virtUOS
Zentrum zur Unterstützung virtueller Lehre
der Universität Osnabrück

Marcus
Lunzenauer

Templates zur Vereinheitlichung der Ausgabeerzeugung

Stud.IP Entwicklerworkshop
Halle
29. März 2007

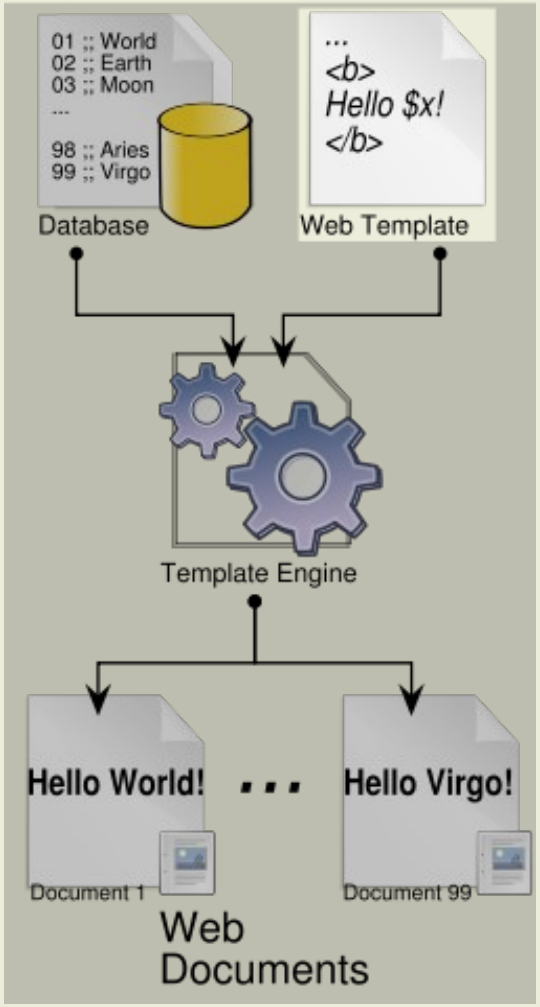
UNIVERSITÄT OSNABRÜCK

Inhalt

- Überblick über die Vorteile durch die Verwendung von Templates
- Darstellung von Beispielen aus Stud.IP
- Implementierung eines Template-Mechanismus
- Diskussion daraus resultierender Änderungen

Templates?

Schema: Template-Engine



Quelle: Wikipedia

Vorteile

Grundannahmen

- Web Templates beruhen auf zwei Grundannahmen:
  - Trennung von “business logic” und “presentation logic” ist gut.
  - Maximale Flexibilität im “presentation layer” ist notwendig.

Vorteile

- lose Kopplung der Programmkomponenten
  - Wartbarkeit
  - Anpassbarkeit
  - Lesbarkeit
- Berücksichtigung unterschiedlicher Fähigkeiten
- Entkopplung (räumlich/zeitlich) im Produktionsprozess
- kürzere Einarbeitungszeit für neue Entwickler

Verwendung

Verwendung

- Gegenwärtige Verwendung nur in Osnabrück:
  - Vips-Plugin
  - EvaSys-Plugin
  - Forschungsdatenbank (Anlegeassistent)
  - Osnabrücker Startseite

StEP00042

Flexi_Templates

StEP00042: Definition Templates

- Templatedateien ...
  - “Lückentext”
  - allerdings “Lückentext on Steroids”, d.h. inkl. Ausgabelogik
- ... und sie repräsentierende Objekte
  - Ablage von Templatewerten unter Schlüsseln

Und wieso Flexi?

- beliebige Template-Engines dank schmaler Abstraktion:
  - Flexi_TemplateFactory
  - Flexi_Template

Flexi_TemplateFactory


```
class Flexi_TemplateFactory {

  function Flexi_TemplateFactory($path);

  function get_path();
  function set_path($path);

  function open($template);
  function render($template[, $attributes[, $layout]]);
}
```

Flexi_Template I


```
class Flexi_Template {
  function get_attribute($name);
  function set_attribute($name, $value);
  function clear_attribute($name);

  function get_attributes();
  function set_attributes($attributes);
  function clear_attributes();

  function set_layout($layout);

  function render([$attributes[, $layout]]);
}
```

Flexi_Template II

- abstrakte Klasse
- derzeit implementierende Unterklassen:
  - Flexi_PhpTemplate: Template-Dateien in PHP

Templates erzeugen I

Zum Erzeugen von Templates benötigt man eine Factory:

```
$path = '/home/mlunzena/templates';
$factory =& new Flexi_TemplateFactory($path);
```

Templates öffnen II

In `~mlunzena/templates/` liegt eine Templatedatei `foo.php`

```
$template =& $factory->open('foo');
```

- Suche im Pfad nach Dateien foo.*
- Endung bestimmt konkrete Template-Engine (.php => Flexi_PhpTemplate)

Ein Flexi_PhpTemplate


```
<html>
  <head>
    <title><?= $title ?></title>
  </head>
  <body>
    <p><?= $title ?></p>
    <ul>
      <? foreach ($items as $item) : ?>
        <li><?= $item ?></li>
      <? endforeach ?>
    </ul>
  </body>
</html>
```

Attribute setzen


```
$template->set_attribute('title', "Meine Lieblingszahlen:");
$template->set_attribute('items', array(23, 42, 2147483647);


$template->get_attributes();

# liefert dann:
#
# array('title' => 'Meine Lieblingszahlen:',
#       'bar' => array(23, 42, 2147483647))
```

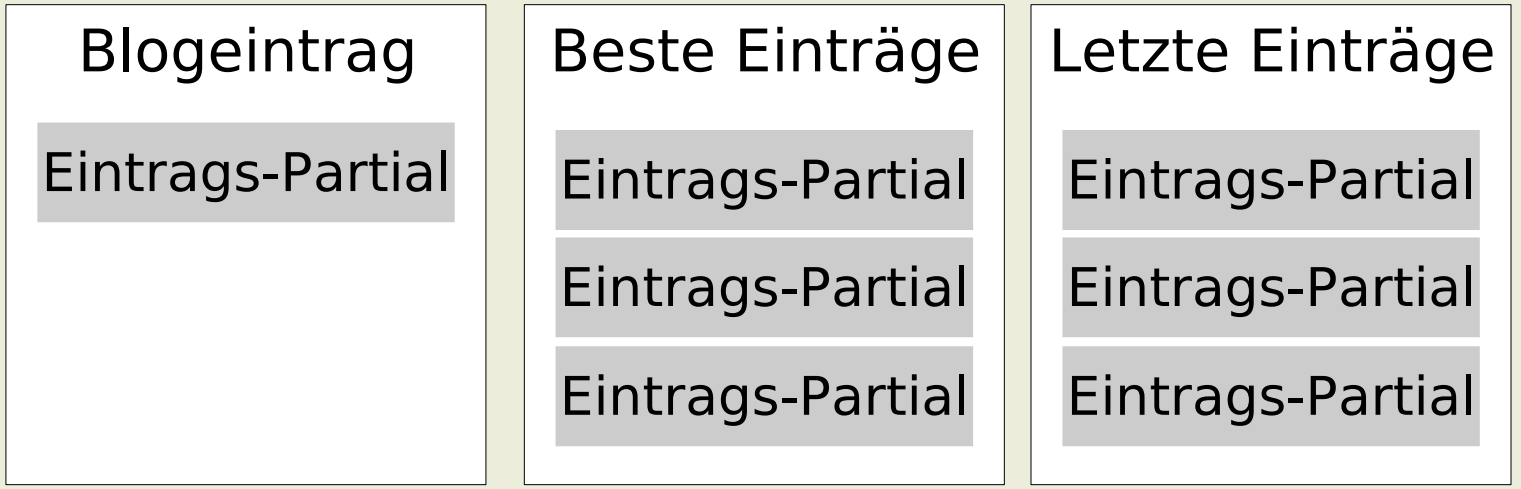
Template ausgeben


```
echo $template->render();

# <html>
#   <head>
#     <title>Meine Lieblingszahlen:</title>
#   </head>
#   <body>
#     <p>Meine Lieblingszahlen:</p>
#     <ul>
#         <li>23</li>
#         <li>42</li>
#         <li>2147483647</li>
#     </ul>
#   </body>
# </html>
```

Partials

- wiederverwendbare Schnipsel von Template-Code
- Beispiel: Blog-Applikation
- Partials sind auch nur normale Templates



Flexi_PhpTemplate#render_partia
| |


```
class Flexi_PhpTemplate extends Flexi_Template {
  function render_partial($partial
                          [, $attributes]);
}
```

- benötigt den Namen eines Partials
- kann optional zusätzliche Argumente erhalten

Flexi_PhpTemplate#render_partial II


```
# im einem Template foo.php
$this->render_partial('foobox',
         array('bar' => 23,
               'baz' => new XY(1, 2)));

# partial template: foobox.php
<div id="foobox">
  <h2>Somewhere <?= $foo ?></h2>
  <p>Beware! <?= $bar ?> of <?= $baz->chunky() ?>
   ahead!</p>
</div>
```

Layouts I

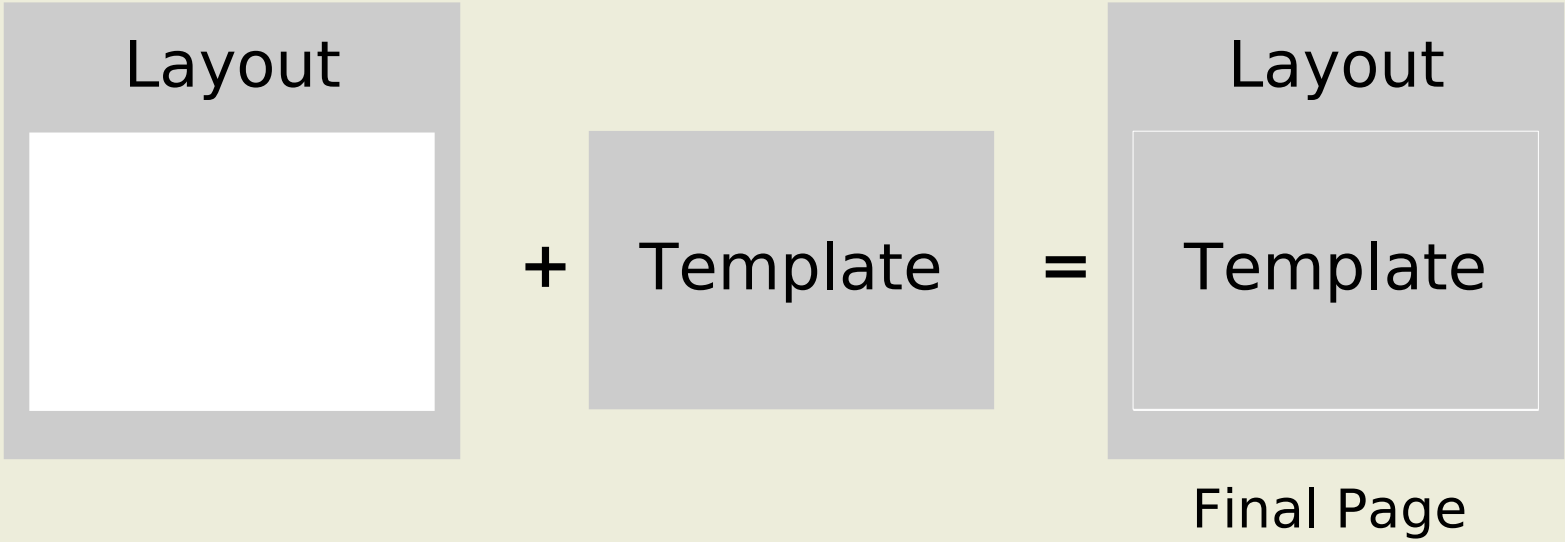
- häufig auftretendes Muster: Kopf- und Fußzeilen in vielen Templates

```
<?= $this->render_partial("shared/header") ?>
Hello World
<?= $this->render_partial("shared/footer") ?>
```

- einigermaßen gutes Mittel für DRY
- Probleme bei Änderung der grundlegenden Struktur von header/content/footer

Layouts II

- Decorator Pattern (Martin Fowler)
- Layout dekoriert Templates
- gemeinsame Struktur weiss nun, wo der Content hingehört



Layouts III

```
# layout template
header
<?= $content_for_layout ?>
footer

# example of a content template
Hello World

# nach dem rendern:
header
Hello World
footer
```

Layouts IV

- Layouts sind auch nur Templates
- Magic Word: $content_for_layout
- Layouts erben alle Attribute 'ihres' Content-Templates und alle darin gesetzten lokalen Variablen

```
$template->set_layout('my_chunky_layout');
```

Diskussion