

*Handout zur Vorlesung*  
**DBSP – PHP II – Unit 5**

## Vorlesung



University of Applied Sciences

# DBSP

## Unit 5 PHP II

Grundlegende Elemente eines PHP Programms (II)

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

1



University of Applied Sciences



**Prof. Dr. rer. nat.  
Nane Kratzke**

*Praktische Informatik und  
betriebliche Informationssysteme*

- Raum: 17-0.10
- Tel.: 0451 300 5549
- Email: [kratzke@fh-luebeck.de](mailto:kratzke@fh-luebeck.de)



@NaneKratzke

Updates der Handouts auch über Twitter #dbsp

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

2

## **Übergreifende Ziele der Lehrveranstaltung**



University of Applied Sciences

Client- und Serverseitige Entwicklung

PHP (Serverseitig)

JavaScript (Clientseitig)

„Hosten“ von Apps

Framework Erfahrungen

CMS (Drupal)

WebServices (Google-Maps)

jQuery

Datenbank-Integration

Berücksichtigung von Sicherheitsaspekten

HTML-Injections

SQL-Injections

Session Hijacking

Login-Systeme

Um sich weitere Web-Technologien autodidaktisch erarbeiten zu können.

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

3

## **Units**



University of Applied Sciences

**Unit 1**  
Cloud Computing IaaS

**Unit 2**  
CMS Drupal

**Unit 3**  
HTML und CSS

**Unit 4 - 7**  
PHP I - IV

**Unit 8**  
Sessions, Cookies, Formulare und Login-System

**Unit 9**  
JavaScript

**Unit 10**  
Drupal Module Development

**Unit 11**  
Datenmodellierung

**Unit 12 - 13**  
Datenbanken und SQL  
Vom Datenmodell zur Datenbank

**Unit 14**  
Datenbank-gestützte Web-Anwendungen

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

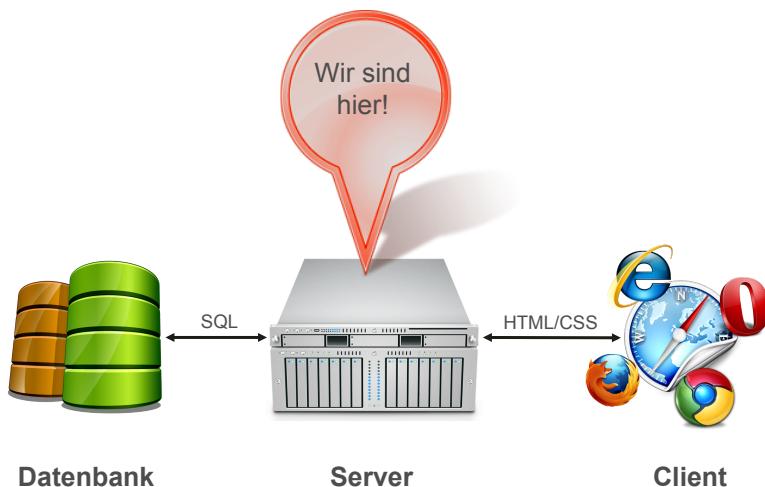
4

## Datenbank – Server – Client

Wo waren wir nochmal?



University of Applied Sciences



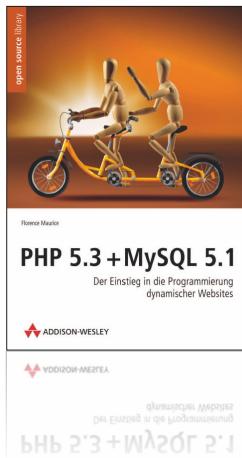
Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

5

## Zum Nachlesen ...



University of Applied Sciences



### Kapitel 5

#### Mehr Basics

Kapitel 5.3 Funktionen schreiben

Kapitel 5.4 Klassen und Objekte

### Kapitel 4

#### Programmstrukturen

Kapitel 4.3 Funktionen



### Kapitel 7

#### Objektorientierte Programmierung

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

6

## Zum Nachlesen ...



### Bereitgestellte Skripte:

#### **PHP Programmierung**

- **Kapitel 4:** Funktionen
- **Kapitel 8:** Einführung in die OO-Programmierung mit PHP
- **Kapitel 9:** Grundlagen der OOP
- **Kapitel 10:** Objekte
- **Kapitel 11:** Sichtbarkeit
- **Kapitel 12:** Magische Methoden

<http://praktische-informatik.fh-luebeck.de/node/39>

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

7

## Inhalte der PHP Units



1

- Laufzeitmodelle und Programmierparadigma

2

- Elemente eines PHP-Programms

3

- Datentypen

4

- Operatoren

5

- Kontrollstrukturen

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

8

## Elemente eines PHP-Programms



University of Applied Sciences

### Programmelemente

- Anweisungen
- Variablen
- **Funktionen**
- Objekte

### Anwendungen

- PHP in HTML einbinden
- Bibliotheken und Archive

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

9

## Funktionen



University of Applied Sciences

### Syntax:

```
function Name([Parameter]) {  
    // Anweisungen der Function  
    return $rvar; // ggf.  
}
```

- Codefragmente,
- die Anweisungen kapseln
- um eine ggf. parametrisierte Funktionalität
- zu realisieren
- und ggf. einen Rückgabewert haben.

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

10

## Funktionen Beispiel



University of Applied Sciences

```
function conc($word1, $word2, $word3) {  
    $concstr = "$word1$word2$word3";  
    return $concstr;  
}  
  
echo conc("Hello", " ", "World");
```

Funktionen-  
name

Aufruf-  
parameter

Rückgabe

Prof. Dr. rer. nat. Nane Kratzke

Praktische Informatik und betriebliche Informationssysteme

11

## Funktionen Beispiel



University of Applied Sciences

```
function conc($word1, $word2, $word3) {  
    $concstr = "$word1$word2$word3";  
    return $concstr;  
}  
  
echo conc("Hello", " ", "World");
```

Funktionen-  
name

Aufruf-  
parameter

Rückgabe

Prof. Dr. rer. nat. Nane Kratzke

Praktische Informatik und betriebliche Informationssysteme

12

## Funktionen Beispiel



University of Applied Sciences

```
function conc($word1, $word2, $word3) {  
    $concstr = "$word1$word2$word3";  
    return $concstr;  
}  
  
echo conc("Hello", " ", "World");
```

Funktionen-name

Aufruf-parameter

Rückgabe

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

13

## Funktionen Beispiel



University of Applied Sciences

```
function conc($word1, $word2, $word3) {  
    $concstr = "$word1$word2$word3";  
    return $concstr;  
}  
  
echo conc("Hello", " ", "World");
```

Methoden-name

Aufruf-parameter

Rückgabe

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

14

## Funktionen Default Parameter



University of Applied Sciences

### Syntax:

```
function Name([$Par, ...], [$DPar = default, ...]) {  
    // Anweisungen der Function  
    return $rvar;  
}
```

- Die zuletzt stehenden Parameter einer Funktion können mit Defaultwerten gesetzt werden,
- So dass diese beim Aufruf nur dann angegeben werden müssen, wenn mit nicht default-Werten gearbeitet werden soll.

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

15

## Funktionen Default Parameter



University of Applied Sciences

Beispiel für Default-Parameter:

```
function add_offset($v, $offset = 10) {  
    return $v + $offset;  
}
```

```
echo add_offset(5, 5); 10
```

```
echo add_offset(5, -3); 2
```

```
echo add_offset(5); 15
```

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

16

## Funktionen

### Call by reference vs. Call by value



University of Applied Sciences

- Es gibt in gängigen Programmiersprachen grundsätzlich zwei Arten Parameter an eine Routine zu übergeben

#### Call by reference

- Es wird ein Zeiger auf eine Variable übergeben.
- Innerhalb der Routine wird über den Zeiger auf der Variable außerhalb der Routine gearbeitet.
- **Der Inhalt der Variable außerhalb der Routine wird verändert.**

#### Call by value

- Der Wert einer Variable wird in die Parametervariable kopiert.
- Innerhalb der Routine wird auf der Kopie gearbeitet.
- **Der Inhalt der Variable außerhalb der Routine wird nicht verändert.**

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

17

## Funktionen

### Referenz-Parameter



University of Applied Sciences

#### Syntax:

```
Function name([&$Par, ...]) {  
    // Anweisungen der Function  
    return $rvar;  
}
```

- Parameter, die **per Referenz** an die Funktion übergeben werden werden durch den **&-Operator** gekennzeichnet.
- Hierdurch wirken sich Änderungen, die an einem Referenzparameter innerhalb einer Funktion gemacht werden, auch außerhalb der Funktion aus.

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

18

## Funktionen Call by value Beispiel



University of Applied Sciences

### Funktionsaufruf mit Call by Value

```
function add($a, $b) {  
    $result = $a + $b;  
    echo "$a + $b = $result";  
    $a = 0;  
    $b = 0;  
}
```

Der folgende Aufruf erzeugt

```
$a = 5;  
$b = 3;  
echo add($a, $b);  
echo add($a, $b);
```

welche Ausgabe?

```
5 + 3 = 8  
5 + 3 = 8
```

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

19

## Funktionen Call by reference Beispiel



University of Applied Sciences

### Funktionsaufruf mit Call by Reference

```
function add($a, &$amp;b) {  
    $result = $a + $b;  
    echo "$a + $b = $result";  
    $a = 0;  
    $b = 0;  
}
```

Der folgende Aufruf erzeugt

```
$a = 5;  
$b = 3;  
echo add($a, $b);  
echo add($a, $b);
```

welche Ausgabe?

```
5 + 3 = 8  
5 + 0 = 5
```

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

20

## Funktionen Geltungsbereich von Variablen (I)



University of Applied Sciences

### Globaler Geltungsbereich

- Variablen, die außerhalb von Funktionen definiert werden

### Lokaler Geltungsbereich

- Variablen, die innerhalb von Funktionen definiert werden

#### Beispiel:

```
$vorname = "Marlen";  
  
function gruss() {  
    $vorname = "Lilli";  
    return "Hallo $vorname.";  
}  
  
echo gruss();  
echo $vorname;
```

Hallo Lilli.

Marlen

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

21

## Funktionen Geltungsbereich von Variablen (II)



University of Applied Sciences

Wenn explizit auf eine Variable des globalen Geltungsbereichs zugegriffen werden soll,

dann müssen ansonsten lokale Variablen explizit in ihrem Nutzungskontext als **global** gekennzeichnet werden.

#### Beispiel:

```
$vorname = "Marlen";  
  
function gruss() {  
    global $vorname;  
    $vorname = "Lilli";  
    return "Hallo $vorname.";  
}  
  
echo gruss();  
echo $vorname;
```

Hallo Lilli.

Lilli

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

22

## Lambda Funktionen



University of Applied Sciences

### Funktionen sind Elemente erster Ordnung

- sie können Variablen zugewiesen werden.
- sie können adhoc und unbenannt definiert werden.

### Sprachmittel

- der sogenannten funktionalen Programmierung,
- die häufig als zu akademisch abgestempelt werden.

#### Beispiel:

```
$lambda = function($a) { return $a/2; };  
echo $lambda(4);
```

2

```
$list = array(6, 8, 10, 14, 18);  
echo array_map($lambda, $list);
```

array(3,4,5,7,9)

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

23

## Funktionen Mini-Übung: Aufrufparameter



University of Applied Sciences

```
function func($x, $y, $z = 0) {  
    return "$x + $y = $z";  
}
```

```
echo func(5, "Hello", 42);  
echo func("Hello", 42);
```

5 + Hello = 42

Hello + 42 = 0

```
function func(&$x, &$y) {  
    $temp = $x;  
    $y = $x;  
    $y = $temp;  
}
```

```
$a = 5;  
$b = "Hello World";  
func($a, $b);  
echo "a: $a";  
echo "b: $b";
```

a: Hello World

b: 5

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

24

## Funktionen

### Mini-Übung: Geltungsbereich von Variablen



University of Applied Sciences

```
$var = 10;  
  
function func($v) {  
    $var = $v + 10;  
    return $var;  
}  
  
echo func($var);  
echo $var;
```

20

10

```
$var = 10;  
  
function func($v) {  
    global $var;  
    $var = $v + 10;  
    return $var;  
}  
  
echo func($var);  
echo $var;
```

20

20

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

25

## Funktionen

### Mini-Übung: Lambda-Funktionen



University of Applied Sciences

```
$f = function($a,$b) { return $a + $b;};  
echo $f(5,6);
```

11

```
$f = function($a) { return "$a";};  
echo $f(3.5e3);
```

3500

```
$f = function($a) { return $a*$a;};  
echo $f(1.1e0);
```

1.21

```
$f = function($ws) { return join($ws); };  
echo $f("Hello", " ", "World");
```

Hello World

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

26

## Elemente eines PHP-Programms



University of Applied Sciences

### Programmelemente

- Anweisungen
- Variablen
- Funktionen
- **Objekte**

### Anwendungen

- PHP in HTML einbinden
- Bibliotheken und Archive

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

27

## Objektorientierung



University of Applied Sciences

Definition  
einer Klasse

Methoden und  
Eigenschaften

Referenzen  
und Klonen

Objekte  
vergleichen

Konstruktoren  
und  
Destruktoren

Vererbung  
und  
Schnittstellen

Zugriffs-  
modifikatoren

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

28

## Definition von Klassen



University of Applied Sciences

### Syntax:

```
class Klassenname {  
    // Datenfeldekl.  
    // Methodenimpl.  
    ...  
}
```

- Klassen sind das wichtigste Strukturierungsmittel objekt-orientierter Sprachen
- Eine **Klasse** enthält
  - **Menge von Datenfeldern** (Zustand eines Objekts der Klasse)
  - **Menge von Methoden** (Verhalten eines Objekts der Klasse)

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

29

## Methoden und Eigenschaften



University of Applied Sciences

Objekte haben Eigenschaften

- **Datenfelder** im OO-Sprachgebrauch
- Haben Sie bisher als Variablen kennengelernt

Objekte haben Verhalten

- **Methoden** im OO-Sprachgebrauch
- Haben Sie bisher als Funktionen kennengelernt

Beispielklasse Kunde:

```
class Kunde {  
    public $name; // Datenfeld  
  
    public function halloSagen() { // Methode  
        echo "Hallo {$this->name}";  
    }  
}
```

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

30

## Erzeugen von Objekten

### Zugriff auf Methoden und Eigenschaften von Objekten



University of Applied Sciences

```
class Kunde {  
    public $name;  
    public function halloSagen() { ... };  
}
```

```
// Erzeugen eines Objekts der Klasse Kunde  
$kunde = new Kunde();  
  
// Setzen des Datenfelds name  
$kunde->name = "Anja";  
  
// Aufrufen einer Methode  
$kunde->halloSagen();  
  
// Lesen eines Datenfelds  
echo "Ich heiße {$kunde->name}";
```

Hallo Anja

Ich heiße Anja

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme 31

## Konstruktoren und Destruktoren



University of Applied Sciences

### Spezielle Methoden

- Aufruf beim Erzeugen (Konstruktor)
- Aufruf beim Löschen eines Objekts (Destruktor)

### Dienen dazu

- Objekte zu initialisieren (z.B. DB-Verbindungen aufzubauen)
- oder zum „Aufräumen“ (z.B. DB-Verbindungen zu schließen).

- In PHP heißen diese
  - `__construct()`
  - `__destruct()`

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme 32

## Konstruktoren und Destruktoren

### Beispiel der Wirkungsweise



University of Applied Sciences

```
class Kunde {  
    public $name;  
    public function __construct($kname) {  
        echo "Kunde $kname wird angelegt.";  
        $this->name = $kname;  
    }  
    public function halloSagen() { ... };  
    public function __destruct() {  
        echo "Kunde {$this->name} wurde gelöscht.";  
    }  
}
```

```
$kunde = new Kunde("Papa Schlumpf");
```

Kunde Papa Schlumpf wird  
angelegt.

```
$kunde->halloSagen();
```

Hallo Papa Schlumpf

```
unset($kunde);
```

Kunde Papa Schlumpf wurde  
gelöscht.

Prof. Dr. rer. nat. Nane Kratzke    33  
Praktische Informatik und betriebliche Informationssysteme

## Referenzen und Klone



University of Applied Sciences

Der Zuweisungsoperator bei Objekten hat eine Besonderheit.

- Es werden nicht die Werte eines Objekts zugewiesen,
- sondern nur die Referenz auf ein Objekt.
- Soll die Wertesemantik genutzt werden, so ist vor der Zuweisung der **clone** Operator anzuwenden.

### clone

- Der **clone** Operator erzeugt ein Duplikat eines Objekts im Hauptspeicher



Prof. Dr. rer. nat. Nane Kratzke    34  
Praktische Informatik und betriebliche Informationssysteme

## Referenzen und Klone Beispiel der Wirkungsweise



University of Applied Sciences

```
class Farbe {  
    public $wert;  
  
    public function __construct($f) {  
        $this->wert = $f;  
    }  
}
```

```
$f1 = new Farbe("blau");  
$f2 = $f1;  
$f1->wert = "rot";  
echo $f2->wert;
```

rot

```
$f1 = new Farbe("blau");  
$f2 = clone $f1;  
$f1->wert = "rot";  
echo $f2->wert;
```

blau

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

35

## Objekte vergleichen mit Vergleichsoperatoren == und ===



University of Applied Sciences

==

- Haben die Datenfelder identische Werte

===

- Zeigen die Objekt-Handler auf dasselbe Objekt
- (JAVA-Semantik)

Selbes Beispiel wie beim Klonen:

```
$f1 = new Farbe("blau");  
$f2 = $f1;  
var_dump($f1 == $f2);  
var_dump($f1 === $f2);
```

TRUE

TRUE

```
$f1 = new Farbe("blau");  
$f2 = clone $f1;  
var_dump($f1 == $f2);  
var_dump($f1 === $f2);
```

TRUE

FALSE

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

36

## Erben einer Spezifikation Zwei Varianten in PHP



University of Applied Sciences

### Variante 1: Implementieren einer Schnittstelle

```
interface IF {
    public function implement_this();
    public function implement_this_too();
}

class An_Object implements IF {

    public function implement_this() {
        // now you have to implement this
    }

    public function implement_this_too() {
        // now you have to implement this
    }
}
```

Definieren einer Schnittstelle

Nutzen einer Schnittstelle

Implementieren der geforderten Funktionalität

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

37

## Erben einer Spezifikation Zwei Varianten in PHP



University of Applied Sciences

### Variante 2: Erben von abstrakten Klassen

```
abstract class AbstractObject {
    function already_implemented() {
        // some magic code
    }
    abstract function implement_this();
    abstract function implement_this_too();
}

class An_Object extends AbstractObject {
    function implement_this() {
        // now you have to implement this
    }

    function implement_this_too() {
        // now you have to implement this
    }
}
```

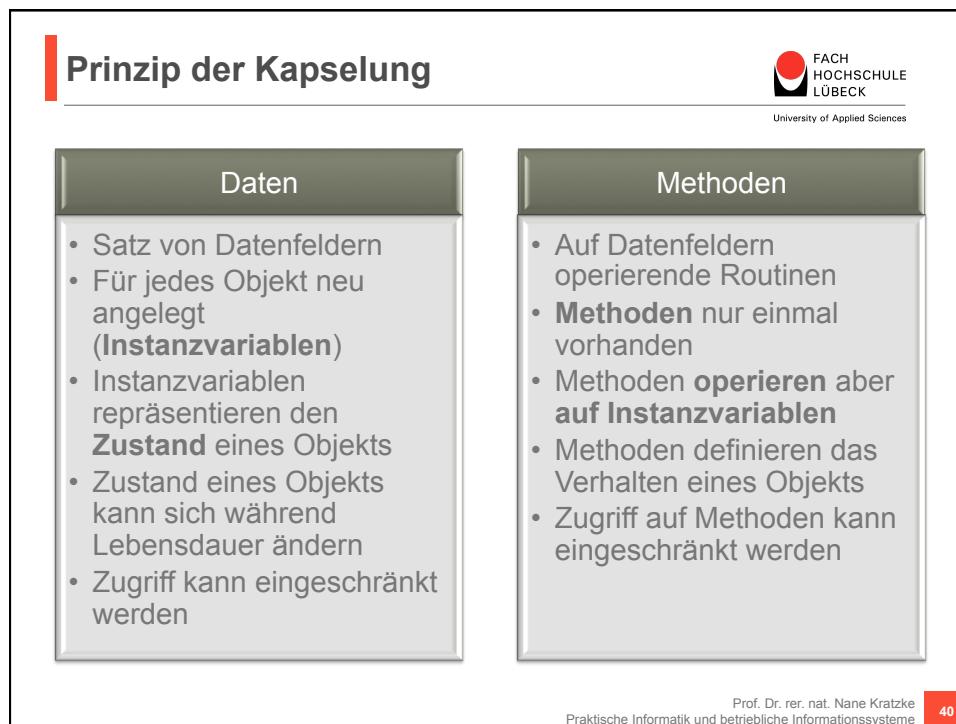
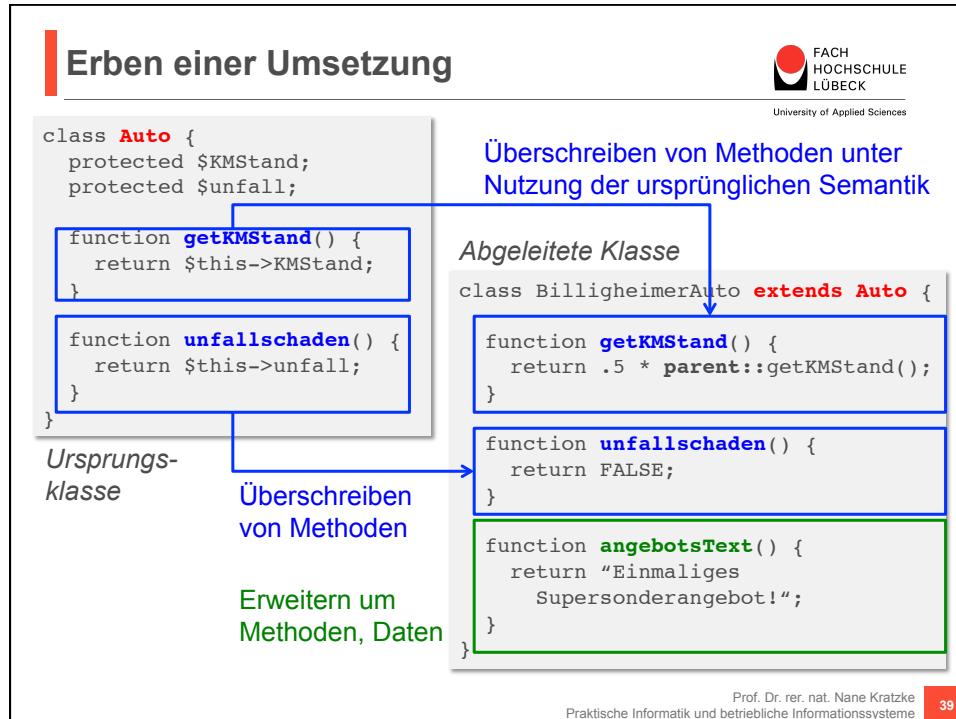
Definieren einer Klasse mit abstrakten Methoden

Ableiten von einer abstrakten Klasse

Implementieren der geforderten Funktionalität

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

38



## Zugriffsmodifikatoren in PHP



University of Applied Sciences

Zugriffs- modifikatoren	Bedeutung
<b>private</b>	Zugriff auf Methode oder Datenfeld ist nur innerhalb der definierenden Klasse erlaubt.
<b>protected</b>	Zugriff auf Methode oder Datenfeld ist nur aus abgeleiteten Klassen der definierenden Klasse erlaubt.
<b>public</b>	Zugriff auf Methode oder Datenfeld wird nicht eingeschränkt.

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

41

## Daten- und Methodenzugriffsmodifikatoren **public, protected und private**



University of Applied Sciences

```
class An_Object {  
  
    public $forall;  
  
    protected $forchildren;  
  
    private $class_eyes_only;  
  
    public function public_method() { ... };  
  
    protected function protected_method() { ... };  
  
    private function private_method() { ... };  
  
}
```

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

42

## Mini-Übung: Objektorientierung (I)



University of Applied Sciences

```
class Heizkoerper {
    protected $temp;

    public function __construct($init) {
        echo "Heizkoerper wird eingeschaltet";
        $this->temp = min(45, max($init, 5));
        $this->fuehlerausgabe();
    }

    public function waermter($grad) {
        $this->temp = min($this->temp + $grad, 45);
        $this->fuehlerausgabe();
    }

    public function kaelter($grad) {
        $this->temp = max($this->temp - $grad, 5);
        $this->fuehlerausgabe();
    }

    public function fuehlerausgabe() {
        echo "Heizkoerper ist {$this->temp} Grad C warm";
    }

    public function __destruct() {
        echo "Heizkoerper wird ausgeschaltet";
    }
}
```

`$h = new Heizkoerper(25);`

Heizkoerper wird eingeschaltet

Heizkoerper ist 25 Grad C warm

`$h->waermter(10);`

Heizkoerper ist 35 Grad C warm

`$h->kaelter(40);`

Heizkoerper ist 5 Grad C warm

`echo "{$h->temp}";`

Fatal Error!!!

`unset($h);`

Heizkoerper wird ausgeschaltet

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

43

## Mini-Übung: Objektorientierung (II)



University of Applied Sciences

```
class IntegrierbarerHeizkoerper
    extends Heizkoerper
{
    public function getTemp() {
        return $this->temp;
    }
}
```

`$h = new IntegrierbarerHeizkoerper(25);`

Heizkoerper wird eingeschaltet

Heizkoerper ist 25 Grad C warm

`$h->waermter(10);`

Heizkoerper ist 35 Grad C warm

`$h->kaelter(40);`

Heizkoerper ist 5 Grad C warm

`echo "{$h->getTemp()}";`

5

`unset($h);`

Heizkoerper wird ausgeschaltet

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

44

## Mini-Übung: Objektorientierung (III)



University of Applied Sciences

```
$h = new IntegrierbarerHeizkoerper(25);  
$h2 = $h;  
$h2->waermer(15);  
echo "{$h->getTemp()}";
```

40, obwohl die Temperatur an  
\$h2 geändert wurde (Referenz).

```
var_dump($h == $h2);  
var_dump($h === $h2);
```

bool(TRUE) - Inhaltsvergleich  
Bool(TRUE) - Referenzvergleich

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

45

## Mini-Übung: Objektorientierung (IV)



University of Applied Sciences

```
$h = new IntegrierbarerHeizkoerper(25);  
$h2 = clone $h;
```

bool(TRUE) – Inhaltsvergleich  
bool(FALSE) - Referenzvergleich

```
$h2->waermer(15);  
echo "{$h->getTemp()}";
```

25, obwohl die Temperatur am  
Klon \$h2 geändert wurde.

```
var_dump($h == $h2);  
var_dump($h === $h2);
```

bool(FALSE) – Inhaltsvergleich  
bool(FALSE) - Referenzvergleich

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

46

## Elemente eines PHP-Programms



### Programmelemente

- Anweisungen
- Variablen
- Funktionen
- Objekte

### Anwendungen

- **PHP in HTML einbinden**
- Bibliotheken und Archive

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

47

## PHP in (X)HTML-Dokumente einbinden



PHP Code Statements werden üblicherweise direkt in HTML-Dokumente eingebunden.

Hierzu müssen PHP-Statement innerhalb der Tags <?PHP und ?> notiert werden.

Die Datei muss die Endung PHP haben.

PHP-Statements können an beliebigen Stellen im HTML-Dokument stehen, sofern sie durch <?PHP und ?> Tags für den PHP-Interpreter gekennzeichnet sind.

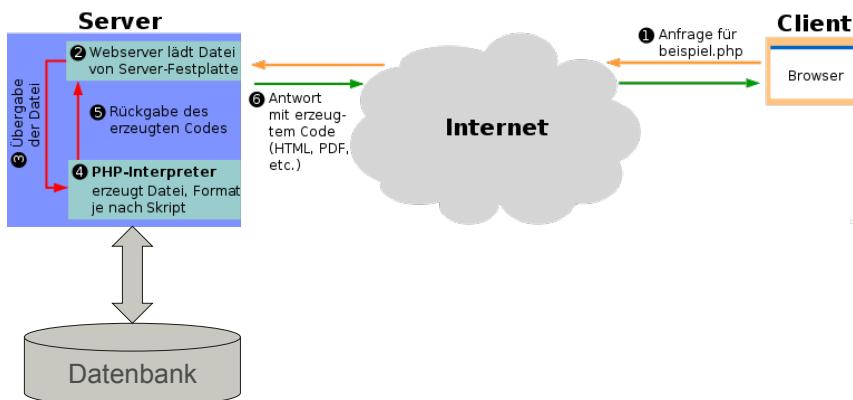
Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

48

## PHP Funktionsweise



University of Applied Sciences



Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

49

## PHP Hypertext Processing



University of Applied Sciences

### beispiel.php:

```
<html>
<head>
<title><?php echo getTitleFromDB(); ?></title>
</head>
<body bgcolor="<?php echo getBGColor(); ?>">

<?php echo "Schönen Tag auch"; ?>
</body>
</html>
```

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

50

## PHP Hypertext Processing



University of Applied Sciences

beispiel.php:

```
<html>
<head>
<title><?php echo getTitleFromDB(); ?></title>
</head>
<body bgcolor="<?php echo getBGCOLOR(); ?>">

<?php echo "Schönen Tag auch"; ?>
</body>
</html>
```

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

51

## PHP Hypertext Processing



University of Applied Sciences

beispiel.php:

```
<html>
<head>
<title>Willkommen</title>
</head>
<body bgcolor="<?php echo getBGCOLOR(); ?>">

<?php echo "Schönen Tag auch"; ?>
</body>
</html>
```

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

52

## PHP Hypertext Processing



University of Applied Sciences

**beispiel.php:**

```
<html>
<head>
<title>Willkommen</title>
</head>
<body bgcolor="<?php echo getbgcolor(); ?>">

<?php echo "Schönen Tag auch"; ?>
</body>
</html>
```

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

53

## PHP Hypertext Processing



University of Applied Sciences

**beispiel.php:**

```
<html>
<head>
<title>Willkommen</title>
</head>
<body bgcolor="yellow">

<?php echo "Schönen Tag auch"; ?>
</body>
</html>
```

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

54

## PHP Hypertext Processing



University of Applied Sciences

**beispiel.php:**

```
<html>
<head>
<title>Willkommen</title>
</head>
<body bgcolor="yellow">

<?php echo "Schönen Tag auch"; ?>

</body>
</html>
```

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

55

## PHP Hypertext Processing



University of Applied Sciences

**beispiel.php:**

```
<html>
<head>
<title>Willkommen</title>
</head>
<body bgcolor="yellow">

Schönen Tag auch

</body>
</html>
```

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

56

## Aufbau eines PHP-Skripts



University of Applied Sciences

### Programmelemente

- Anweisungen
- Variablen
- Funktionen
- Objekte

### Anwendungen

- PHP in HTML einbinden
- **Bibliotheken und Archive**

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

57

## Bibliotheken und Archive



University of Applied Sciences

- Viele Funktionalitäten werden bereits durch externe PHP-Bibliotheken bereitgestellt
- Diese lassen sich einbinden und in eigenen Projekten nutzen
- Gleiches gilt für eigene Funktionalitäten, die an mehreren Stellen in Projekten genutzt werden kann
- Hierzu sieht PHP zwei Sprachkonstrukte vor



Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

58

## Importieren von Dateien, Bibliotheken und Archiven



University of Applied Sciences

### **include**

- Zum Einbinden von Dateien
- HTML und PHP
- Die Skript-Ausführung wird fortgesetzt (nur Warning)

### **require**

- Zum Einbinden von Dateien
- HTML und PHP
- Die Skript-Ausführung wird abgebrochen

### **include\_once**

- Die Datei wird bei Mehrfacheinbindungen nur einmal geladen.
- Ggf. erforderlich wenn Variablen und Datenstrukturen nur einmal initialisiert werden sollen.
- Ansonsten wie **include**

### **require\_once**

- Die Datei wird bei Mehrfacheinbindungen nur einmal geladen.
- Ggf. erforderlich wenn Variablen und Datenstrukturen nur einmal initialisiert werden sollen.
- Ansonsten wie **require**

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

59

## PHP Archive



University of Applied Sciences

### Bei der Verteilung von PHP „Applikationen“ oder Bibliotheken

- ist es sinnvoll, diese als ein „Paket“ bereitzustellen.
- Zur Erstellung distributierbarer PHP-Archive, existiert eine **PHAR**-Klasse
- Das folgende Listing erzeugt ein PHAR-Archiv namens **distrib.phar** aller im Unterverzeichnis **bibliothek** liegenden Dateien.

Erzeugen eines PHAR Archivs

```
$phar = new Phar("distrib.phar");
$phar->buildFromDirectory("bibliothek");
```

Hierzu muss die Erstellung von PHAR-Archiven in der **PHP.ini** erlaubt werden. **phar.readonly off**

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

60

## PHP Archive importieren und finden



University of Applied Sciences

PHAR Archive können mittels **include** und **require** importiert werden

```
include "distrib.phar";
```

Unter den folgenden Links können PHP Bibliotheken bezogen werden, die den Standardumfang von PHP erweitern.

<http://pear.php.net>

PHP Extension  
And Application  
Repository



<http://pecl.php.net>

PHP Extension  
Community  
Library

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

61

## Zusammenfassung



University of Applied Sciences

- **Elemente eines PHP-Programms (II)**
- Funktionen, unbenannte Lambda-Funktionen
- Funktionale Sprachfeatures von PHP
- Objekte und Klassen
- PHP Einbindung in HTML
- Pakete und Archive

Prof. Dr. rer. nat. Nane Kratzke  
Praktische Informatik und betriebliche Informationssysteme

62