Das Portal für Familien-WGs, Generationenwohnen und alternative Wohnkultur

Ein Projekt zum Modul Web-Programmierung

Fachhochschule Brandenburg
Onlinestudiengang Medieninformatik

SoSe 2015

Ein Projekt von

Ulrike Exner

1. Aufgabenstellung

Als Ergebnis des Moduls Web-Programmierung sollte eine Web2.0-Anwendung erstellt werden, die folgende Anforderungen erfüllt:

- Verwendung von AJAX
- Verwendung von HTML5, CSS3 und Javascript
- Integration RSS-Feeds
- Mashups

Bei Verwendung von fremden Materialien (Fotos, Videos, Audiosequenzen) muss dabei vor allem auf Urheberrechte geachtet werden. Sie können sich aber auch auf Ressourcen bei Flickr oder YouTube verlinken.

2. Themenfindung

Als Projekt wählte ich ein interaktives Portal zum Thema „Alternatives Wohnen“. Die Idee dazu gab es schon länger, nun bot sich die Gelegenheit, einen ersten Prototypen umzusetzen.

3. Projektbeschreibung

URL: http://summer-song.de/wohn/

Entstehen soll der Prototyp eines Kleinanzeigenportals für Wohnungsanzeigen in *einfacher* Ausführung: Das Erstellen von Anzeigen soll für jeden Besucher ohne Login möglich sein.

Die Begrüßungsseite enthält einen Slider, der drei Bilder in Dauerschleife verblendet.

Über ein Webformular kann der Nutzer seine *Suche/Biete*-Anzeige inkl. Foto hochladen.

Die Ausgabe der Anzeigen erfolgt über einen anderen Menüpunkt als Tabellenübersicht. Zu jedem Item (= Anzeige) soll durch Anklicken des Titels eine Detail-Ansicht möglich sein. Die Detailansicht enthält einen Googlemaps-Ausschnitt, angelehnt an den ausgewählten Bezirk, ggf. das hochgeladene Bild sowie ein Kontaktformular.

Darüber hinaus existiert die Unterseite Blog, die über aktuelle Themen im Umfeld "Alternatives Wohnen" informieren soll. (Diese ist vorerst nicht als CMS umgesetzt und daher auch nicht als RSS-Feed umgesetzt, was naheliegen würde.)

Es gibt einen Zugriff auf einen RSS-Feed, der momentan noch manuell gepflegt werden muss. Er enthält die neuesten Anzeigen.

4. Umsetzung

4.1 Backend / My-SQL-Datenbank

Grundlage des Anzeigenbereichs ist eine My-SQL-Datenbank bestehend aus zwei Tabellen **biete_table** für die Anzeigen und **uploads2**für die hochgeladenen Bilder. Verknüpft sind diese über f_id (**uploads2)** - Bei Anlegen eines neuen Datensatzes wird f_id automatisch der gleiche Wert wie biete_id (**biete_table)** übergeben.

Die Tabelle **uploads2** enthält nicht das Bild selbst, sondern nur den Pfad zum uploads-Ordner im Server-Dateisystem (**uploads**). Um zu verhindern, dass sich Bilddateien mit dem gleichen Namen (z. B. 1.jpg) gegenseitig überschreiben, wird eine Methode zu Generierung eines zufälligen Dateinamens implementiert.

Die beiden folgenden Abbildungen zeigen die Spalten und deren Eigenschaften:

Anzeigen | Struktur | SQL | Suche | Einfügen | Exportieren | Importieren

#	Name	Typ	Kollation	Attribute	Null	Standard	Extra
1	biete_id	int(4)			Nein	*kein(e)*	AUTO_INCREMENT
2	biete_titel	varchar(255)	latin1_swedish_ci		Nein	*kein(e)*	
3	biete_wohnraumart	varchar(255)	latin1_swedish_ci		Nein	*kein(e)*	
4	biete_bezirk	varchar(255)	latin1_swedish_ci		Nein	*kein(e)*	
5	biete_beschreibung	text	latin1_swedish_ci		Nein	*kein(e)*	
6	biete_name	varchar(255)	latin1_swedish_ci		Nein	*kein(e)*	
7	biete_email	varchar(255)	latin1_swedish_ci		Nein	*kein(e)*	
8	biete_date	timestamp			Nein	CURRENT_TIMESTAMP	

Abbildung 1 Tabelle "biete_table"

Anzeigen | Struktur | SQL | Suche | Einfügen | Exportie

#	Name	Typ	Kollation	Attribute	Null	Standard	Extra
1	id	int(11)			Nein	*kein(e)*	AUTO_INCREMENT
2	name	varchar(30)	latin1_swedish_ci		Nein	*kein(e)*	
3	type	varchar(30)	latin1_swedish_ci		Nein	*kein(e)*	
4	size	int(11)			Nein	*kein(e)*	
5	path	varchar(60)	latin1_swedish_ci		Nein	*kein(e)*	
6	f_ID	int(11)			Ja	*NULL*	

Abbildung 2 Tabelle "uploads2"

Angaben zur Datenbankanbindung befinden sich in der Datei **include/config.inc.php**. Sie wird dort, wo sie gebraucht wird, inkludiert.

Zum Ein- und Auslesen der Datenbank wird PHP mit integrierten My-SQL-Abfragen verwendet.

4.2 Technologien, Dateien, Code-Beispiele

Das Frontend wurde in HTML5, CSS3 und Javascript umgesetzt. Es kann nicht als responsiv bezeichnet werden. Als praktikabel hat sich die Nutzung der Javascript-Bibliothek jquery in der neuesten Version 2.1.4 erwiesen.

Das Grundgerüst wird im folgenden Schema dargestellt: Die Idee des Grundgerüsts ist ein Design, das drei Hauptelemente als div-Layer (Header, Inhalt, Footer) bereitstellt. Header und Footer spiegel n sich in etwa in den gleichnamigen .php-Dateien wider, die in fast allen anderen .php-Dateien inkludiert werden.

Im Header befinden sich allgemeine Angaben und vor allem die wichtigen Verweise zur .css-Datei und zur extern gehosteten Javascript-Bibliothek jquery. Zudem inkludiert er basics.php, in der einige HTML-Elemente ausgelagert werden.

```
<?php
        $searchicon = "<img src='images/iconmonstr-magnifier-4-icon-80.png'
alt=''>";
        $contactlink ="<b onclick='$.fn.overlay();'>Kontaktieren</b>";
        $url = "http://www.summer-song.de/wohn/detailview.php?id=";

?>
```

Der Inhaltsbereich wird stets mit **div id="container"** umschlossen. Darin befinden sich **sections** <section></section> die in HTML5 inhaltliche Bereiche voneinander abgrenzen. Darin wiederum sind **articles** <article></article> eingebettet. Diese werden per CSS auf verschiedene Weise gestaltet: Das CSS ermöglicht z. B. drei Artikel nebeneinander oder vier Artikel nebeneinander (entsprechend kleiner) zu setzen, die sich über die gesamte Breite des Hauptinhalts strecken. So können unterschiedlich gestaltete **sections** entstehen. Ein anderes Gestaltungsmerkmal sind **figures**, die im vorliegenden Design auf der Einstiegsseite verwendet werden und hier dazu dienen, eine zusätzliche Navigation bereitzustellen.

Der Footer beinhaltet lediglich eine Copyright- und Validierungsangabe.

index.php

Die Einstiegsseite begrüßt den Besucher mit einem Bilder-Slider: Drei gleichformatige Fotos werden per Dauerschleife ineinander verblendet. Die Fotos werden dem div-Layer “Slideshow” zugeordnet. Der Code verwendet die Javascript-Bibliothek jquery.

```
<script>
    $("#slideshow >div:gt(0)").hide();

setInterval(function() {
    $('#slideshow >div:first')
    .fadeOut(1000)
    .next()
    .fadeIn(1000)
    .end()
    .appendTo('#slideshow');
},  3000);
</script>
```

anzeigen-biete.php

Zentral sind auf dieser Seite zwei Elemente: Die freie Suche und die (nicht sortierbare) Darstellung der Anzeigen.

Die freie Suche

Die freie Suche wird in Zusammenspiel von PHP, Javascript und AJAX gelöst. Sie sucht in allen Feldern aller Anzeigen, die Text enthalten und gibt das Ergebnis in einem überlappenden div-Layer darunter aus. Die Ergebnisliste ist klickbar - d.h. man gelangt direkt auf eine Anzeige, wenn man mit dem Suchergebnis zufrieden ist.

Das HTML-Grundgerüst:

```
<formid="searchform" method="post">
<inputtype="text" name="search_query" id="search_query"
placeholder="Wonachsuchst Du?" size="50"/>


</form>
<divid="display_results">
</div>
```

Javascript / AJAX:

```
<scripttype='text/javascript'>
        $(document).ready(function(){
            $("#search_results").slideUp();
            $("#button_find").click(function(event){
event.preventDefault();
search_ajax_way();
            });
            $("#search_query").keyup(function(event){
event.preventDefault();
search_ajax_way();
            });

        });

functionsearch_ajax_way(){
            $("#search_results").show();
varsearch_this=$("#search_query").val();
            $.post("search.php", {searchit : search_this}, function(data){
                $("#display_results").html(data);

            })
        }
</script>
```

PHP

```
$term = strip_tags(substr($_POST['searchit'], 0, 100));
$term = mysql_real_escape_string($term);
if ($term == "")
    echo "Gib ein Suchwort ein.";
else {
    $query  = mysql_query("select * from biete_table where biete_titel like
'%{$term}%' OR biete_bezirk like '%{$term}%' OR biete_beschreibung like
```

```
'%{$term}%'", $dbconnection);
    $string = '';

    if (mysql_num_rows($query)) {
        while ($row = mysql_fetch_assoc($query)) {
            echo "<p>&#8226; " . '<a href="detailview.php?id=' .
$row['biete_id'] . '">' . $row['biete_titel'] . '</a> ';
        }

    } else {
        $string = "Keine Treffer!";
    }
```

Ausgabe der Anzeigen in Listenansicht

Nachfolgend ist die Schleife zur Ausgabe der Anzeigen in der MySQL-Datenbank dokumentiert. Zur besseren Übersichtlichkeit wurde außerdem eine PHP-Struktur implementiert, die die Anzeigen pro Seite auf 5 begrenzt.

```
while ($row = mysql_fetch_assoc($result)) {
    echo "<article class='three_third'>";
    echo $searchicon;
    echo "<span class='em'>" . $row['biete_wohnraumart'] . " in Berlin - "
. $row['biete_bezirk'] . "</span>";
    echo "<h2>" . '<a href="detailview.php?id=' . $row['biete_id'] . '">' .
$row['biete_titel'] . ' </h2></a> ';
    echo "<span class='am'>am " . date("d.m.Y",
strtotime($row['biete_date'])) . " von " . $row['biete_name'] . "
gepostet</span>";
    echo "<hr>";
```

detailview.php

Zur Darstellung der Inhalt einer einzelnen Anzeige wird folgende PHP-Anweisung ausgeführt:

```
$id = $_GET['id'];
$strSQL = " SELECT * FROM biete_table WHERE biete_id='$id' ";
$rs = mysql_query($strSQL);

if ( ! $strSQL )                          {
    die('Ungültige Abfrage: ' . mysql_error());
}

while($row = mysql_fetch_array($rs)) {
    // Schreibe den Wert der Spalte Vorname (der jetzt im Array $row ist)
    echo "<article class='three_third'>";
    echo "<h1>". $row['biete_titel'] . "</h1>";
    echo $searchicon;
    echo "<span class='em'>" . $row['biete_wohnraumart'] . " in Berlin - "
. $row['biete_bezirk'] . "</span>";
    echo "<p>". $row['biete_beschreibung'] . "</p>";
    echo "<footer class='more'>" . $row['biete_name']. " " . $contactlink .
"</footer>";
}

$query = "SELECT path FROM upload2 WHERE f_id = '$id'";
$result = mysql_query($query) or die('Error, query failed');
```

```
while($row = mysql_fetch_array($result)) {
    $filePath = $row['path'];
    echo '<img src="' . $filePath . '" width="200" /> ';
}
```

Mit $_GET wird hier der Parameter 'id' aus der vorher vom Besucher aufgerufenen URL entnommen und verwendet, um die richtige Anzeige aus der MySQL-Datenbank zu filtern.

Da **biete_id** und **f_id** aus den beiden MySQL-Tabellen gleich sind (s. 4.1) kann somit auch das dazugehörige Bild aus der Datenbank entnommen werden.

Zudem ist hier eine Googlemaps-Karte per API eingebunden. Das Script zeigt ausgehend vom in der Datenbank hinterlegten Bezirk (hier könnte natürlich auch eine genauere Adresse hinterlegt werden) den Ort auf der Karte an.

Zu guter Letzt kann der Anzeigenersteller per Kontaktformular kontaktiert werden. Hierzu dient **contact.php**. Die .php-Seite enthält nach Absenden des Formulars eine Validierung per **send_form_email.php** und sendet danach eine Email an die in der Anzeige hinterlegten Email-Adresse.

biete.php

Auf dieser Seite befindet sich das zentrale Eingabeformular für das Einstellen von Anzeigen. Es besteht je nach abgefragtem Bestandteil aus einfachem Eingabefeld (Anzeigentitel, Name, E-Mailadresse), erweitertem Eingabefeld (Anzeigenbeschreibung), voreingestellten Auswahloptionen (Wohnraumart, Bezirk) sowie einem Button, der es ermöglicht ein Bild hochzuladen.

Wichtig ist die **method** und **action.** Darin wird festgelegt, was nach Abschicken des Formulars mit den Formulardaten geschehen soll: Mit **post** werden die Daten an den Server geschickt und mit **action** die Seite **biete_check_php** aufgerufen.

Das Attribut**multipart/form-data** sorgt dafür, dass das Bild mit geschickt wird.

```
<formmethod="POST" action="biete_check.php" enctype="multipart/form-data">

…

        <inputname="submitted" type="submit" value="Speichern"/>
        <inputtype="reset" value="Eingaben l&ouml;schen"/>
</form>
```

Einige HTML5- Funktionen sorgen dafür, dass das Formular client-seitig geprüft wird: So legt das Attribut **required** fest, welches die Pflichtfelder im Formular sind und das Attribut **email**sorgt dafür, dass der Nutzer darauf hingewiesen wird, wenn er keine valide Emailadresse eingibt. Dies bietet jedoch keinen Schutz vor böswilligen Attacken.

Was passiert nach Absenden des Formulars? / biete_check_php

In biete_check.php wird **process_biete_form.php** inkludiert. Zum Schutz vor Attacken mittels *SQL-Injection* wurde eine serverseitige Formularvalidierung mittels regulärer Ausdrücke implementiert. Erst nach Passieren der Validierung werden die Formulardaten an die MySQL-Datenbank übertragen bzw. das Bild in den dafür vorgesehenen Dateiordner hochgeladen.

Der reguläre Ausdruck für das Feld *biete_beschreibung* ist beispielsweise:

```
$pattern            = "/^[a-zA-Z0-9\_]{2,500}/";
```

Benutzt der Anwender verbotene Sonderzeichen, erhält er eine entsprechende Fehlermeldung, z. B. *Bitte verwende keine Sonderzeichen in der Anzeigenbeschreibung* und muss zurückkehren zum Formular.

Die Datenübernahme in die MySQL-Datenbank erfolgt mitder folgenden Anweisung:

```
    $eintrag = "INSERT INTO biete_table
(biete_titel, biete_wohnraumart, biete_bezirk, biete_beschreibung,
biete_name, biete_email)

VALUES
('" . mysql_real_escape_string($biete_titel) . "', '" .
mysql_real_escape_string($biete_wohnraumart) . "', '" .
mysql_real_escape_string($biete_bezirk) . "', '" .
mysql_real_escape_string($biete_beschreibung) . "', '" .
mysql_real_escape_string($biete_name) . "', '" .
mysql_real_escape_string($biete_email) . "');";

    $eintragen = mysql_query($eintrag) or die(mysql_error());
```

blog.php

Dies ist eine einfach aufgebaute Seite, die als Newsbereich fungieren könnte. Es ist jedoch kein CMS installiert und auch kein RSS-Feed angebunden.

uber.php

Eine einfache Seite mit Link zu dieser Projektdokumentation.

feed/feed.xml

Der RSS-Feed kann über http://summer-song.de/wohn/feed/feed.xml aufgerufen werden und manuell gepflegt werden. Momentan sind die neuesten Anzeigen enthalten.

images/..

In diesem Ordner befinden sich die von der Website verwendeten Bilder und Icons.

6. Fremdquellen für Code-Snippets und Grafiken

- Slider auf der Homepage

Simple Auto-PlayingSlideshow
https://css-tricks.com/snippets/jquery/simple-auto-playing-slideshow/

- Live-Suchfeld bei Anzeigen

Create Ajax Based Searching System with PHP and jQuery
http://www.infotuts.com/create-ajax-based-search-system-php-jquery/

- **Overlay-Effekt beim Kontaktformular**

jQuery Overlay
http://frontend.pro/javascript/jquery-overlay

- **Kontaktformular**

Simple Form to Email PHP Contact Form
http://www.freecontactform.com/email_form.php

- Google Maps einbinden per API

Google Maps Karte einbinden ohne Plugin (API V3)
http://calyrium.org/2012/04/13/google-maps-karte-ohne-plugin-api-v3/

- Icons

iconmonstr
http://iconmonstr.com/

- Bilder

https://pixabay.com/de/haus-hinterhof-b%C3%A4ume-durch-garten-690836/

https://pixabay.com/de/bootshaus-wochenendhaus-gew%C3%A4sser-192990/

http://www.splitshire.com/smiling/

http://www.splitshire.com/friendship/

http://www.morguefile.com/archive/display/918161

(alle unter CC0 Public Domain - Lizenz)