Vorlesungsreihe
Entwicklung webbasierter Anwendungen

PHP– Erweiterungen
PEAR & DB-Interfaces
- Template-Engines

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

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

Gliederung

- Für komplexe Aufgabenstellungen im Bereich der webbasierten Datenverarbeitung sind einige größere Zusatzmodule oder Frameworks entstanden.
- Bei der Verwendung sind ggf. aktuelle Versionen zu analysieren und neue Module zu evaluieren.
  - PEAR / PECL
  - Template-Engines

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

PEAR – Einführung & Historie

PHP Extension and Application Repository (PEAR) **: (pear.php.net)**

- ist eine Sammlung von Modulen und Erweiterungen für PHP (ähnlich der CPAN-Bibliothek für PERL oder SWING-Framework in JAVA)
- Entw. seit 2001 durch Stig S. Bakken, seit 2003 Verwaltung durch PEAR Group.

Motivation / Ziele :

- Opensource-Referenzimplementierungen zu Webprogr.-Standardaufgaben, welche nach einheitlichen Standards erstellt, weiterentwickelt und getestet werden
- damit (meist) höhere Codequalität als eigenentwickelter Code,
- wiederverwendbare Basisobjekte und Pakete zur Redundanzminimierung im Code,

Technische Parameter : (http://pear.php.net/packages.php)

- ca. 600 Packages in 35 Kategorien <- 350 Package-Verwalter / 2500 Entwickler
- Installation mit PEAR-Installer (mit Kontrolle von Abhängigkeiten in den genutzten Paketen, aufwändige Qualitätsabstimmung mit PEAR Proposal-System („PEPr"), – Internetlink zum Installer : http://pear.php.net/go-pear

PECL-Bibliothek – Parallel-Bibliothek implementiert unter C/C++ http://pecl.php.net/

- verwendet gleiche Installationstools, teilweise performanter, aber weniger Module

PEAR – Module

Module (ca. 600 „Pakete“, engl. packages) – Hauptgruppen :

- Authentifizierung von Benutzern
- Automatisiertes Caching von Webseiten
- Verschiedene Datenbankzugriffsmodule
- Verschlüsselung
- Internationalisierung (Translator, Sprachversionsverwaltung)
- Konfiguration
- Grafikgenerierung (z.B. Barcodes, 3D-Grafiken ...)
- Payment-Module
- HTML-Generierung und Prüfung (Tidy)
- Web Services
- XML-RPC und XML

Neu: PEAR-Channels :

- Verteilte Portal-Website zum Management von PEAR-Channel-Server

Pear -Bsp. 1 - HTML_QuickForm (bzw. Quickform2)

- **HTML_QuickForm ist eine Bibliothek zum Handling von HTML-Formularen**
- **Unterstützung von Javascript and server-side-validation**
- Bsp.-Quelle : http://pear.php.net/manual/en/package.html.html-quickform.php

```
<?php // Load the main class require_once 'HTML/QuickForm.php';
// Instantiate the HTML_QuickForm object
$form = new HTML_QuickForm('firstForm'); // Set defaults for the form elements
$form->setDefaults(array(     'name' => 'Joe User' ))// Add some elements
$form->addElement('header', null, 'QuickForm tutorial example');
$form->addElement('text', 'name', 'Enter your name:', array('size' => 50,
 'maxlength' => 255));
$form->addElement('submit', null, 'Send'); // Define filters and validation rules
$form->applyFilter('name', 'trim');
$form->addRule('name', 'Please enter your name', 'required', null, 'client');
// Try to validate a form
if ($form->validate()) {     echo '<h1>Hello, ' . htmlspecialchars($form-
>exportValue('name')) . '!</h1>';     exit; }
// Output the form $form->display(); ?>
```

Bsp. 2 - Image – Generierung

- **Zeichnung beliebiger Grafiken mit Image_Canvas**

```
<?php require_once 'Image/Canvas.php';
$Canvas =& Image_Canvas::factory('png', array('width' => 400, 'height' => 300));
$Canvas->setLineColor('black');
$Canvas->rectangle(array('x0' => 0, 'y0' => 0, 'x1' => 399, 'y1' => 299));
$Canvas->setGradientFill(array('direction' => 'horizontal', 'start' => 'red', 'end' => 'blue'));
$Canvas->setLineColor('black');
$Canvas->ellipse(array('x' => 199, 'y' => 149, 'rx' => 50, 'ry' => 50));
 $Canvas->setFont(array('name' => 'Arial', 'size' => 12));
 $Canvas->addText(array('x' => 0, 'y' => 0, 'text' => 'Demo -> what Image_Canvas do!'));
$Canvas->addText(array('x' => 199, 'y' => 295, 'text' => '[Changing format is  done
 by changing 3 letters]', 'alignment' => array('horizontal' => 'center', 'vertical' => 'bottom')));
$Canvas->addVertex(array('x' => 50, 'y' => 200));
$Canvas->addVertex(array('x' => 100, 'y' => 200));
$Canvas->addVertex(array('x' => 100, 'y' => 250));
$Canvas->setFillColor('red@0.2');
$Canvas->polygon(array('connect' => true));
$Canvas->image(array('x' => 398, 'y' => 298, 'filename' => './pear-
icon.png', 'alignment' => array('horizontal' => 'right', 'vertical' => 'bottom')));
$Canvas->show();
?>
```

Bsp. 3 - Database- Abstraction Layer MD2

- **PEAR-Modul zur Unterstützung verschiedener Datenbanken (u.a. Oracle, Postgres, MSSQL , ...)**
- **Connect zu einer Postgres-Datenbank :**

```
require_once 'MDB2.php';
$dsn = 'pgsql://someuser:apasswd@unix(/tmp)/thedb';
options = array(     'debug'       => 2,     'portability' => MDB2_PORTABILITY_ALL, );
$mdb2 =& MDB2::factory($dsn, $options);
if (PEAR::isError($mdb2)) {     die($mdb2->getMessage()); }
```

- **Zusätzliche Methoden zum Manipulieren (hier Zeilen-Limit)**

(Quelle: http://pear.php.net/manual/en/package.database.mdb2.intro-query.php)

```
$sql = "SELECT * FROM clients";
$mdb2->setLimit(20, 10); // read 20 rows with an offset of 10
$affected =& $mdb2->exec($sql);
```

Weitere Alternativen zur MYSQL-Anbindung

SQLite

- „einfache“ Datenbankfunktionalität auf Dateibasis
- benötigt keinen getrennt laufenden DB-Server (günstig für PHP auf Provider-Server, da durch Datei-Upload „installierbar“)
- sehr schnell bei Leseoperationen, langsam (und durch Gesamt-Lock kritisch) bei vielen Schreiboperationen
- **wird von vielen Programmen im Hintergrund zur Datenspeicherung eingesetzt (Firefox, Apple – Mail, ...)**

AdoDB

- Anbindung in Richtung Microsoft´s ADO-Schnittstelle
- wird aber bereits bei PEAR nicht mehr gepflegt

PEAR-Systeme - PEAR::DB / PEAR::MDB2

- Datenbankinterfaces aus dem PEAR-System

PEAR – Verschiedenes ...

- **Universelles Payment-Modul - pear.php.net/package/Payment_Process**

```
$card = & Payment_Process_Type::factory('CreditCard');
if (!PEAR::isError($card)) {   // (hier mit Kreditkarte)
    $card->type = PAYMENT_PROCESS_CC_VISA;
    $card->invoiceNumber = 112345145;
    $card->customerId = 1461264151; $card->cardNumber = '411111111111';
     $card->expDate = '01/2005';  $card->zip = '48197'; $card->cvv = '768';
     $result = Payment_Process_Type::isValid($card);
     if (!PEAR::isError($result)) {   ...  // Bearbeitung der Kartenprüfung ...
```

- **Ausgabe von Text als Sprache** (http://pear.php.net/package/Text_Spell_Audio)

```
require_once 'Text/Spell/Audio.php';
$options = array('distort' => 1);
$a = new Text_Spell_Audio($options);
$a->output('abC123#');  /* Output wav file 'abC123#, */
```

- **Ausgabe einer Barcode-Grafik**

```
<?php require_once 'Image/Barcode.php';
Image_Barcode::draw('1234', 'int25', 'png'); ?>
```

Template – Engines unter PHP - Zusammenfassung

Allgemein

- Bei „normaler“ PHP-Programmierung wird HTML-Code und PHP gemixt
- Probleme beim Design komplexer Seiten durch bruchstückhaften HTML-Code, welcher mit HTML-Autorensystemen schlecht wartbar ist

Zielstellung

- Trennung der Präsentationsebene von der Programmlogik

Lösungsansatz:

- Erstellung einer HTML-Template-Datei, welche mit Platzhaltern gefüllt wird
- (an sich altes Prinzip - vgl. erste Beispiele zur Client-Server-Prog. aus der Mitte der 90iger Jahre mit Coldfusion ..)
- Aktuelle verfügbare Technologien (unter PHP)
  - Smarty-Engine (www.smarty.net) / Twig (twig.sensiolabs.org/)
  - Diverse andere Template-Engines VLIb, TBS (bzw. Eigenentwicklungen)
  - Code-Vergleich : http://gonzalo123.com/2011/01/17/php-template-engine-comparison/
  - Aktuell (2015) http://www.sitecrafting.com/blog/top-5-php-template-engines/

Template – Engines - Smarty

Allgemein

- Smarty ist eine Open-Source Template-Engine für PHP.
- relativ hohe Geschwindigkeit :
  - Vorlagen-Dateien werden eingelesen und es werden neue PHP-Scripte generiert werden, die temporären PHP-Scripte werden bei einem erneuten Seitenaufruf direkt von PHP geparst und dann an den Client geschickt.
  - dadurch auch Vorteile auch bei Compiler-Cache Lösungen (ZEND)
  - Optional kann Smarty auch den HTML-Output zwischenspeichern (starke Einsparungen bei statischen DB-Anfragen)

- **Kritik an Smarty**
  - Die Trennung von Logik und Präsentation wird nicht strikt eingehalten – so werden IF-Bedingungen doch wieder im HTML-Code kodiert ...
  - Bei sehr komplexen Seiten keine wesentlichen Einsparungen bei der Entwicklung – ggf. eher Problem mit Details ...

Template – Engine - Smarty - Beispiel 1

Ablage des HTML-Codes in der Datei tab1.tpl

```
table>  <tr>  <td>Name: {.. $name ..} Vorname: {.. $first_name ..}
      </td>  </tr>  </table>
```

(Einmalige) PHP-Code mit Smarty-Anbindung und Analyse der Datei

```
<?php require_once('/smarty/libs/Smarty.class.php');
$my_smarty = new Smarty();
$my_smarty->security = true;
$my_smarty->secure_dir = '/templates';
$my_smarty->compile_dir = '/smarty_cache';
$my_smarty->left_delimiter = '{..';     $my_smarty->right_delimiter = '..}';
$my_smarty->debugging = false; ?>
```

PHP-Code mit Variablenzuweisung und Templateanalyse

```
<?php $my_smarty->assign('name', $name);
my_smarty->assign('first_name', “tom“); // oder   $first_name);
$template->display('tab1.tpl');    ?>
```

Template – Engine - Smarty - Beispiel 2 mit if then

Ablage des HTML-Codes in der Datei tab2.tpl

```
<p style="color:  {if $color == "purple"} purple {else} gray  {/if}">
   This text is gray without "color" set. Click
    <a href="tmpl_if.php?color=purple">here to set "color"</a>
     to make it purple.</p> <p style="font-size: 1.5em">

if_condition: > {$if_condition}<</p> {if $if_condition == 1} <p style="color:
    green">"if_condition" is equal to 1.</p>
{elseif $if_condition > 5} <p style="color: red">"if_condition" is more than
    5.</p> {else} <p>Smarty ELSE is used, when "if_condition" is not 1
    or more than 5, to set it <a href="tmpl_if.php?if_condition=10">
    click here</a>.</p> {/if}
```

PHP-Code mit Variablenzuweisung und Templateanalyse

```
<? $smarty->assign('color', $_GET['color']);
    $smarty->assign('if_condition', $_GET['if_condition']);
    $smarty->display(,tab2.tpl‘);
```

Template – Engines unter PHP

Allgemeiner Stand

- Bisherige Template-Engines haben sich noch nicht sehr stark durchgesetzt
- Einige größere Frameworks wie Content-Managementsysteme nutzen Templates – Engines, z.B. Drupal und Wordpress das Twig-System
- Aktuelle Bewertung: **https://www.sitecrafting.com/top-5-php-template-engines/**
- mit Twigs als Spitzenreiter **http://acmeextension.com/best-templating-engine/**

PRO´s

- Meist eine Vereinfachung / Reduzierung des notwendigen Codes
- Automatisierung von Standardaufgaben (wie Escaping, Sicherheit o.ä.)

Contra´s

- neue Syntax ist zu lernen
- Dokumentation / langfristige Wartung ggf. kritisch

Fazit

- für eine verteilte Arbeit an größeren Projekten mit starker Spezialisierung (Trennung von Design / Programmierung in Persona) sind Templates Engines wahrscheinlich am besten geeignet