Cookies und Sessions zur temporären Datenspeicherung

HTTP ist ein zustandsloses Protokoll, daher sind für den Webserver die Seitenaufrufe unabhängig voneinander. Eine Webanwendung, deren Interaktion mit dem Benutzer über mehrere Seitenaufrufe andauert, muss mit Tricks arbeiten, um den Teilnehmer über mehrere Zugriffe hinweg identifizieren zu können. Dieses wird durch Speicherung temporärer Daten, entweder clientseitig mittels **Cookies** oder serverseitig durch **Sessions**, realisiert. Der Zugriff auf diese Informationen erfolgt über die entsprechenden php-Global-Arrays **`$_COOKIE`** bzw. **`$_SESSION`**.

Wichtig : Sowohl Cookie- als auch Session-Funktionen sind Bestandteile eines HTTP-Headers und müssen aufgerufen werden, noch bevor irgendeine andere Ausgabe an den Browser erfolgt.

Beispiel Onlineshop: Beim Bezahlvorgang müssen Kundendaten und alle Artikel aus dem Warenkorb bekannt (oder zwischengespeichert) sein, damit der Vorgang korrekt abgeschlossen werden kann.

Variante 1 – COOKIE : Serverseitig wird eine php-Anweisung zur Speicherung von Cookies ausgeführt. Diese wird im Header des folgenden Responce an den Client gesendet und führt dort zur Speicherung innerhalb einer geschützten Browserumgebung. Mit dem nächsten Request an diese Domain sendet der Browser alle gültigen Cookies an den Server (daher stehen Cookies auch erst in der nächsten Seite zur Verfügung).

Funktion	Beschreibung
`bool setcookie ( string $name [, string $value [, int $expire = 0 [, string $domain [, bool $secure = false [, bool $httponly = false ]]]]] );`	Name des Cookies Wert des Cookies Zeitpunkt, an dem das Cookie ungültig wird (mit **time()**+60*60*24*30 wird das Cookie in 30 Tagen ablaufen) Domain, für die das Cookie zur Verfügung steht **true** : dass Cookie wird nur über eine HTTPS-Verbindung übertragen **true** : auf das Cookie ist nur via HTTP-Protokoll zugreifbar
`$value = $_COOKIE["name"] ;`	liest den Wert des Cookies **name** aus

Cookies können clientseitig deaktiviert werden und stehen deshalb nicht zuverlässig zur Verfügung. Ein einfacher Test wird durchgeführt, indem der Server probiert ein Cookie zu speichern und im nächsten Aufruf zu lesen.

Variante 2 – SESSION : Bei einer Session (Sitzung) werden alle Daten auf dem Server gespeichert. Zur eindeutigen Identifikation wird eine **`SESSION-ID`** verwendet, die dem Client beim Start einer Session übermittelt wird. Für den Zugriff auf diese Daten muss sich der Client mit dieser **`SESSION-ID`** beim Server identifizieren. Diese **`SESSION-ID`** wird entweder als Cookie gespeichert oder mit an die URL gehängt. Das Session-Handling (Übertragung und Speicherung der **`SESSION-ID`**)erfolgt in php weitgehend automatisch.

Funktion	Beschreibung

`bool session_start ();`
`bool session_destroy ();`
`string session_id ([ $id ]);`
	Erzeugt eine neue Session oder setzt eine vorhandene fort.
	Löscht alle in einer Session registrierten Daten.
	Liefert und/oder setzt die aktuelle Session-ID. Alternativ kann die PHP-Konstante **SID** auch dazu verwendet werden.

`$_SESSION["name1"] = $wert ;` `$value2 = $_SESSION["name2"] ;`	Registriert eine Session-Variable **name1** mit dem Inhalt aus **$wert**. Liest den Inhalt der registrierten Session-Variable **name2**.

Beispielablauf zur Personalisierung (hier Name und Anrede) einer WEB-Site mit **Cookies**

index.php

Client fordert WEB-Site zur Eingabe von Daten an. ①

```
   :
<form action="page1.php">
   <input name="user">
   :
</form>
   :
```

Name [ ]
senden
index.php

index.php

Name Udo
senden
index.php

② Server sendet Formular mit Text-Eingabefeld zur Datenübertragung.

page1.php?user=Udo

Eingabe des User-Namen und Übertragung per GET. ③

```
<?php
$N = $_GET["user"];
setcookie("name",$N);
   :
echo "Guten Tag $N !";
   :
```

Guten Tag **Udo !**
Herr ○ Frau ○
senden
page1.php

SET COOKIE : name=Udo
page1.php

④ Auslesen des $_GET-Arrays, Anweisung zur Speicherung eines Cookies.

Guten Tag **Udo !**
Herr ◉ Frau ○
senden
page1.php

SEND COOKIES : name=Udo
page2.php?anrede=Herr

1. Cookie wird gespeichert und an den Server gesendet. ⑤

name=Udo

```
<?php
$A = $_GET["anrede"];
setcookie("anrede",$A);
$N = $_COOKIE["name"];
   :
echo "Guten Tag $A $N !";
   :
```

Guten Tag
Herr Udo !
page2.php

SET COOKIE : anrede=Herr
page2.php

⑥ Erst jetzt steht das Cookie dem Server zur Verfügung und kann ausgelesen werden.

Guten Tag
Herr Udo !
page2.php

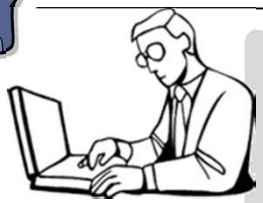
SEND COOKIES : name=Udo/anrede=Herr
weiter.php

anrede=Herr

name=Udo

Cookies und Sessions zur temporären Datenspeicherung

Beispielablauf zur Personalisierung (hier Name und Anrede) einer WEB-Site mit einer **Session**



index.php

```
<?php
session_start();
<form action="page1.php">
   <input name="user">
   :
</form>
   :
```

Name [ ]
senden
index.php

SESSION-ID	34a9...

Client fordert WEB-Site zur Eingabe von Daten an. ①

SET COOKIE : PHPSESSID=34a9....
index.php

Name Paul
senden
index.php

② Server beginnt eine SESSION und sendet eine **SESSION-ID** die clientseitig als Cookie gespeichert werden soll.

SEND COOKIES : PHPSESSID=34a9....
page1.php?user=Paul

```
<?php
$N = $_GET["user"];
session_start();
$_SESSION["name"]=$N;
echo "Hallo";
echo $_SESSION["name"];
```

Hallo **Paul**
Herr ○ Frau ○
senden
page1.php

Cookie wird geschrieben und nun immer wieder gesendet. PHPSESSID =34a9.... ③

page1.php

SESSION-ID	34a9...
name	Paul

④ Mit **session_start()** und dem empfangenen Cookie wird die Sitzung weiter geführt. Die Variable **name** wird registriert.

Hallo **Paul**
Herr ◉ Frau ○
senden
page1.php

SEND COOKIES : PHPSESSID=34a9....
page2.php?anrede=Herr

```
<?php
$A = $_GET["anrede"];
session_start();
$_SESSION["anrede"]=$A;
echo $_SESSION["anrede"];
echo $_SESSION["name"];
```

Herr Paul
page2.php

Cookie (PHPSESSID) und Daten werden gesendet. PHPSESSID =34a9.... ⑤

page2.php

SESSION-ID	34a9...
name	Paul
anrede	Herr

⑥ Analog Schritt ④ kann die SESSION beliebig fortgeführt werden.

SESSION-Ende

```
<?php
$_SESSION = array();
session_destroy();
setcookie("PHPSESSID","",0);
   :
```

pageN.php

Zur Beendigung einer Session sollten:
- alle Variablen gelöscht werden
- Daten der Session gelöscht werden
- das Session-Cookie gelöscht werden