JUST DO IT
HÖHERE ABTEILUNG FÜR INFORMATIONSTECHNOLOGIE

tgm
Die Schule der Technik

Arbeitsprotokoll

WordPress PHP Plugin

INSY - CMS
4CHITT 2015/16

Daniel May

Version 1.0

Note:

Begonnen am 10. März 2016

Betreuer: Christoph Roschger

Beendet am 23. März 2016

Inhaltsverzeichnis

1 Einführung

„Stoesst man bei der Verwendung eines CMS an die Grenzen der verfuegbaren Erweiterungen (Plugins,Extensions,Module,...), bieten alle gaengigen CMS Systeme Schnittstellen zur Anpassung an. Zum Beispiel stellt die Integration externer Datenquellen eine solche Herausforderung dar.“ [ROS]

1.1 Ziele

„Es soll eine Wordpress-Seite aufgesetzt werden, die Daten aus der SchokoDB anzeigen kann, sowohl statisch als auch interaktiv abhaengig von Benutzereingaben.“ [ROS]

1.2 Aufgabenstellung

„Installiere und konfiguriere eine Wordpress-Seite fuer unsere Schokofabrik. Die Seite muss keine besonderen statischen Inhalte aufweisen.

Entwickle ein Wordpress-Plugin [1], dass du verwendest, das auf zwei Seiten eingebunden wird:

1. Schnäppchen sind entweder Kunstwerke mit einem Schätzwert unter 2000 Euro oder Süßigkeiten unter 3 Euro. Gib auf der Seite eine Liste von Schnäppchen (bezeichnung und gewicht) aus. Vermerke dabei auch in einer eigenen Spalte, ob es sich um ein Kunstwerk oder ein Produkt aus dem Standardsortiment handelt.
2. Auf einer anderen Seite soll der Benutzer die Moeglichkeit haben, eine der in der Datenbank angelegten Kunstschauen auszuwaehlen. Anschlieszend soll zur gewaehlten Kunstschau eine Liste mit allen ausgestellten Kunstwerken samt erreichten Plaetzen angezeigt werden.“ [ROS]

1.3 Hinweise

- „Verwende dazu die mysql-Version der schokoDB (ist im Kurs hochgeladen).
- Nutze Statements um sql injections zu vermeiden.
- Abzugeben sind ein Arbeitsprotokoll sowie das php Plugin

[1] https://codex.wordpress.org/Writing_a_Plugin“ [ROS]

2 Ergebnisse

2.1 Theorie

In PHP ist es üblich den schließenden Tag wegzulassen, da so automatisch der Schlusstag gesetzt wird und sichergestellt wird, dass keine Leerzeilen eingefügt werden. Außerdem ist es üblich HTML und PHP zu mischen, auch wenn dies eine unsaubere Art der Entwicklung ist.

Für die zurückzugebenden Ergebnisse werden Funktionen erstellt, welche mittels `shortcode` verwendet werden können. Dafür definiert man eine Funktion und später kann man mittels des Funktionsaufrufs `add_shortcode(kurzname, funktionsname);` den Shortcode setzen. In der WordPress Seite lässt sich nun mittels [kurzname] der Rückgabewert der Funktion darstellen.

2.2 Designüberlegung

Die WordPress Datenbank wird getrennt von der Schokofabrik Datenbank angelegt um eine klare Trennung der Anwendungsfälle zu wahren. Für beide Datenbanken gibt es unterschiedliche Benutzer die in der jeweils eigenen Datenbank alle Rechte haben. Der WordPress-Benutzer erhält jedoch auch SELECT Rechte für die benötigten Datenbankobjekte in der Schokofabrik.

Die erste Teilaufgabe benötigt keine Userinteraktion und deswegen wird aus Performance- und Sicherheitsgründen eine View erstellt, die nur die geforderten Inhalte enthält.

Für die zweite Teilaufgabe könnte für jede Kunstschau eine View erstellt werden, jedoch wäre dies ein sehr hoher Aufwand und außerdem ist diese Lösung zukünftig nicht einfach erweiterbar. Deshalb werden Daten aller benötigter Tabellen in Views gespeichert, damit der WordPress-User mit dem Prepared Statement nur noch auf diese Views zugreifen muss.

2.3 Allgemeines

Zuerst wurde WordPress heruntergeladen und ins `/var/www` Verzeichnis entpackt. Danach wurden die entsprechenden Dateirechte gesetzt. Nun wurde ein `ServerAlias /wp` gesetzt und ein `Allow from All` für die Seite gesetzt, damit man sie erreichen kann. Zusätzlich wurde in der Host Datei ein Eintrag für die Adresse mit dem Namen `wp.local` erstellt.

2.4 Datenbank

Nun wurde in mysql eine Datenbank mit entsprechendem User erstellt.

1. `CREATE DATABASE schokofabrik;`
2. `CREATE USER schokouser@localhost IDENTIFIED by ‘schokouser‘;`
3. `GRANT ALL PRIVILEGES ON schokofabrik.* TO schokouser@localhost;`
4. `FLUSH PRIVILEGES; -- neu laden der privileges`

Die Inhalte der Datenbank wurde mittels dem Befehl `source` und dem beigelegten SQL File eingefügt.

Diese Schritte wiederholen sich für den `schokowpuser` mit der `schokowpdb` Datenbank.

2.5 Schulübung

WordPress wurde so konfiguriert, dass es die `schokowpdb` Datenbank verwendet.

Nun wurde noch in der WordPress Konfiguration `/var/www/wp/wp-config.php` der Wert `WP_DEBUG` auf `true` gesetzt, damit Fehler bei der Plugin-Entwicklung einfacher erkannt werden.

Nun wurde in dem Ordner `/…/wp-content/plugins` eine PHP Datei erstellt und danach wurden die richtigen Dateiberechtigungen gesetzt.

Damit dieses File als Plugin erkannt und richtig beschrieben wird bedarf es eines Kommentars am Anfang des Files.

```
/*
Plugin Name: SchokoDB Plugin
Author: Daniel M.
Version: 1.0
Description: Ein Plugin fuer die SchokoDB
*/
```

Abbildung 1: Fileheader

Nun wird das Plugin in WordPress erkannt und es kann aktiviert werden.

Abbildung 2: aktiviertes Plugin

```php
<?php
    /*
        Plugin Name: SchokoDB Plugin
        Author: Daniel M.
        Version: 1.0
        Description: Ein Plugin fuer die SchokoDB
    */

    function schokoProdukte_func(){
        global $wpdb;
        ob_start();

        $res = $wpdb->get_results("select * from Produkt");

        $text = '';
        foreach($res as $r) {
            echo $r->bezeichnung;
        ?>
        <br>
        <hr>
        <?php
            echo '<br>';
        }

        return ob_get_clean();;
    }

add_shortcode('schokoProdukte','schokoProdukte_func');
```

Abbildung 3: Bisheriges Plugin

Das Plugin, das bis jetzt erstellt wurde, gibt nur alle Produkte der Schokofabrik aus. Dazu wird der Ausgabepuffer gestartet, damit die Ausgabe zuerst zwischengespeichert wird. Danach wird mittels der Datenbankvariable die `get_results` Funktion aufgerufen, welche einen Iterator auf eine Ergebnisliste einer Query zurückliefert. Nun wird mit einer foreach Schleife durchiteriert und jede Produktbezeichnung wird ausgegeben. Nach jeder Bezeichnung folgen ein Zeilenumbruch, ein horizontaler Trennstrich und noch ein Zeilenumbruch. Mit der Funktion `ob_get_clean` wird der aktuelle Ausgabepuffer zurückgegeben und danach gelöscht.

Die Seite wird jetzt in WordPress wie folgt erstellt:

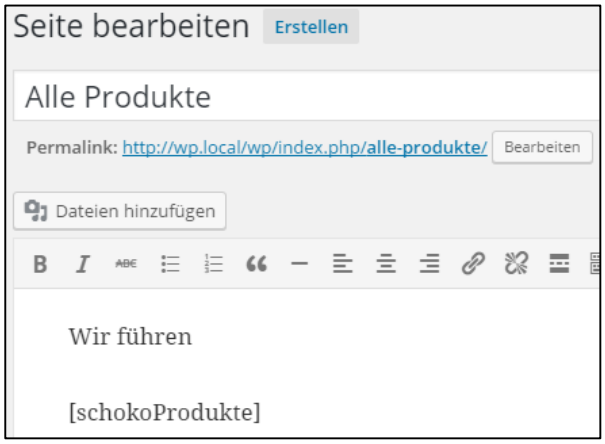

Abbildung 4: Seite für alle Produkte

Nun ist vorerst eine Seite vorhanden die alle Produkte der Schokofabrik ausgibt.

Abbildung 5: Alle Produkte

Nun wurde statt horizontalen Trennstrichen eine Tabelle verwendet, die mittels der Bootstrap Klassen schön formatiert wurde. Außerdem wurde noch das Gewicht der Produkte hinzugefügt.

```
function schokoProdukte_func()
{

    //wenn eine globale Variable verwendet wird muss ma
    global $wpdb;
    ob_start();

    $res = $wpdb->get_results("SELECT * FROM Produkt");

    echo '<table class="table-striped table-hover">';
    foreach ($res as $r) {
        ?>
        <tr>
            <td>
                <?php echo $r->bezeichnung; ?>
            </td>
            <td class="small">
                <?php echo $r->gewicht; ?>
            </td>
        </tr>
        <?php
    }
    echo '</table>';

    return ob_get_clean();
}
```

Abbildung 6: schokoProdukte nach der 2. Schuleinheit

Danach wurde eine Funktion geschrieben, die alle Kunden in einem Formular ausgibt. Wählt man einen Kunden aus werden entsprechende Details angezeigt.

```php
/**
 * Funktion um alle Kunden auszugeben.
 *
 * Gibt alle Kunden der Schokofabrik in einem Formular zurueck.
 * @return Bezeichnung aller Kunden in einem Formular.
 */
function schokoKunden_func()
{
    global $wpdb;
    ob_start();
    ?>
    Bitte ausw&auml;hlen:
    <form method="GET">
        <select name="kunde">
            <?php
            $kunden = $wpdb->get_results("SELECT * FROM Kunde");
            foreach ($kunden as $k) {
                echo '<option value="' . $k->firmenname . '">' . $k->firmenname . '</option>';
            }
            ?>
        </select>
        <input type="submit" value="Anzeigen" class="btn btn-default">
    </form>
    <?php
    if (isset($_REQUEST['kunde'])) {
        $kundendaten = $wpdb->get_results($wpdb->prepare('SELECT * FROM Kunde WHERE firmenname = %s', $_REQUEST['kunde']));
        $kunde = $kundendaten[0];

        echo '<div class="well">Name: ' . $kunde->firmenname . '<br>' . 'Adresse: ' . $kunde->adresse . '<br>' . 'Telefonnummer: ' . $kunde->telefonnummer . '</div>';
    }

    return ob_get_clean();
}
```

Abbildung 7: schokoKunden

Alle Firmennamen werden in einem Select Feld dargestellt. Das Formular funktioniert mit der GET action, da die Aufrufe einfach über die URL funktionieren. Deshalb müssen aber Prepared Statements verwendet werden, damit keine SQL Injections möglich sind. Schickt man das Formular also ab, ist bei erneutem Aufrufen der Seite die $_REQUEST[‘kunde‘] Variable gesetzt. Danach wird ein Statement vorbereitet (statt ? wie in Java werden hier %s für Strings verwendet). Das Statement wird anschließend wieder ausgeführt und da es nur eine Ergebnisreihe geben kann, wird der erste Eintrag des Arrays gespeichert. Danach werden die Firmendetails ausgegeben.

2.6 Hausübung

Zuhause wurde erstmals WordPress neu installiert, da es bei meiner Installation aufgrund eines Fehlers nicht möglich war nach Themes zu suchen. Danach habe ich zwei Datenbanken, wie in den Designüberlegungen erwähnt, aufgesetzt.

Teil 1

Die erste Aufgabenstellung war bereits in der SQL Query Aufgabe enthalten, deswegen habe ich den SELECT Befehl kopiert, eine View daraus erstellt und dem schokowpuser die SELECT Rechte dafür erteilt.

```
-- View der Schnaeppchen:
CREATE VIEW schokofabrik.schnaeppchen
AS SELECT bezeichnung, gewicht, 'Kunstwerk' AS ist
FROM Kunstwerk NATURAL JOIN Produkt
WHERE schaetzwert < 2000
UNION
SELECT bezeichnung, gewicht, 'Standardsortiment' AS ist
FROM Standardsortiment NATURAL JOIN Produkt
WHERE preis < 3;
-- SELECT Rechte fuer die View fuer den Wordpress User
GRANT SELECT ON schokofabrik.schnaeppchen TO schokowpuser@localhost;
```

Abbildung 8: View der Schnäppchen

Die Funktion, die alle Produkte ausgibt musste nur geringfügig verändert werden, damit sie die erste Aufgabenstellung erfüllt. Zuerst wurde eine Datenbankverbindung zu einer externen Datenbank aufgebaut. Dies funktioniert so:

```
$wpschokofabrik = new WPDB('schokowpuser', 'schokowpuser', 'schokofabrik', 'localhost');
```

Abbildung 9: neue Datenbankverbindung

Danach musste nur noch die Query angepasst werden und kleine Layout-Änderungen wurden vorgenommen.

```
function schokoSchnaeppchen_func()
{
    $wpschokofabrik = new WPDB('schokowpuser', 'schokowpuser', 'schokofabrik', 'localhost');
    ob_start();

    $res = $wpschokofabrik->get_results("SELECT * FROM schnaeppchen");

    echo '<table class="table-striped table-hover"><tr><th>Bezeichnung</th><th>Gewicht in g</th><th>ist</th></tr>';
    foreach ($res as $r) {
        ?>
        <tr>
            <td>
                <?php echo $r->bezeichnung; ?>
            </td>
            <td class="small">
                <?php echo $r->gewicht; ?>
            </td>
            <td class="small">
                <?php echo $r->ist; ?>
            </td>
        </tr>
        <?php
    }
    echo '</table>';

    return ob_get_clean();
}
```

Abbildung 10: finale Version der Schnäppchen

Das Ergebnis sieht letztendlich so aus:

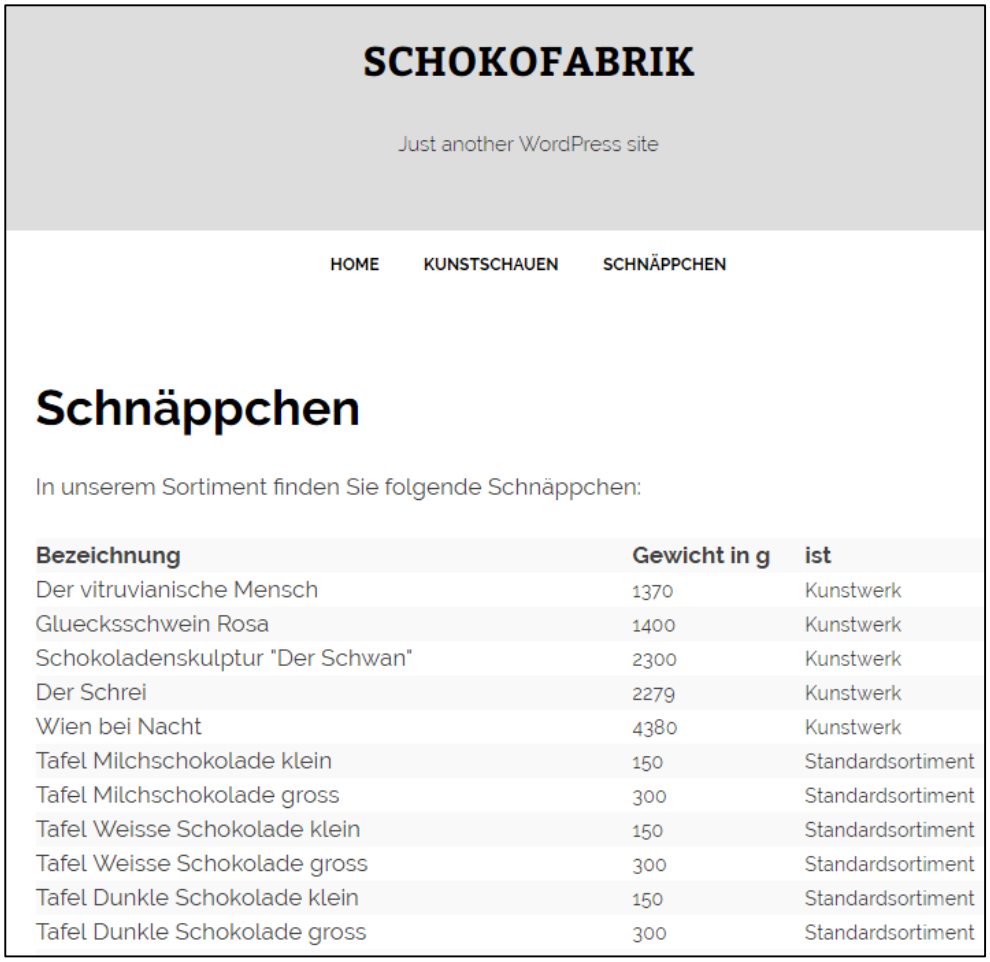

Abbildung 11: Schnäppchen Ansicht

Teil 2

Für das Auswählen der Kunstschauen benötigt man die Primärschlüssel der Kunstschau. Deswegen wurde mit diesen eine eigene View erstellt.

```
-- View aller Kunstschauen
CREATE VIEW schokofabrik.kunstview AS SELECT datum,name FROM Kunstschau;
GRANT SELECT ON schokofabrik.kunstview TO schokowpuser@localhost;
```

Abbildung 12: kunstview

Um die Platzierung herauszufinden, benötigt man aus der Tabelle `zeigt` die Werte für `datum`, `name`, `platz` und `kunstwerknummer`.

```
-- View aller Platzierungen auf Kunstschauen
CREATE VIEW schokofabrik.platzierung AS SELECT datum,name,platz,kunstwerknummer FROM zeigt;
GRANT SELECT ON schokofabrik.platzierung TO schokowpuser@localhost;
```

Abbildung 13: platzierung

Schlussendlich muss man auch eine Bezeichnung der Kunstwerke angeben, deswegen wird noch eine View `prodview` mit der `nummer` und der `bezeichnung` erstellt.

```
-- View aller Produkte
CREATE VIEW schokofabrik.prodview AS SELECT nummer,bezeichnung FROM Produkt;
GRANT SELECT ON schokofabrik.prodview TO schokowpuser@localhost;
```

Abbildung 14: prodview

Danach habe ich mir noch das Prepared Statement für die Aufgabenstellung überlegt.

```
-- Prepared Statement fuer die 2. Seite
SELECT bezeichnung, platz
FROM platzierung JOIN prodview ON platzierung.kunstwerknummer=prodview.nummer
WHERE name=? AND datum=?;
```

Abbildung 15: Prepared Statement

Die Aufgabenstellung ist zwar korrekt gelöst, jedoch kann bei einer Kunstschau das gleiche Kunstwerk zwei verschiedene Plätze erreichen, da es von verschiedenen Künstlern ausgestellt werden kann. Dies ist für den Endbenutzer eventuell verwirrend, wenn er diese Zusatzinformation nicht hat.

Wieder wurde die Funktion großteils von der Schulübung übernommen, jedoch ergab sich durch die Neuinstallation von WordPress ein Problem. Die Permalinks der Seiten waren anfangs auf Plain gestellt.

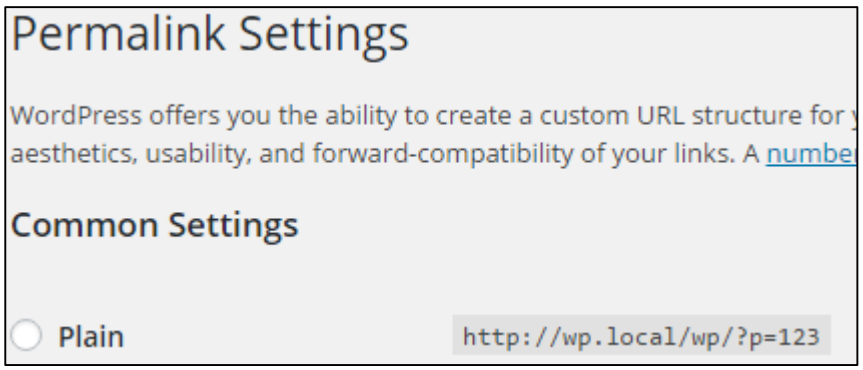

Abbildung 16: Permalink Einstellungen

Nach Absenden des Formulars wurden die GET Parameter jedoch nicht an die richtige Seite sondern an die Home-Seite `wp.local/wp` übergeben. Da auf dieser Seite das Plugin keine Funktion übernimmt, hat es anfangs auch nicht funktioniert. Deswegen wurden zuerst die Permalinks auf Postname umgestellt.

Abbildung 17: Post name setting

Hier habe ich anfangs Fehlermeldungen bekommen, da die gesuchten Seiten nicht gefunden wurden. Um dies zu beheben, musste ich in Apache das `rewrite` Modul aktivieren und `AllowOverride FileInfo` für diese Wordpress Installation erlauben. [FRU][APA]

Da der Primary Key der Kunstschauen aus zwei Teilen besteht, wurde dieses Wertepaar jeweils als einzelner Wert, der durch ein Komma getrennt ist verwendet. Mit `explode(delimiter, string)` lässt sich das Wertepaar anschließend wieder trennen.

```
$wpschokofabrik = new WPDB('schokowpuser', 'schokowpuser', 'schokofabrik', 'localhost');
ob_start();
?>
Bitte ausw&auml;hlen:
<form method="GET">
    <select name="kunstschau">
        <?php
        $kunstschauen = $wpschokofabrik->get_results("SELECT * FROM kunstview");
        foreach ($kunstschauen as $k) {
            echo '<option value="' . $k->datum . ',' . $k->name . '">' . $k->datum . ': ' . $k->name . '</option>';
        }
        ?>
    </select>
    <input type="submit" value="Anzeigen" class="btn btn-default">
</form>
```

Abbildung 18: Formular zum Auswählen

Nun wird noch das Prepared Statement ausgeführt und die Ergebnisse werden in einer Tabelle zurückgegeben.

```
<?php
if (isset($_REQUEST['kunstschau'])) {
    //aufteilen der durch , getrennten Primary Keys
    $requested = explode(',', $_REQUEST['kunstschau']);
    $kunstschaudetails = $wpschokofabrik->get_results(
    $wpschokofabrik->prepare('SELECT bezeichnung, platz FROM platzierung JOIN prodview ON platzierung.kunstwerknummer=prodview.nummer WHERE name=%s AND datum=%s',
    $requested[1], $requested[0]));
    echo '<h4 class="well">Platzierungen f&uuml;r ' . $requested[1] . ': ' . $requested[0] . '</h4>';
    echo '<table class="table-striped table-hover"><tr><th>Bezeichnung</th><th>Platz</th></tr>';
    foreach ($kunstschaudetails as $r) {
        ?>
        <tr>
            <td>
                <?php echo $r->bezeichnung; ?>
            </td>
            <td class="small">
                <?php echo $r->platz; ?>
            </td>
        </tr>
        <?php
    }
    echo '</table>';
}
return ob_get_clean();
```

Abbildung 19: Ausgabe des Ergebnisses

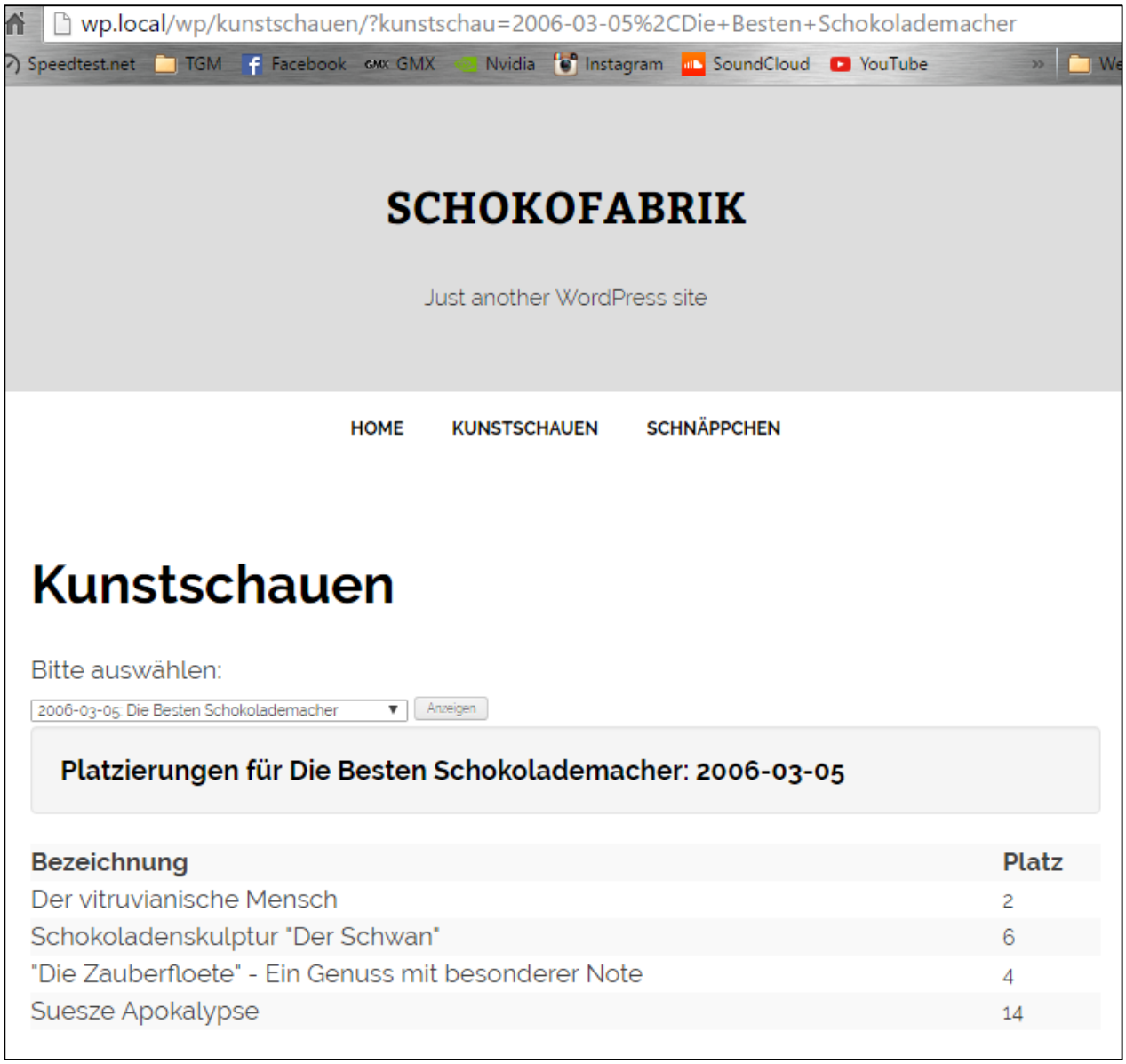

Abbildung 20: visuelles Ergebnis

2.7 GitHub Link

https://github.com/dmay-tgm/Wordpress-Plugin

3 Zeitmanagement

Der Aufwand wird wie folgt geschätzt:

Arbeitsteil	**geschätzter Aufwand**
Übung	2.5 h
Protokoll	1.5 h
Gesamt	**4 h**

Der tatsächliche Aufwand ist wie folgt aufgeschlüsselt:

Durchgeführte Arbeit	**Datum**	**tatsächlicher Aufwand**
Übung	10. - 23.03.2016	3 h
Protokoll	13. - 23.03.2016	1.5 h
Gesamt	**10. – 23.03.2016**	**4.5 h**

4 Literaturverzeichnis

[ROS] Christoph Roschger.
DATENBANKZUGRIFFE AUS WORDPRESS HERAUS [Online].
Available at:
https://elearning.tgm.ac.at/mod/assign/view.php?id=47516
[abgerufen am 23.03.2016]

[FRU] frustschieber (17. Mai 2015 16:51).
mod rewrite [Online]. Available at:
https://wiki.ubuntuusers.de/Apache/mod_rewrite/
[abgerufen am 23.03.2016]

[APA] The Apache Software Foundation (Version 2.4).
Apache Core Features: AllowOverride Directive [online].
Available at:
https://httpd.apache.org/docs/2.4/en/mod/core.html#allowoverride
[abgerufen am 23.03.2016]