PHP Fortgeschrittenen Kurs Sommercampus 2006

09.08.2006

Christoph Hermann, Dominik Benz
{hermann,dbenz}@informatik.uni-freiburg.de

Pear Quickform

- Einfache Formulargenerierung
- Vorbelegung der Felder möglich
- Validierung der Formularfelder (Server- und Clientseitig)
- Komplexe Regelerstellung möglich
- Hierarchical Selects möglich

Pear Quickform Beispiel

```php
<?php
require_once "HTML/QuickForm.php";
$user = array("lastname"=>"Mouse","sal_id"=>4,"user_id"=>7);
$salutations = array("0"=>"Mr", "1"=>"Miss",..., "4"=>"Sir");
$form = new HTML_QuickForm('frmTest', 'get');
$form->addElement('header', 'MyHeader', 'Edit␣username');
$form->addElement('hidden', 'user_id');
$form->addElement('select', 'sal_id',
        'Address␣me␣as:', $salutations);
$form->addElement('text', 'lastname', 'Last␣name:');
$form->addRule('lastname',
        'Your␣lastname␣is␣required', 'required');
$form->addElement('reset', 'btnClear', 'Clear');
$form->addElement('submit', 'btnSubmit', 'Submit');
if ($form->validate()) {
  // Formular ist validiert, daten verarbeiten
  $form->freeze(); $form->process('process_data', false);
}
$form->setDefaults($user);
$form->display();
?>
```

Pear Quickform - Datenauswertung

```php
<?php
function process_data ($values) {
        echo "<pre>";
        foreach ($values as $key=>$value) {
                        echo $key."=".$value."<br>";
        }
        echo "</pre>";
}
?>
```

.

Smarty

- Smarty ist eine Template-Engine für PHP.
- Erlaubt es die einfache Trennung von Applikations-Logik und Design/Ausgabe.
- Sehr schnell im Gegensatz zu anderen Template-Engines
- Unbegrenzte Verschachtelung von 'section', 'if' und anderen Blöcken.
- Konfigurierbare Syntax für Template-Tags: `{}`, `{{}}`, `<¸ ¸>`,! ...

Einfaches Smarty Beispiel

```
{* Smarty *}
<table>
<tr>
<td class="text">
{section name=count loop=$news}
        {if $smarty.section.count.index % 5 == 0}
                </td><td class="text">
        {/if}
        <a href="news.php?id={$news[count].id}">
        <b>{$news[count].thema}</b>
        </a><br />
{/section}
</td>
</tr>
</table>
```

Smarty Beispiel

- Eingebauter Caching-Support!

```
<?php
require('Smarty.class.php');
$smarty = new Smarty(); $smarty->caching = true;
if(!$smarty->is_cached('index.tpl')) {
    // kein Cache gefunden, also Variablen zuweisen
    $contents = get_database_contents();
    $smarty->assign($contents);
}
$smarty->display('index.tpl');
?>
```

- Per-User Caching z.b. durch cacheids möglich:

```
<?php
require('Smarty.class.php');
$smarty = new Smarty(); $smarty->caching = true;
// ... cacheid erzeugen
$smarty->display('index.tpl',$cacheid);
?>
```

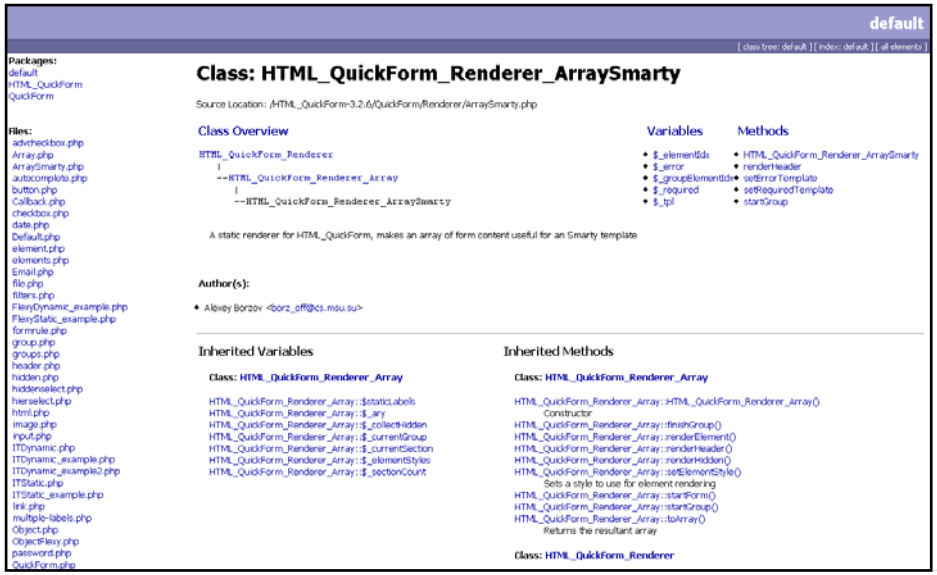
.

Pear Quickform Controller (Multipage Forms)

- Vorteile von Quickform
- Mehrseitige Formulare
- Vor- / Zurückblättern
- Integration mit Smarty auch möglich: `smarty-dynamic.tpl`
  Siehe QuickForm docs: `docs/renderers/templates`
- Code rel. komplex (lang), daher hier kein Beispiel

phpDocumentor

- phpDoc(umentor) ist ein Dokumentationsstandard der JavaDoc sehr ähnelt.
- Installation 1/2
  - Download&unzip: http://pear.php.net/package/PhpDocumentor/download
  - PHP konfigurieren: `$ setup lang/php5`
  - `$ which php`
    /usr/local/lang/php/5.1.4/bin/php




```
239.
240.
241.    /**
242.     * Creates an array representing an element
243.     *
244.     * @access private
245.     * @param  object    An HTML_QuickForm_element object
246.     * @param  bool      Whether an element is required
247.     * @param  string    Error associated with the element
248.     * @return array
249.     */
250.    function _elementToArray(&$element, $required, $error)
251.    {
252.        $ret = array(
253.            'name'      => $element->getName(),
254.            'value'     => $element->getValue(),
255.            'type'      => $element->getType(),
256.            'frozen'    => $element->isFrozen(),
257.            'required'  => $required,
258.            'error'     => $error
259.        );
260.        // render label(s)
261.        $labels = $element->getLabel();
262.        if (is_array($labels) && $this->_staticLabels) {
263.            foreach($labels as $key => $label) {
264.                $key = is_int($key)? $key + 1: $key;
265.                if (1 === $key) {
266.                    $ret['label'] = $label;
267.                } else {
268.                    $ret['label_' . $key] = $label;
269.                }
270.            }
271.        } else {
272.            $ret['label'] = $labels;
273.        }
274.
275.        // get the style for the element
276.        if (isset($this->_elementStyles[$ret['name']])) {
277.            $ret['style'] = $this->_elementStyles[$ret['name']];
278.        }
279.        if ('group' == $ret['type']) {
280.            $ret['separator'] = $element->_separator;
281.            $ret['elements']  = array();
282.        } else {
283.            $ret['html']      = $element->toHtml();
284.        }
285.        return $ret;
286.    }
287.
```

phpDocumentor

- Installation 2/2
- Eclipse konfigurieren: `$ setup devel/eclipse`
- Integration in (PHP)Eclipse:
  - Eclipse → Run → External Tools → External Tools
  - Program → New
  - Name: PHPDocumentor
  - Location: /usr/local/lang/php/5.1.4/bin/php
  - Arguments: `path/to/phpDocumentor/phpdoc.inc -t ${project_loc}/doc -o HTML:Smarty:PHP -d ${project_loc} -s on`
  - Tab Refresh: [x] Refresh resources upon completion
    (x) The Project containing the selected resource
    [x] Recursively include subfolders

PHPUnit

PHPUnit installieren

- Download: http://pear.php.net/package/PHPUnit2/download
- phpunit Skript vorbereiten:
  - `pear-phpunit` Skript in `phpunit` umbenennen
  - Den String `@php_bin@` durch den Pfad zu PHP ersetzen
  - phpunit skript ausführbar machen
- `@package_version@` in `PHPUnit2/Runner/Version.php`
  durch die Versionsnummer der PHPUnit2 Version ersetzen die heruntergeladen wurde
- `phpunit UnitTest UnitTest.php`
  Führt die Tests der Testfall-Klasse `UnitTest` aus.

PHPUnit Beispiel

```
<?php
require_once 'PHPUnit2/Framework/TestCase.php';
class ArrayTest extends PHPUnit2_Framework_TestCase {
    protected $arr;
     protected function setUp() {
        $this->arr = Array();
    }
    public function testNewArrayIsEmpty() {
        // Der erwartete Wert von sizeof($this->arr) ist 0.
        $this->assertEquals(0, sizeof($this->arr));
    }
    public function testArrayContainsAnElement() {
        // Ein Element dem Array hinzufügen.
        $this->arr[] = 'Element';
         // Der erwartete Wert von sizeof($this->arr) ist 1.
        $this->assertEquals(1, sizeof($this->arr));
    }
}
?>
```

php.ini

Konfigurations-Datei für PHP: php.ini

- befindet sich oft in /etc/php5/apache2/php.ini; Pfad wird auch in `phpinfo()` angezeigt
- Einstellungen können in dieser Datei vorgenommen werden oder dynamisch zur Laufzeit per Funktion !ini_set! (nicht alle, siehe http://www.php.net/manual/en/ini.phpini.list
- Veränderungen an php.ini werden erst nach Neustart des Webservers wirksam

`register_globals <boolean>` schaltet an / aus, ob per POST/GET übergebene Variablen automatisch global verfügbar sind. Sollte ausgeschaltet sein!

`post_max_size <integer>`definiert die maximale Datenmenge in Bytes, die per POST hochgeladen werden kann (auch Dateien). Muss für grosse Dateien grösser sein als `upload_max_filesize <integer>`

php.ini - 2

`register_globals <boolean>` schaltet an / aus, ob per POST/GET übergebene Variablen automatisch global verfügbar sind. Sollte ausgeschaltet sein!

`memory_limit <integer>`legt den maximalen Speicherverbrauch eines Skripts fest. Der Standard von 8 MB muss für einige Anwendungen, besonders CMS, hochgesetzt werden

`include_path <string>`gibt eine Liste von Pfaden an, in denen nach einzubindenen Dateien gesucht wird. Praktisch für Libraries wie PEAR

`file_uploads <boolean>` erlaubt / verbietet das Hochladen von Dateien

`upload_tmp_dir <string>` gibt den Pfad an, in dem hochgeladene Dateien zwischengespeichert werden

php.ini - 3

`session.save_path <string>` legt den Pfad fest, in dem Session-Daten gespeichert werden

`session.use_trans_sid <boolean>` erlaubt / verbietet die Übergabe von Session-IDs per URL (Sicherheitsrisiko!)

- weiter Direktiven: in den jeweiligen Kapiteln im Handbuch

XML

- Extensible Markup Language
- mächtiger Mechanismus zum Speichern, Verarbeiten und Austauschen von (semi-)strukturierten Daten
- viele Techniken "drumherum":
  - DTD / XMLSchema: Definition der erlaubten Struktur eines XML-Dokuments
  - XPath: Sprache zum Adressieren von Teilen eines XML-Dokuments
  - XQuery / XSLT: Transformation von XML
  - XPointer, XUpdate, ...
- verbreitetste, zukunftssicherste, flexibelste Art, strukturierte Daten zu repräsentieren

XML in PHP

- Seit PHP5 einiges mehr an XML-Unterstützung
  - SimpleXML-Funktionen:

```
$sxe = simplexml_load_file('text.xml');
foreach ($sxe->elem as $name => $data)
  print "$name -> $data";
}
```

  - DomDocument-Klasse

```
$dom = new DomDocument();
$root = $dom->createElement("elem");
$text = $dom->createTextNode("content);
...
```

PEAR: XML_Serializer

- Klasse zum Umwandeln von PHP-Datenstrukturen (Arrays) in XML (Serialisieren)
- umgekehrt auch möglich (XML -> Array, Deserialisieren)

```
require_once 'XML/Serializer.php';

$palette = array('red', 'green', 'blue');

$serializer_options = array (
   'encoding' => 'ISO-8859-1',
   'indent' => '  ', ...
);

$Serializer = &new XML_Serializer($serializer_options);
$status = $Serializer->serialize($palette); // after this: error
checking
header('Content-type: text/xml');
echo $Serializer->getSerializedData();
```

siehe: http://www.sitepoint.com/article/xml-php-pear-xml_serializer

XML: Übungsvorschlag

- Einlesen einer XML-Datei (z.B. SVN-History im XML-Format)
- Generieren eins RSS-Feeds daraus
  - üblicher Weg: Transformation per XSLT
  - auch möglich: deserialisieren in Array, umwandeln, serialisieren
- Integration des RSS-Feeds als LiveBookmark in Firefox