vidíme zcela běžnou obecnou hlavičku, kde není spuštěn příkaz (funkce session) a server odpovídá anonymnímu uživateli.

```
GET /programujeme-v-php/sessions?rev=1 HTTP/1.1
Host: pehapko.cz
Accept-Encoding: gzip,deflate,sdch
Accept-Language: en,cs;q=0.8
Cookie: PHPSESSID=9g0jg1mgs26c1i0e4hok7rgbd1; nette-browser=spx2kdfq2y
```

V druhém případě již hlavička obsahuje záznamy o cookies, vidíme tady hned dvě. Cookie s název PHPSESSID je klíč k identifikaci uživatele. Cookie nette-browser není tak důležitý. Identifikuje prohlížeč.

Pokud není na serveru nastavený session.auto_start (což ve výchozím stavu není), zahájíme session relaci funkcí **session_start()**. Kvůli nastavování cookie musíme tuto funkci volat před jakýmkoliv výstupem, typicky zcela na začátku souboru (až na výjimky…). V této chvíli již můžeme začít pracovat se superglobálním polem $_SESSION (dříve $HTTP_SESSION_VARS).

Hodnotu do session přiřadíme běžným operátorem přiřazení $_SESSION[‚time‘] = time(). Přečíst (s vypsáním) ji můžeme standardní konstrukcí echo $_SESSION[‚time‘]. Obecně lze říci, že se session pracujeme jako s běžným polem, můžeme používat in_array, isset, unset a podobně všechny funkce pro práci s poli, jak byly uvedeny v díle o polích. Jediný rozdíl při práci mezi session a běžnými poli je jejich superglobálnost, tj. **jsou viditelé v libovolné funkci**.
Pozn. Na rozdíl od nastavení cookie pomocí setcookie() se změna session projeví ještě na téže stránce (změněná cookie až po reloadu, protože obvykle neupravujeme přímo $_COOKIE).

Příklad základní práce se session – čas strávený na stránce a další statistiky

```
<?php
session_start();

$firstVisit = false;
if (!isset($_SESSION['firstVisit']))
{
    $_SESSION['firstVisit'] = new \Datetime;
    $_SESSION['lastVisit'] = new \Datetime;
    $_SESSION['count'] = 0;
    $firstVisit = true;
}
$timeSum = (new \Datetime)->diff($_SESSION['firstVisit']);
$timeLast = (new \Datetime)->diff($_SESSION['lastVisit']);
$lastVisit = $_SESSION['lastVisit'];
$_SESSION['lastVisit'] = new \Datetime;
$_SESSION['count']++;

echo ($firstVisit ? 'Vítejte, jste u nás poprvé.' : 'Děkujeme, že se k nám vracíte.') . PHP_EOL;
echo '<br>Čas první návštěvy: ' . $_SESSION['firstVisit']->format('H:i:s - d.m.Y') . PHP_EOL;
echo '<br>Celkový čas strávený na našich stránkách: '. $timeSum->format('%H:%i:%s') . PHP_EOL;
echo '<br>Čas předchozí návštěvy: ' . $lastVisit->format('H:i:s') . PHP_EOL;
echo '<br>Čas od předchozí návštěvy: ' . $timeLast->format('%H:%i:%s') . PHP_EOL;
echo '<br>Počet návštěv: ' . $_SESSION['count'] . PHP_EOL;
?>
```

Vyzkoušejte si tento příklad takto:

- Aktualizacemi téže stránky se budou měnit časové údaje. Aplikace si pamatuje svoji historii.
- Otevřete si stránku ve dvou různých prohlížečích (nebo v jednom + anonymním režimu) a paralelně jednotlivé stránky aktualizujte. Ověřte, že se jednotlivé relace neovlivňuji.
- Naopak otevřete stránku ve dvou běžných oknech prohlížeče (sdílejících cookies). Zjistíte, že nyní se relace vzájemně ovlivňují.

- Můžete vytvořit několik (volitelně odkazy provázaných) stránek a includovat do nich výše uvedený kód. Zjistíte, že se session sdílí mezi stránkami (session_start() musí být na každé stránce).
- Můžete si měnit jednotlivé cookie, když je smažete, bude vaše návštěva označena jako první. Zakážete-li ve svém prohlížeči přijímat cookies, pokaždé bude návštěva označena jako první.
- Výchozí nastavení pamatování si session bývá 24 minut (1440 sekund), nechte relaci delší dobu neaktivní a pak zkuste aktualizovat. Zřejmě uvidíte, že se v relaci pokračuje a nic se nestalo, mazání starých záznamů je totiž pravděpodobnostní. Existence po dobu 24 minut od poslední aktualizace je zaručena. Smazání po této době nikoliv, to závisí na aktivitě garbage collectoru, která je při výchozím nastavení 0.01, tj. s pravděpodobností 1 % při každé inicializaci smaže staré session záznamy. **Na některých systémech (Debian, Ubuntu) bývá gc zcela neaktivní a mazání periodicky provádí cronová úloha.

Demo k testování.

Zdroj: http://www.pehapko.cz/programujeme-v-php/sessions