Vorlesungsreihe
Entwicklung webbasierter Anwendungen

PHP - Teil 2 –
Typische Anwendungsmodule

Prof. Dr.-Ing. Thomas Wiedemann
email: wiedem/skuehn@informatik.htw-dresden.de

HOCHSCHULE FÜR TECHNIK UND WIRTSCHAFT DRESDEN (FH)
Fachbereich Informatik/Mathematik

Gliederung

- Zugriff auf gesendete Daten (GET / POST , isset)
- spezielle Formulartechniken (mehrere Formular-Seiten, Captchas, htmlentities())
- Sessionverwaltung , Cookies
- Emailversand , Netzwerkfunktionen (sendmail, curl)
- Filefunktionen, Datenexport und –import (fgetcsv)
- PDF-Erzeugung

Quelle(n) : www.selfphp.de www.php.net www.php.de **...**

Allgemeines

- Die nachfolgenden Beispiele stellen die Lösung typischer Aufgaben bei der Entwicklung von webbasierten Anwendungen mittels PHP vor.
- In anderen Programmiersprachen und Frameworks sind diese Aufgaben in ähnlicher Weise zu lösen.

Verarbeitung von Formulardaten über GET oder POST

Versand von Formulardaten über GET oder POST :

- sinnvoll für kleine, genau definierte Datenmengen (max. 2 KByte)
- Trennung URL / Daten mit ?, Wertepaare mit &, Wertname/Value mit =
- Bsp.: Bestellen
- Vorteile: per Link realisierbar, schnell debuggbar im Browser
- Nachteil: Daten sind in der URTL sichtbar
- PHP-Zugriff über superglobales Array : $_GET['fid']

Versand über POST :

- auch umfangreiche Datenmengen (Textarea / Uploads) versendbar
- Bsp.: <input type="text" name="fid" size="10">
- PHP-Zugriff über superglobales Array $_POST['fid']

Formulardaten auf Existenz testen und Mehrfachwerte abfragen

Nicht alle Formularfelder werden immer übertragen :

- nur angekreuzte Checkboxen werden übermittelt
- keine Wert bei nicht ausgewählten Pulldownlisten
- Leere Input-Felder übertragen jedoch zumindest Leerstring (bzw. default-value)
- Prüfung auf Existenz des Formularwertes mit isset() oder direkt mit if (leer = false)

```
if ( isset( $_POST[privat] ))  echo "Privat !";  else  echo "privat nicht definiert !";
```

- Pulldownlisten können Mehrfachauswahl erlauben
- Deklaration des Feldnamens mit []´ notwendig :

```
<select name=“optionen[]" size="4" multiple>
```

- Auswertung mit Schleife über die Array-Werte :

```
if ($_POST['optionen']) {
        echo "Es sind folgende Optionen selektiert :<br>";
                foreach($_POST['optionen'] as $element) {
                                echo "$element<br>";
                        }
```

Formulare über mehrere Seiten

Bei umfangreicheren Eingaben ist eine Aufteilung sinnvoll :

- Gefahr des Datenverlustes durch technische Fehler ist geringer, Layout besser
- Zwischenergebnisse können ggf. schon geprüft oder geloggt werden
- Problem : Übertragung der Werte auf letzte Formularseite :
- Lösung 1 - durch Zwischenspeicherung (entspricht mehreren Einzelform.)
- Lösung 2 - durch automatische, versteckte Speicherung auf Folgeformularen
- gesendete Werte werden automatisch als hidden-Felder eingebunden :

```
if (isset($_POST)) {
   foreach ($_POST as $key => $element) {
     echo "<input type=\"hidden\" name=\"$key\" value=\"$element\">";
   }
}
```

- Beispiel kann auch für andere Formulargenerierungen verwendet werden
- **Sicherheit : alle Sicherheitsprüfungen müssen beim letzten Absenden komplett noch einmal durchgeführt werden, da diese Vorgehensweise nicht sicher ist**

Kodierung von Daten in URL´s und Formular-Aktionen

Leerzeichen und Sonderzeichen sind in URL´s nicht zulässig :

- Ihr☐Name?
- Automatische Umkodierung mit

  $kodiert = urlencode($name); // ergibt %20 statt Leerzeichen

- **Kodierung von Binärdaten mit base64**

  $code = base64_encode($string);

- **Analyse aller Header-Wert**

  $array = get_headers ('http://www.selfphp.de');
  print_r ($array);

Formulare mit Captcha´s vor Automatikbedienung schützen

- Bei einigen Anwendungen (Gewinnspiele., Umfragen, ressourcenbelastende Serviceangebote) ist ein Schutz gegen automatisierte Eingaben notwendig.
- Stellung einer Aufgabe zur Unterscheidung zwischen Mensch und Maschine !
- Optionen : Rechenaufgabe, Texterkennung aus Grafik (Captcha)

  Captcha (Completely Automated Public TuringTest to Tell Computers and Humans Apart)
- Bsp. Captcha – Kodierung

```
$im = @imagecreate($imgWidth, $imgHeight)
                                    or die("GD! Initialisierung fehlgeschlagen");
$fileTTF =  './arial.ttf';  // Fontdatei angeben
for($x=0;$x<6;$x++){
    $angel = rand(-25,25); // Zufallszahlenerzeugung
    $y = rand($size,$imgHeight-20);
    imagettftext($im, $size, $angel, $next, $y, $colors[$x],$fileTTF,$text[$x]);
    $next += $size + ($imgWidth/$size);
    $fileName .= $alphabet[$x];
}   verifycode =
```

- Komplettes Demobsp.: /selfphp/kochbuch/kochbuch22.html (->PHP-Skripte)

Cookies

Allgemein

- von Netscape als Ergänzung des zustandslosen HTTP-Protokolls entwickelt (ab Netscape Navigator 1.1)
- Cookies sind kleine Textdateien auf der Clientseite mit maximal 4 Kilobyte Größe
- ein Browser kann maximal 300 Cookies speichern.
- pro Domain können maximal 20 Cookies angelegt werden.
- Cookies können vom Browser nur Absender-Server zurückgesendet werden.
- Lebensdauer von - begrenzt auf Sitzungsdauer bis hin zu Tagen oder Monaten

Behandlung in PHP

- vorhandene Cookies werden in PHP als Variablen importiert, die den Namen des Cookies tragen
- globale Servervariable $_COOKIE ($HTTP_COOKIE_VARS) enthält alle gesetzten Cookies.

Cookies in PHP setzen und abfragen

Definieren eines Cookies

- Achtung: da Cookies im http-Header definiert werden, ist der nachfolgende Befehl VOR allen anderen Ausgaben zu schreiben !

```
$inhalt = "color1 =#770000";     // Inhalt festlegen, hier z.B. Lieblingsfarbe
// Cookie erzeugen
setcookie("cook1col1",$inhalt, time()+600);
//  600 entspricht der Lebensdauer in [s]  - hier  10 Minuten
```

Abfragen gesetzter Cookies

- Abfrage eines spezifischen Cookies

```
$colorinfo =   $cook1col1;   // ggf. vorher mit isset()  auf Existenz prüfen !
```

- Abfrage aller gesetzten Cookies

```
// Testanzeige
if (isset($_COOKIE)) {
  foreach ($_COOKIE as $key=>$element) {
    echo "$key : $element<br>";
  }    }
```

Session-Management mit PHP

Allgemeines

- HTTP ist verbindungsloses Protokoll, d.h. nach jedem Request wird die bestehende Verbindung beendet – es gibt keine Verknüpfung zwischen einzelnen Aufrufen
- Bei komplexen Anwendungen (Webshop, andere Workflowprozesse) sind damit zusammenhängende Transaktionen nicht mehr erkennbar !
- Aufgabe des Session-Management ist die Herstellung eines Zusammenhangs zwischen den Aufrufen des gleichen Nutzers

Generelle Optionen

- immer Generierung einer eindeutigen Session-ID (aus Zeit und NutzerID etc.)

1. Speicherung der SessionId in einem Cookie
2. Speicherung der Session als Hidden-Value oder als URL-Parameter

Sessions-Management unter PHP

- umfangreiche Funktionen (siehe auch Session-Modul von PHP in der PHP.ini)
- Daten der Session werden auf Server in Dateien oder im Hauptspeicher abgelegt

Session-Management mit PHP II

Erzeugen eindeutiger UserID´s

- Beim Start eines Workflows ist ggf. eine eindeutige UserID zu erzeugen :

```
$uid1 = uniqid (“Session"); // erzeugt eine einfache ID auf Basis der akt. Zeit
$uid2 = md5 (uniqid (“S")); // besser, mit MD5 kodiert
$uid3 = md5 (uniqid (rand())); // noch besser – mit Zufallszahl als Startwert
```

Starten einer Session

- Zum Starten einer Session ist session_start(); ganz am Anfang aufzurufen

```
session_start(); // Session Starten
echo “Seite 1 - <br><a href="seite2.php"> Gehe zu Seite2</a><br>,;
$_SESSION['uid'] = $udi3; // nun können der Session Daten zugeordnet werden
$_SESSION['Angebotnr'] = 2712;
$_SESSION['zeit'] = time();
```

- In Seite 2 können dann die Daten wieder abgerufen werden :

```
session_start(); // erneutes Starten der Session
echo “uid=“ . $_SESSION['uid'];
```

Email-Versand mit PHP

Notwendige Grundkonfiguration

- In der PHP.ini muss in [mail function] ein SMTP-Server eingetragen sein !

  SMTP = imail.informatik.htw-dresden.de

  smtp_port = 25

Versand von Emails mit PHP

- die Nachricht kann beliebig zusammengesetzt werden :

```
$an = "empfaenger@test.de";    $an =  $an . ",empfaenger2@test.de";
$betreff = "Betreff:  Test ";
$nachricht = " "<b> Hallo ... ";
$header = "From: mail@sender.de (Tom)\r\n";
$header .= "Content-Type: text/html\r\nContent-Transfer-Encoding: 8bit\r\n";
$header .= "X-Mailer: PHP ". phpversion();

mail($an, $betreff, $nachricht, $header );
```

- Über den Email-Header können auch base64-Kodierte Attachments versendet werden !

Netzwerkfunktionen mit CURL

Curl ist eine sehr umfangreiche Bibliothek zur Netzwerkkommunikation

- muß in php.ini freigeschaltet werden (Check mit php_info())
- automatischer Up- und Download von Dateien

```
$fp   = fopen($ftpFile, "r");  $handle = curl_init();
$url  = "ftp://test.de".$ftp['user'].":".$ftp['passwd']."@".
$url  = $url .  $ftp['host'].":21" .$ftp['pfad'].$ftp['file'];
curl_setopt($handle, CURLOPT_URL, $url);
curl_setopt($handle, CURLOPT_UPLOAD, 1);
curl_setopt($handle, CURLOPT_INFILE, $fp);
$result = curl_exec($handle);
```

- Test (Ping) von Servern und Auslesen von Serverinhalten

```
$fp = curl_init(„“www2.informatik.htw-dresden.de“);
curl_setopt($fp,CURLOPT_TIMEOUT,10);
curl_setopt($fp,CURLOPT_FAILONERROR,1);
curl_setopt($fp,CURLOPT_RETURNTRANSFER,1);
curl_exec($fp);
if (curl_errno($fp) != 0)    {  echo “Server OFFLINE ????“;      }
 else    {   echo “Server OK“;         }         curl_close($fp);
```

Generierung von PDF-Dateien

Speziell für den Qualitätsdruck ist die PDF-Generierung sinnvoll :

- Es muß in der php.ini extension=php_pdf.dll (Quelle www.pdflib.com)
- die Extension umfasst mehrere Dutzend Funktion zum Zeichen & Positionieren ...
- Einfaches Bsp.:

```
$file = fopen ( 'php.pdf', 'w' );
$dokument = pdf_open ( $file );
        pdf_begin_page ( $dokument, 200, 100 );
        pdf_set_leading ( $dokument, 40 );
        pdf_set_font ( $dokument, 'Times-Roman', 18, 'winansi' );
                pdf_show_xy ( $dokument, 'PDF-Funktionen', 10, 80 );
                pdf_set_font ( $dokument, 'Times-Roman', 14, 'winansi' );
                pdf_set_leading ( $dokument, 20 );
                pdf_continue_text ( $dokument, 'pdf_continue_text' );
                pdf_continue_text ( $dokument, 'pdf_stringwidth' );
        pdf_end_page ( $dokument );
pdf_close ( $dokument );  fclose ( $file );
```

PHP-Dateifunktionen

Einlesen und Schreiben von Dateien

```
echo readfile($filename);     //  gibt den Inhalt im Browser aus
```

- Alternativ Einlesen in ein Array mit file()

```
$datei = file("info.txt");  foreach($datei as $zeile) { echo "$zeile<br>";  }
```

- Beim Schreiben von Dateien kann optional mit flock() gesperrt werden !

```
$datei = fopen("daten.txt","w");  echo fwrite($datei, "Hallo Welt",100);
fclose($datei);
```

- Mit fgetcsv() können Komma-formatierte (Excel-) Dateien eingelesen werden.

```
 $datei = fopen("csvdaten.txt", "r");    $daten = fgetcsv($datei, 1000);
while ($daten) {     echo implode(" – ", $daten) . "<br>";
  $daten = fgetcsv($datei, 1000);  }
```

Backups von Variablen in Dateien speichern (Serialisierung)

-
```
$personen = array( "Matthias",  "Caroline",    "Gülten"  );
$daten = serialize($personen);   //  hier ggf. speichern und wieder lesen
// erzeugt :  a:3:{i:0;s:8:"Matthias";i:1;s:8:"Caroline";i:2;s:6:"Gülten";}
$personenbak = unserialize($daten);
```

PHP- Dateiupload verarbeiten

- Per Formular können Dateien hochgeladen werden

```
<input type="file" name="datei" size="40" maxlength="100000">
```

- Im PHP-Programm ist die temporäre Datei auf das Zielverzeichnis zu kopieren, da ansonsten bei Beendigung des Skriptes automatisch gelöscht wird.

```
if (isset($_FILES["datei"])) {     // Prüfen des Existenz des Arrays $_FILES
  if ($_FILES["datei"]["error"] == UPLOAD_ERR_OK) {  // Fehler ???
        // Muster zur Überprüfung des  Dateinamens
        $regExp = "/^[a-z_]([a-z0-9_-]*\.?[a-z0-9_-])*\.[a-z]{3,4}$/i";
        if (preg_match($regExp,$_FILES["datei"]["name"]) &&
            $_FILES["datei"]["size"] > 0 && $_FILES["datei"]["size"] < 100000)


 { // alles ok - Temporäre Datei in das Zielverzeichnis des Servers verschieben.
  move_uploaded_file($_FILES["datei"]["tmp_name"],"/".$_FILES["datei"]["name"]);
    header("Location: status.html"); // Redirect zur Erfolgsmeldung
  }
  else {   echo "Fehler: Im Dateinamen oder Dateigrössen Limit!";
  }  } else {    echo "Fehler: Während der Übertragung aufgetreten!";
  }  }   else {   echo "Fehler: Dateiupload fehlgeschlagen!";   }
```

PHP- XML-Daten verarbeiten

- Hauptaufgabe ist das Parsen von XML-daten (das Generieren ist einfacher)
- Zwei unterschiedliche Arten von XML-Parsern :

Tree-basierte Parser (Tree-Based Parsers)

- Laden das GESAMTE Dokument in den Arbeitsspeicher und stellen ein komplettes DOM der XML-Daten bereit
- geeignet für kleinere (ggf. Komplexere) Dokumente, bei großen Dokumenten können Performance und Speicherplatzbedarf eventuell kritisch sein
- Verfügbare Tools: SimpleXML DOM

Event-basierte Parser (Event-Based Parsers)

- Laden das Dokument elementsweise und „informieren“ das aufrufende PHP-Programm mit events über die gefundenen Inhalte
- Günstig für sehr große (oder auch laufend eingehende) XML-Daten, da keine komplette Ablage des DOM im Speicher, meist auch schneller
- Verfügbare Tools: XMLReader XML Expat Parser

PHP- SimpleXML Parser

Tree-basierter Parser SimpleXML Parser

• Als Standard in PHP ab Version 5 enthalten (keine Installation notwendig)

•Details unter http://www.w3schools.com/php/php_ref_simplexml.asp

•SimpleXML Parser kann aus einem String oder einer Datei laden

```
$xml=simplexml_load_string($myXMLData) or die("Error: Cannot create object");
```

```
•$xml=simplexml_load_file("note.xml") or die("Error: Cannot create object");
```

•In beiden Fällen wird eine Objektstruktur analog zum XML erzeugt:

•Mit print_r() kann Struktur (am Beispiel Buch-XML-Übung) ausgegeben werden:

•SimpleXMLElement Object ([buch] => Array ([0] => SimpleXMLElement Object ([Produkttitel] => SelfPHP[Produktcode] => 123456789 [Autor] => SimpleXMLElement Object ([Autorname] => Herr Maier [AutorID] => 1) [Verlag] => ….

•Direkter Zugriff auf einzelne Elemente über children( ) und Elementname:

```
foreach($xml->children() as $buch)
{  echo $buch->Produkttitel . ", " . "<br>";  }
```

PHP-Referenz, SelfPHP, PHP-Handbuch,

PHP- SimpleXML Parser – Zugriff auf Unterelemente und Attribute

```
XML:  <Autor><Autorname>Herr Maier</Autorname>
                    <AutorID>1</AutorID> </Autor>
      <abmessungen einheit="mm">
          <laenge>100</laenge>    <breite>20</breite>
      </abmessungen>
```

- Auf Unterelemente oder Attribute kann entsprechend über Objektverweise oder Arrayelemente zugegriffen werden:

```
foreach($xml->children() as $buch) {
   echo $buch->Produkttitel . ", "  ;
     $autorObj = $buch->Autor;   // Verweis auf Autor-Objekt !
     echo " Autor=" . $autorObj->Autorname;
     echo " Abm=" . $buch->abmessungen->laenge;
     echo $buch->abmessungen["einheit"];  // Attribute als Array
  }
```

```
PHP-Referenz, Autor=Herr Maier Abm=120mm
SelfPHP, Autor=Herr Schmidt Abm=100mm
PHP-Handbuch, Autor=Martin Mark Abm=90mm
```

PHP- XML Expat Parser

Event -basierter Parser (als Standard in PHP ab Version 5 enthalten)

•Details unter http://www.w3schools.com/php/php_xml_parser_expat.asp

Der Event-Parser muss mit entsprechenden Event-Handlern ausgestattet werden:

```
$parser=xml_parser_create();
// Function to use at the start of an element
function start($parser,$element_name,$element_attrs) {
  echo "<br>El.: " . $element_name . " = " ;    /*switch($element_name)
          { case "Produkttitel": echo "--<br>-- Titel: >"; break;   */        }
function stop($parser,$element_name) { echo " end!"; } // element end
function char($parser,$data) {  echo $data; }  // for character data

// Specify element handler
   xml_set_element_handler($parser,"start","stop");
// Specify data handler
   xml_set_character_data_handler($parser,"char");
```

PHP- XML Expat Parser

Danach zeilenweises Einlesen und Parsen der Daten:

```
// Open XML file and Read data
$fp=fopen("ue4.xml","r");
while ($data=fread($fp,4096)) {

  xml_parse($parser,$data,feof($fp)) or
     die (sprintf("XML Error: %s at line %d",
          xml_error_string(xml_get_error_code($parser)),
          xml_get_current_line_number($parser)));
}
// Free the XML parser
xml_parser_free($parser);
```

```
El.: PRODUKTE =
El.: BUCH =
El.: PRODUKTTITEL = PHP-Referenz end!
El.: PRODUKTCODE = 123456789 end!
El.: AUTOR =
El.: AUTORNAME = Herr Maier end!
El.: AUTORID = 1 end! end!
```

Zusammenfassung

- PHP stellt im vergleich zu anderen Lösungen sehr mächtige und effizient einsetzbare Funktionen bereit

- ggf. noch fehlende Funktionen können über die PEAR-Bibliotheken (vgl. Folge-VL) gesucht und eingebunden werden oder mittels C auch neu entwickelet werden