Doing Web Apps

Persistente Daten verwalten mit PHP

Autor: Rüdiger Marwein
Letzte Änderung: 2009-04-03
Version: 0.7

Inhaltsverzeichnis

1.Einleitung

Was sind eigentlich persistente Daten? HTTP ist ein Zustandsloses Protokoll. Nachdem eine HTTP-Antwort ausgeliefert wurde, hat der Webserver selbst keine Daten über den Vorgang „im Gedächtnis“. Bei der zweiten Anfrage ist serverseitig nichts über die vorangegangene Abfrage bekannt.

Was aber, wenn wir Informationen beibehalten müssen? Z.B. ob sich ein Benutzer bereits erfolgreich eingeloggt hat. Man benötigt ein Möglichkeit persistente (beständige, dauerhafte) Daten zu verwalten.

Es gibt mehrere Möglichkeiten, sich Dinge über und für einen Benutzer zu merken, während er mehrere Seiten besucht.

2.URI

Man könnte sich vorstellen, dass **jeder Link** mit einem GET-Parameter ausgestattet wird, der Benutzerinformationen oder einen Verweis auf selbige darstellt. Z.B.

```
index.php?eingeloggt=1
```

Damit könnte man sich zumindest mal diesen Status relativ einfach merken. Dass das natürlich wahnsinnig unsicher ist, kann man leicht erkennen.

Die URI könnte aber auch so aussehen:

```
index.php?nutzer_daten=12345
```

Wobei hier 12345 eine generierte Nummer ist und unter dieser Nummer dann serverseitig Daten gespeichert werden.

Das ist schon ein „bisschen“ sicherer, weil die Nummer erraten werden müsste... aber so ganz das gelbe von Ei ist beides noch nicht.

3.Kekse backen

3.1.Cookies mit PHP

Cookies sind Textdaten, die für eine Webseite Informationen speichern können. Dabei wird eine Art GET-Zeichenkette in diese Datei gespeichert. Das Cookie kann (wenn nicht mit Absicht falsch ausgezeichnet) nur von der Seite, die es erstellt hat auch wieder ausgelesen werden. Das Cookie ist also eine relativ sichere Methode um einen Benutzer zu bestimmen.

Wie speichert man in ein Cookie Dinge? Mit PHP geht es folgendermaßen:

```
// setzen
setcookie('eingeloggt',1,time()+65000);
// holen
if( $_COOKIE['eingeloggt'] )==1) echo 'OK';
// löschen
setcookie('eingeloggt');
```

Es ist zu beachten, dass das Cookie mit dem HTML-Header geschickt wird. Das wiederum bedeutet, dass man es so nur setzen kann wenn noch keine Ausgabe mit PHP gemacht wurde. Sonst erhält man folgende Fehlermeldung

```
Warning: Cannot modify header information - headers already sent by ...
```

3.2.Cookies mit JavaScript

Cookies kann man auch mit **JavaScript** setzen und auslesen. Da JavaScript sowieso immer beim Benutzer auf dem Rechner ausgeführt wird (also nachdem Daten geschickt wurden), kann ein Cookie zu jedem beliebigen Zeitpunkt gesetzt werden. Allerdings braucht es zum Auslesen etwas Mühe.

Mit JavaScript gesetzte Cookie kann man auf PHP-Seite auslesen und setzen.

Mit PHP gesetzte Cookies kann man mit JavaScript auslesen und setzen.

```
<script language="JavaScript">
<!--
  // Cookie ohne Verfallsdatum
  document.cookie = 'eingeloggt=1';

  // Dauer = 60 Sekunden
  var ablauf = new Date();
  var eineMinute = ablauf.getTime() + 60000;
  ablauf.setTime(eineMinute);

  document.cookie = 'spass=2; expires='+ablauf;

  // Ausgabe eingeloggt=1; spass=2
  alert( document.cookie );

  alert( getcookie('eingeloggt') );
  alert( getcookie('spass') );

  // löschen
  document.cookie = 'spass=2; expires=0;

  function getcookie(cookie_name) {
    // zerlegen an Stellen mit Strichpunkt
    var teile = document.cookie.split('; ');
    // jeden Treffer durchlaufen
    for(i in teile) {
      // ein Teil am = nochmals teilen
      tmp = teile[i].split('=');
      // 0=key, 1=value auswerten
      if(tmp[0]==cookie_name) {
        // falls gefunden, zurückgeben
        return tmp[1];
```

```
      }
    }
    return false;
  }
//-->
</script>
```

Die Funktion **getcookie** bietet hier die Schnittstelle, um auf die in einer Zeichenkette gespeicherten Cookies einzeln lesend zuzugreifen.

4.Die Session

Eine Session bezeichnet eine Sitzung. Für den Benutzer ist seine Sitzung anhand der Sitzungs-Nummer (Session-ID) entweder in der URL ersichtlich oder bei ihm in einem Cookie gespeichert.

Die Session vereinigt die oben angesprochenen Punkte.

Eine Session in PHP besteht aus 3 Teilen:

GET-Parameter oder Cookie

Hier wird die Session-ID (Standard PHPSESSID) festgehalten.

Datei auf der Festplatte des Servers

Hier werden die mit PHP erzeugten Daten gespeichert.

Die Variable $_SESSION

Alles was hier drin steht wird beim Schließen der Session gespeichert.

Falls es möglich ist die Session-ID in einem Cookie zu speichern, sollte man das tun.

Ein einfaches Beispiel soll verdeutlichen, wie das mit der Session funktioniert:

```
<?php
  // session starten
  session_start();

  // wurde befehl zum zerstören der session benutzt?
  if($_GET['zuruecksetzen']==1) {
    session_destroy();
    exit();
  }

  // initialisiere „wert“ auf 1 oder erhöhe um 1
  if(!isset($_SESSION['wert'])) {
    $_SESSION['wert'] = 1;
  } else {
    $_SESSION['wert']++;
  }

  // „wert“ ausgeben. Erhöht sich bei jedem Seitenaufruf um 1
```

```
    echo $_SESSION['wert'];

    // Sitzungsdaten serverseitig speichern
    session_write_close();
?>
```

In die Session kann beliebig viel und von jedem Typ alles gespeichert werden. Die Daten werden serialisiert (siehe Funktion **serialize**) in die Session-Datei geschrieben und beim erneuten Starten der Session wieder ausgelesen und deserialisiert.

Das Beispiel benutzt die Variable `$_SESSION['wert']` und erhöht sie bei jedem Seitenaufruf um 1.

Will man Objekte in der Session hinterlegen, ist es wichtig, dass die Dateien mit der Klassen-Definition eingelesen wurden, bevor die Session gestartet wurde. Andernfalls erhält man Objekte vom Typ **__PHP_Incomplete_Class**.

4.1.PHP-Config

Die Einstellungen der Session kann man sich mittels der Funktion phpinfo(); ansehen.

Die wichtigsten PHP-Config Einstellungen sind

`session.use_cookies`

Legt fest, dass (falls möglich) Cookies benutzt werden sollen. Ansonsten wird der Weg über die URL gewählt.

`session.use_only_cookies`

Legt fest, dass nur Cookies benutzt werden dürfen (weil sicherer). Akzeptiert ein Client keine Cookies funktioniert die Session nicht.

`session.name`

PHPSESSID ist etwas gewöhnungsbedürftig. Den Firmennamen oder einfach eine schönere Bezeichnung ist manchmal wünschenswert. Z.B. „sid“ ist ganz nett.

`session.gc_maxlifetime`

Regelt die Lebensdauer einer Session in Minuten.

Die Werte kann man via `ini_set` setzen bevor die Session gestartet wird.

4.2.Seiteneffekte / Wichtiges

Sessions sind nicht unbedingt sicher. Vor allem nicht für den Programmierer. Sobald dieses Konstrukt ins Spiel kommt, ist Vorsicht geboten und mit angeblichen Bugs vorsichtig umzugehen. Einige Hinweise:

- Wird die Session-ID an die URL geheftet, so verliert der Benutzer die Session, wenn er sie aus der URL entfernt. Er ist dann bspw. wieder ausgeloggt.
- Schickt ein Benutzer in eingeloggtem Zustand die aktuelle URL (mit Session-ID) an einen Freund, kann es sehr gut sein, dass dieser sofort als der Sender eingeloggt ist. Das ist zumindest der Fall, wenn man sich bspw. von einem Firmennetz aus mit der

gleichen IP-Adresse bewegt.

- Es gibt einen Session-Timeout. Wenn die Session ausläuft, ist der Benutzer wieder ausgeloggt. Die Session verfällt, wenn alle Browser-Fenster geschlossen wurden oder bis zum Timeout keine weitere Anfrage an den Server gestellt wurde.
- Hat ein Benutzer Cookies deaktiviert und die Session ist als use_only_cookie ausgezeichnet, dann kann der Benutzer sich zwar augenscheinlich einloggen, doch sobald er auf der Folgeseite einen Link betätigt, ist er wieder ausgeloggt.
- Wird die Session-ID in einem Cookie gespeichert und man befindet sich an einem öffentlichen Platz (z.B. Internet-Cafe) und vergisst, sich auszuloggen, ist der nächste Besucher der URL an diesem Rechner gleich wieder eingeloggt.
- Werden beispielsweise ganze Datensätze in der Session festgehalten und die Quellen werden verändert, so ändert sich die Session nicht. Meist sind die neuen Daten erst nach dem nächsten einloggen verfügbar. Hier ist also Vorsicht geboten.
- Wird die URL-Variante verwendet fügt PHP automatisch an alle Links in der Webseite die Session-ID an. Wenn PHP ein Formular entdeckt, fügt es die Session-ID als verstecktes Feld mit ein.
- Die PHP-Session-ID ist ein md5-Hash (verschlüsselt).

Ohne Session geht heute nichts mehr. Die anderen Varianten sind wesentlich unsicherer und falls man so etwas selbst entwickeln möchte, braucht man eine Menge Zeit um alle Eventualitäten abzudecken.

