Superglobals und Formulare

mit PHP

Problemstellung

- HTTP ist ein zustandsloses Protokoll
  - Viele Websites möchten jedoch eine Resource über mehrere Seiten hinweg verfolgen können
- Arbeiten über Cookies hat sich als nicht praktikabel erwiesen
  - Daten werden beim Client gespeichert
  - Daten können durch den Client analysiert werden
- PHP implementiert ein Sessionmanagement über eine superglobale Variable `$_SESSION`
- Auch hier kann PHP keinen 100%igen Schutz bieten
  - Sensitive Daten gehören dennoch in die Datenbank
- Die Session ID selbst wird ebenso über Cookies realisiert

Lösung: Superglobals

- Globale Variabeln, die über alle Scripts hinweg gültig sind, somit nicht mit `global` bekannt gemacht werden müssen
- `$_SESSION`
- `$_GET`
- `$_POST`
- `$_COOKIE`
- `$_SERVER`
- `$_FILES`
- weitere
  - `$_REQUEST`
  - `$_ENV`

Session starten

- Eine Session muss explizit gestartet werden
  - session_start() muss auf jeder Seite aufgerufen werden
  - Ansonsten steht das $_SESSION Array nicht zur Verfügung, resp. ist leer oder nicht gesetzt
  - Identifiziert den Client mittels Session ID und weist so die korrekte Session zu
- Eine Session kann explizit zerstört werden
  - session_destroy() zerstört die aktuelle Session des aktiven Clients
  - session_unset() löscht alle Sessionvariabeln
- Nach dem Start steht das Array $_SESSION zur Verfügung

Objekte in Sessions

- Es wird davon abgeraten, ganze Objekte in einer Session zu speichern, folgende Probleme können dadurch entstehen:
  - Performanceprobleme, grosse Objekte persistent über das Filesystem verwaltet werden müssen
  - Ist eine Klasse nicht deklariert, wird das Objekt vom Typ sdtClass erneut erreichbar
  - Das Objekt muss ohnehin serialisiert werden, was für grosse Objekte ebenfalls zu Problemen führen kann
- session_register sollte unter keinen Umständen verwendet werden
- Eine Alternative wäre, das Laden eines Objektes aus der Datenbank und nur die ID in der Session zu speichern
  - Performanceprobleme werden somit auf die Datenbank verlagert

Beispiel I

```
<?php // page1.php
session_start();
echo 'Willkommen auf Seite #1';
$_SESSION['favcolor'] = 'green';
$_SESSION['animal'] = 'cat';
$_SESSION['time'] = time();
echo '<br /><a href="page2.php">page 2</a>';
?>
```

Beispiel I

```
<?php // page2.php
session_start();
echo 'Welcome to page #2<br />';
echo $_SESSION['favcolor']; // green
echo $_SESSION['animal']; // cat
echo date('Y m d H:i:s', $_SESSION['time']);
echo '<br /><a href="page1.php">page 1</a>';
?>
```


Veraltete Sessionfunktionen

- Die meisten veralteten Sessionfunktionen werden aus Sicherheitsgründen nicht ersetzt, sondern deprecated, bspw.
  - session_register()
    - Registriert eine globale Sessionvariable
    - Dies war im Zusammenhang von Objekten in Sessions sehr nützlich
    - This function has been DEPRECATED as of PHP 5.3.0 and REMOVED as of PHP 5.4.0.
- session_unregister()
  - Setzt eine als global registrierte Sessionvariable zurück in eine lokale Sicht
- session_name()
  - Ändert den Namen einer zuvor registrierten Session
- session_is_registered()
  - Prüft, ob eine mit session_register registrierte Variable vorhanden ist

Nützliche Session Funktionen

- session_regenerate_id()
  - Ist sehr nützlich in Bezug auf Sessionsicherheit
  - Sollte in jeder Applikation periodisch aufgerufen werden
- session_id()
  - Kann eine Session ID setzen oder auslesen
  - Für persistente Besucher geeignet, wo die Sessiondaten in einer Datenbank verwaltet werden
- session_write_close()
  - Wird eher selten gebraucht
  - Normalerweise wird die Session nach Beendigung des Skriptes geschrieben
  - Hiermit wird die Session vorzeitig beendet und die Daten geschrieben

$_GET

- Enthält ein assoziatives Array mit den Variabeln, die über die URL Parameter dem Skript übergeben wurden
  - Ist nur für den aktuelle Request, resp. für das aktuelle Skript gültig
- Sollte aus Sicherheitsgründen immer mit htmlspecialchars() auf den Inhalt geprüft werden
- Die $_GET Variable wird automatisch mit urldecode() geparst
- ```
  <?php
  echo 'Hallo ' . htmlspecialchars($_GET["name"]) . '!';
  ?>
  ```
  ```

$_POST

- Enthält ein assoziatives Array mit den Variabeln, die über HTTP-POST-Parameter dem Skript per Request übergeben wurden
  - Ist nur für den aktuellen Request, resp. für das aktuelle Skript gültig
- Sollte aus Sicherheitsgründen immer mit htmlspecialchars() auf den Inhalt geprüft werden
- ```
  <?php
  echo 'Hallo ' . htmlspecialchars($_POST["name"]) . '!';
  ?>
  ```


$_GET vs. $_POST

- GET
  - schickt den Request in der URL (mehr Sichtbarkeit)
  - Request wird auf Webserver geloggt
    - nicht geeignet für sensitive Daten
  - Einfache Manipulation der Werte über URL
  - Datenmenge ist auf 2KB beschränkt (!)
- POST
  - Wird direkt vom Browser an die URL übermittelt
  - Manipulationen ganz wenig schwieriger
  - Kein serverseitiges Logging der Daten
  ```

$_COOKIE

- Enthält ein assoziatives Array mit den Variabeln, die über HTTP-Cookie-Parameter dem Skript übergeben wurden
  - Cookies sind persistent beim Client gespeichert und werden bei Requests vom Client an den Server übertragen
- Bevor ein Cookie verfügbar ist, muss es beim Client gesetzt werden mit setcookie()

```
<?php
setcookie("TestCookie", $value, time()+3600);
echo 'Hallo ' . htmlspecialchars($_COOKIE["TestCookie"]) .
'!';
?>
```


$_SERVER

- Enthält wichtige Informationen über den Applikationsserver
  - Darin enthaltene Daten werden verwendet, um dem Benutzer Konfigurationsarbeit zu ersparen
  - Können jedoch auch sicherheitsrelevante Daten sein wie z.B. ob der Request von über SSL läuft
  - Zugriff ist unkritisch, da der Server dieses Array bei jedem Request zur Verfügung stellt

```
<?php
echo $_SERVER['SERVER_NAME'];
?>
```

Grundelemente von Formularen

- Bisher einzeilige Texteingabefelder verwendet
- Weitere Formularfelder werden in 3 Gruppen unterteilt
  - Textelemente
  - Auswahlelemente
  - Aktionselemente

Textelemente

- einzeilige Texteingabefelder
  - <input name="vorname" type="text" size="30" maxlength="30" value="Michaela">
- Passwortfelder
  - <input name="kennwort" type="password" size="12" maxlength="12">
- mehrzeilige Texteingabe
  - <textarea name="user_eingabe" cols="50" rows="10"></textarea>
- Versteckte Elemente
  - <input type="hidden" name="UserBrowser" value="">

Auswahlelemente

- Verhindern Eingabefehler durch den Benutzer
- Verringern den Prüfaufwand für die Übertragenen Daten
- Sind, falls möglich den mehrzeiligen Textelementen vorzuziehen
- Man unterscheidet bei den Auswahlelementen zwei Gruppen
  - einfache Auswahlelemente
    - Auswahl von einem Eintrag durch Benutzer
  - mehrfache Auswahlelemente
    - Auswahl von mehreren Einträgen durch Benutzer

Einfache Auswahlelemente I

- Radiobutton-Gruppe
  - Eine Gruppe muss immer das gleiche name-Attribut haben
  - Elemente mit checked=”checked” vorbelegen (resp. gem. DOCTYPE)
- <input type="radio" name="Zahlmethode" value="Mastercard"> Mastercard
  <input type="radio" name="Zahlmethode" value="Visa"> Visa
  <input type="radio" name="Zahlmethode" value="AmericanExpress"> American Express

Einfache Auswahlelemente II

- Einfaches Auswahlmenü, erledigt denselben Zweck wie eine Radiobutton-Gruppe
  - Geringer Platzbedarf
- <select name="top5" size="3">
  <option>Heino</option>
  <option>Michael Jackson</option>
  <option>Tom Waits</option>
  <option>Nina Hagen</option>
  <option>Marianne Rosenberg</option>
  </select>

Mehrfache Auswahlelemente I

- Checkboxen können wie Radiobuttons in Gruppen organisiert sein
  - Einzelne Checkbox ist auch möglich
- Das name-Attribut bei Gruppen von Checkboxen immer mit eckiger Klammer
- <input type="checkbox" name="zutat[]" value="salami">
  Salami
  <input type="checkbox" name="zutat[]" value="pilze">
  Pilze
  <input type="checkbox" name="zutat[]" value="kapern">
  Kapern

Mehrfache Auswahlelemente II

- Mehrfaches Auswahlmenü
  - Leider auch in HTML5 kein geeignetes Element dafür
  - Attribut multiple in Bezug auf DOCTYPE beachten
- <select name="top5" size="5" multiple>
  <option>Heino</option>
  <option>Michael Jackson</option>
  <option>Tom Waits</option>
  <option>Nina Hagen</option>
  <option>Marianne Rosenberg</option>
  </select>

Aktionselemente

- Zwei verschiedene Ausprägungen der Buttons
  - type=submit zum Absenden der eingegebenen Formulardaten
  - type=reset zum Zurücksetzen der eingegebenen Formulardaten
- Weitere Möglichkeit sind grafische Buttons
  - <input type="image" src="absende.gif" alt="Absenden">
  - oder Styling mit CSS und Auslösung des Requests mittels JavaScript
- Generische Buttons mit Attribut type=”button”
  - Machen nur Sinn im Zusammenhang mit JavaScript oder Flash
  - <input type="button" name="Text 1" value="Text 1 anzeigen">

HTML5: neue Formularelemente (Achtung IE)

- progress
  Zeigt ein Fortschrittsbalken z.B. für einen Dateiupload
- meter
  Ein Messwert innerhalb eines Bereiches, geeignet für Temperaturen oder Gewichtsangaben
- datalist
  Darstellen von erweiterten Dropdownboxen
- keygen
  Zum erstellen Schlüsselpaaren über den Webbrowser
- output
  Zeigt ein Ergebnis an z.B. einer Berechnung

HTML5: neue Eingabetypen

- Die neuen Eingabetypen werden jeweils mit einem <input> Element verwendet
  - wobei das type-Attribute den jeweiligen Wert annimmt
  - Verwendung der neuen Typen oft mit pattern
- tel (Telefonnummern), search (gleich wie text), url (zur Eingabe einer URL)
- email (Zur Eingabe einer Mailadresse), datetime (Datum und Uhrzeit), date (Datum ohne Zeitzone), month, week (Kalenderwoche mit Jahr), time, datetime-local
- number (zur Eingabe von Nummern), range (Schieberegler), color (Colorwheel)

HTML5: neue Attribute I

- autofocus
  - Fokussiert ein Element beim Seitenaufruf
- placeholder
  - Zeigt einen Alternativtext in einem input Element
- required
  - markiert ein Feld als Pflichtfeld, ggf. Meldung via title Attribut
- autocomplete
  - Hat nichts mit AJAX zu tun, aktiviert oder deaktiviert die Browserunterstützung dafür
- pattern
  - Der Inhalt des Feldes muss dem Musterpattern (REGEX) entsprechen, ggf. title Attribut verwenden

HTML5: neue Attribute II

- novalidate
  - Verhindert die browsergesteuerte Validierung
- formaction
  - Überschreibt das action-Attribut im form Element (input oder button)
- formenctype
  - Überschreibt den Übertragungsmodus im form Element (input oder button)
- formmethod
  - Überschreibt die HTTP Methode zum senden des Formulars (input oder button)
- formtarget
  - Überschreibt das target Attribut im form Element (input oder button)

HTML5: neue Elemente prüfen/CSS

- :invalid {...}
  - Den ungültigen Elementen die entsprechenden Styles zuweisen
  - Wird umgehend geprüft und angepasst, z.B. ein roter Rahmen
- :required {...}
  - Den erforderlichen Elementen die entsprechenden Styles zuweisen
  - Wird umgehend geprüft und angepasst, z.B. ein roter Hintergrund
- oninput="CustomCheckFunction(this)"
  - Aufruf einer JavaScript-Funktion während der Eingabe
  - z.B. überprüfen, ob zweimal dasselbe Passwort angegeben wurde

Weitere Funktionen in Formularen

- Tabulator-Reihenfolge
  - Attribut tabindex="1" verwenden mit aufsteigenden Integerwerten
- Tastaturkürzel
  - Attribut accesskey="b" verwenden in Kombination mit Alt
- Elemente ausgrauen
  - Attribut disabled="disabled" verwenden, DOCTYPE beachten
- Formularelemente können mit CSS gestylet werden
  - Selektoren beachten: z.B. input[type=submit], input.button
  - Einige Elemente sind nicht formatierbar: Fileuploadbutton oder Selectboxbutton - hierfür gibt es Flash/JS Workarounds

Dateiupload I

- Der Formularkopf muss einen angepassten MIME-Typ haben
  - enctype=”multipart/form-data”
- Mit HTML5 können gleichzeitig mehrere Dateien hochgeladen werden (noch nicht in den Empfehlungen)
- Kann einfach missbraucht werden
  - Hochladen von Skripten
  - Captcha verwenden (!)
  - Dateigrösse clientseitig manipulieren
- <form enctype="multipart/form-data" action="_URL_" method="POST">
  Datei wählen: <input name="userfile" type="file">
  <input type="submit" value="Hochladen">
  </form>

Dateiupload II

- Die Superglobale Variable $_FILES wird beim entsprechenden enctype von PHP automatisch gesetzt
- $_FILES['userfile']['name']
  - Der ursprüngliche Name
- $_FILES['userfile']['type']
  - Der Dateityp, resp. MIME-Type, sollte der Browser das korrekt liefern oder der Client nicht gefälscht haben
- $_FILES['userfile']['size']
  - Die Grösse der hochgeladenen Datei
- $_FILES['userfile']['tmp_name']
  - Der temporäre Name
- $_FILES['userfile']['error']
  - Der Fehlercode

Dateiupload III

```
$uploaddir = '/var/www/uploads/';
$uploadfile = $uploaddir.
basename($_FILES['userfile']['name']);
print "<pre>";
if (move_uploaded_file($_FILES['userfile']['tmp_name'],
$uploadfile)) {
   print "File is valid, and was successfully uploaded.";
   print "Here's some more debugging info:\n";
   print_r($_FILES);
} else {
   print "Possible file upload attack! Here's some debugging
info:\n";
   print_r($_FILES);
}
print "</pre>";
```


Literatur

- http://www.php.net/manual/en/reserved.variables.php
- http://www.php.net/manual/en/ref.session.php
- http://de.selfhtml.org/html/formulare/index.htm
- http://html5doctor.com/html5-forms-introduction-and-new-attributes/
- http://php.net/manual/de/features.file-upload.post-method.php