Barbara Zimmermann

Separierung der Darstellung durch Templates

Seminar der Projektgruppe
„Generierung von Web-Anwendungen aus visuellen Spezifikationen“

Universität Paderborn

Agenda

- Motivation
- Template Engines
  - Konzepte
  - Smarty
  - Vlib
- Formularerzeugung
  - HTML_QuickForm
- Fazit

Probleme

- Problem: Mischung von PHP und HTML innerhalb des Codes

```
echo „<table width=’80%’>\n\t<tr bgcolor=’$farbe’>“
```

- Codeduplizierung

```
01 float max (float a, float b)
02 {
03     return (a<b) ? b : a;
04 }
05
06 int max (int a, int b)
07 {
08     return (a<b) ? b : a;
09 }
```

Lösung durch Templates

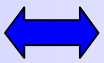
Barbara Zimmermann

- „Schablonen“ erzeugen

template <…> leitet Template ein, alles Folgende kann unbestimmte Typen enthalten

Template-Parameter, Platzhalter für Typ

```
01 template<typename T>
02 const T & max (const T &
       a, const T & b)
03 {
04     return (a<b) ? b : a;
05 }
```

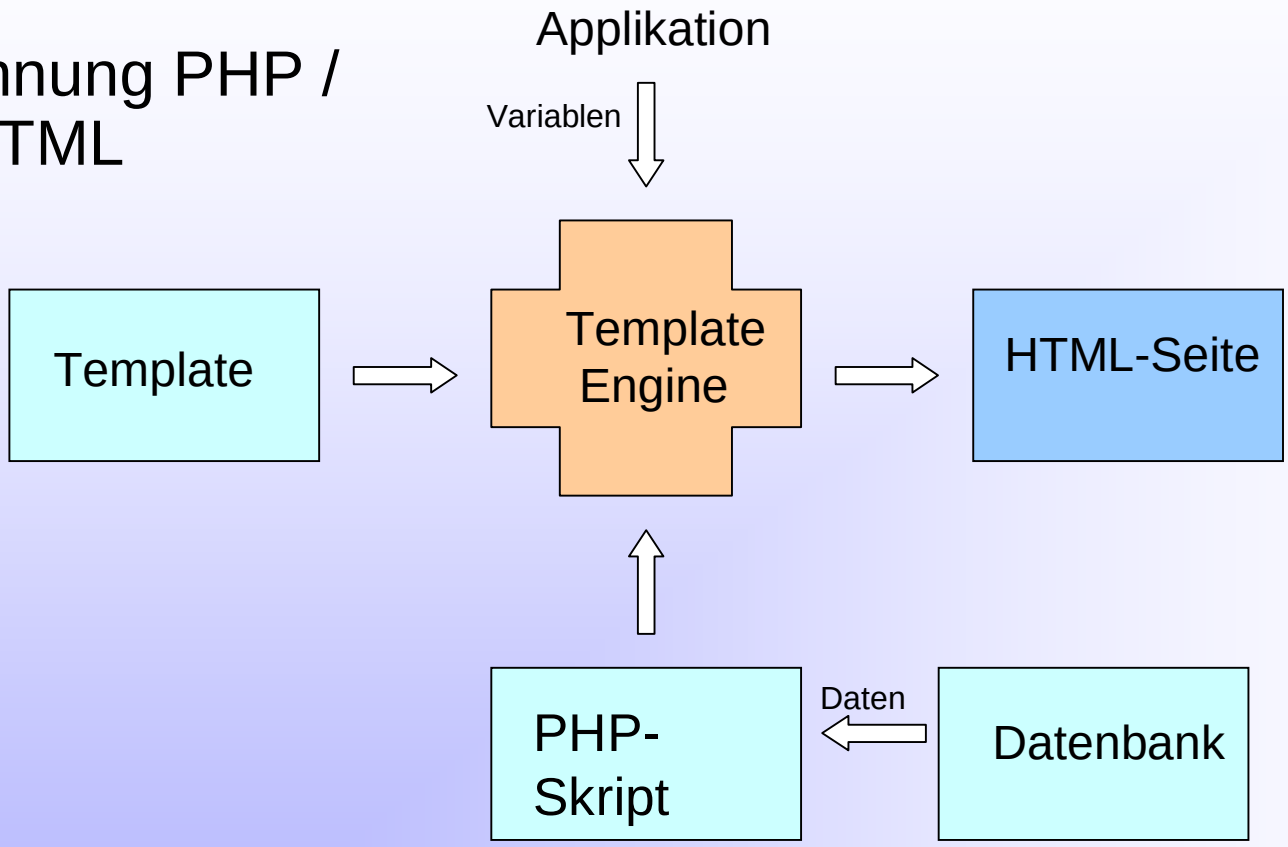


```
01 float f1 = 1.2, f2 = 2.3;
02 float f3 = max (f1, f2);
03
04 int i1 = 1, i2 = 2;
05 int i3 = max (i1, i2);
```

Template-Parameter kann überall vorkommen, wo sonst Typ stehen darf

Konzept

Trennung PHP / HTML

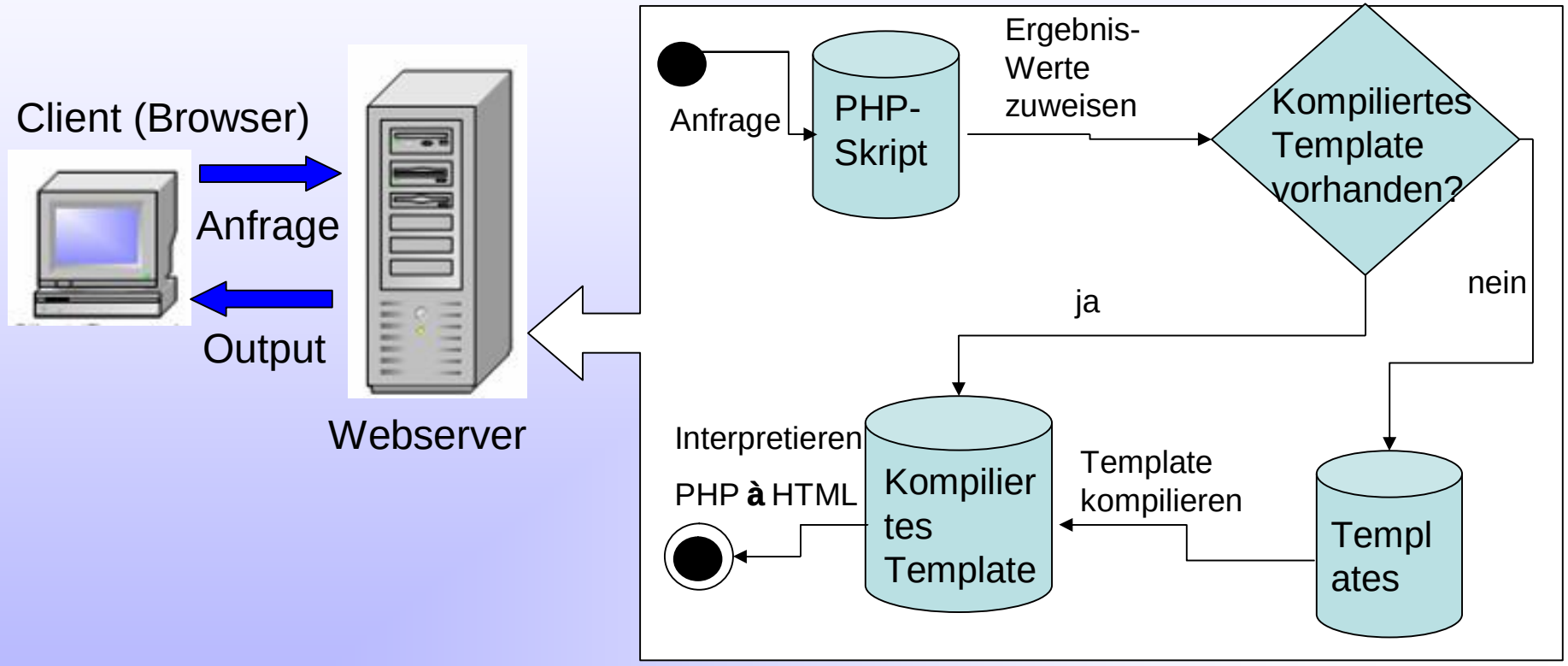


- **è** Trennung der Verantwortlichkeiten
- **è** Codereduzierung

Smarty

Barbara Zimmermann

Die „selbstkompilierende“ Template Engine



Unterstützt übliche Programmierkonstrukte

- Schleifen, Verzweigungen, Funktionen, Datenbankanfragen

Beispiel Smarty

Barbara Zimmermann

index.php

```
include('Smarty.class.php');
$smarty = new Smarty;
$smarty->assign('name', 'george smith');
$smarty->assign('address', '45th Harris');
$smarty->display('index.tpl');
```

index.tpl

```
<html>
<head>
<title>User Info</title>
</head>
<body>
User Information:<p>
Name: {$name}<br>
Address: {$address}<br>
</body>
</html>
```

Beispiel Smarty

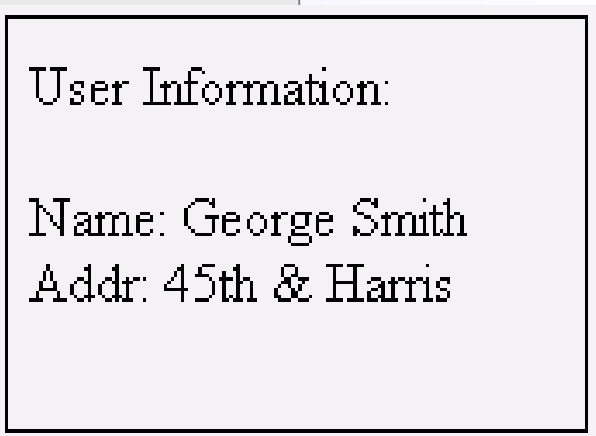
Barbara Zimmermann

index.php

```
include('Smarty.class.php');
$smarty = new Smarty;
$smarty->assign('name', 'george smith');
$smarty->assign('address', '45th Harris');
$smarty->display('index.tpl');
```

Ausgabe

```
<html>
<head>
<title>User Info</title>
</head>
<body>
User Information:<p>
Name: George Smith<br>
Addr: 45th Harris<br>
</body>
</html>
```



Beispiel Vlib

Barbara Zimmermann

Ausgangsbeispiel: Durchmischung PHP / HTML

```
01 <html>
…
06 <body>
07 <?php
08    if (isset($input))
09    {
10        echo "You typed <b>$input</b>.\n";
11    }
12    else
13    {
14        echo <<<form
15            <form action="form.php" method="post">
16                <input name="input">
17                <input type="submit">
18            </form>
19 form;
20    }
21 ?>
22 </body>
23 </html>
```

Beispiel Vlib

Barbara Zimmermann

form.php

```
01 <?php
02 require_once
"../vlibTemplate.php";
03 $tmpl = new
       vlibTemplate('tmpl/vlibT
       emplate_form.htm');
04
05 if (isset($input))
06 {
07  $tmpl->setVar('display',1);
08  $tmpl->setVar('input'
       ,$input);
09 }
10 else
11 {
12  $tmpl->setVar('display',0);
13 }
14     $tmpl->pparse();
15 ?>
```

form.htm

```
01 <html>
02 <head>
03 <body>
04
05 <tmpl_if name='display'>
06  You typed <b>{tmpl_var
       name='input'}</b>.
07 <tmpl_else>
08 <form
       action="vlibTemplate_form.
       php" method="post">
09 <input name="input">
10 <input type="submit">
11 </form>
12 </tmpl_if>
13
14 </body>
15 </html>
16
```

Beispiel Vlib

Barbara Zimmermann

Ausgabe vor Submit

```
<html>
<head>

</head>
<body>

<form
action="vlibTemplate_form.php"
method="post">
<input name="input">
<input type="submit">

</form>

</body>
</html>
```

Ausgabe nach Submit

```
<html>
<head>

</head>
<body>

You typed <b>Hallo</b>.

</body>
</html>
```

HTML_QuickForm

Barbara Zimmermann

HTML_QuickForm-Bibliothek

Generierung von HTML-Formularen

HTML-Seite

PHP-Skript

```
01 <?php
02 require_once 'HTML/QuickForm.php';
03 $myForm=new HTML_QuickForm('Formular',
       'POST');
04 $myForm->addElement('text', 'textName',
       'Name:');
05 $myForm->addElement('submit',
       'submitButton','Daten senden');
06 $name =& $myForm->getElement('textName');
07 $myForm->addRule('textName',’Pflichtfeld',
       'required');
08 if ( $myForm->validate() )
09 {print 'Ihre Angaben sehen wie folgt aus:';
10 $myForm->removeElement('submitButton');
11 $myForm->freeze();}
12 $myForm->display();
13 ?>
```

Fazit

- Template Engines
  - **à** Sinnvolle Methode zur
    - Codereduzierung
    - Steigerung der Übersichtlichkeit
- Generierung von Formularen
  - **à** Zur Ergänzung der Template Engines

è Strikte Trennung von Layout und Funktion

Fazit

- Nutzen für unsere Projektgruppe?

à Technik zur Implementierung

- Trennung von Funktion und Design
- Vereinfachung der Strukturen durch Einteilung des Codes in Teilgebiete
- Codereduzierung durch „Schablonen“

à Konzepte übernehmen

- Generatorkonzept erweitern
- Temporäres „Kompilieren“ von Projekten zur Laufzeitreduzierung

Danke

für Ihre Aufmerksamkeit!