

Leistungsnachweis "Entwicklung Webbasierter Anwendungen"
Sommersemester 2019
19.07.2019

Familienname	Vorname	Matrikelnummer	Note

Aufgabe	1	2	3a	3b	3c	3d	4	Gesamt
Punkte	12	13	12	4	11	6	14	72
Erreichte Punkte								

Bemerkungen:

- Bearbeitungszeit: 100 Minuten.
- Persönliche Notizen auf Papier sind als Hilfsmittel erlaubt, jedoch keine elektronischen Geräte.
- Sie dürfen die Klammerung der Aufgabenblätter lösen. Es liegt dann aber in Ihrer Verantwortung sicherzustellen, dass alle Blätter der Klausur abgegeben werden.
- Schreiben Sie klar und deutlich; Unleserliches kann nicht bewertet werden.
- Sollten Sie eine Frage nicht klar beantworten können, treffen Sie Annahmen und fügen Sie diese Ihrer Antwort hinzu.
- Sollten Sie mehr Platz für die Beantwortung einer Frage benötigen, verwenden Sie die enthaltenen leeren Seiten oder wenden Sie sich an die Aufsichtsperson.
- Den Platz nicht bearbeiteter Fragen können Sie für die Beantwortung anderer Fragen verwenden (bitte entsprechende Verweise zur Aufgabe schreiben).
- Sie können sowohl in Deutsch als auch in Englisch antworten (You may also answer in English).

Aufgabe "FormularGenerator"

Schreiben Sie eine Webseite zur bequemen Erzeugung von nutzerfreundlichen Formularen!

In der Datenbanktabelle "formelemente" (s.u.) wird für ein Formular festgelegt, welche Arten von Formularelementen (Typ) es enthalten soll und welche Attribute für die Anzeige (Beschriftung) und Übertragung (NameAttr) verwendet werden sollen.

Für die folgenden Daten soll das Formular aus dem nebenstehenden Screenshot erzeugt werden (Die Werte in den Feldern dienen nur zur Veranschaulichung!).

formelemente			
ID	Beschriftung	NameAttr	Typ
1	Name, Vorname	fullname	text
3	<Bemerkung>	bemerkung	text
2	Straße	strasse	text

Die ID bestimmt die Reihenfolge der Formularelemente!

Später (Aufgabe 3) soll das Formular noch gegen versehentliches Neuladen geschützt werden. Dazu werden die eingegebenen Daten im Hintergrund mit AJAX übermittelt und so gespeichert, dass sie bei einem erneuten Abruf der Seite wiederhergestellt werden können.

Hinweis: Die Auswertung der abgeschickten Daten und das Styling der Seite gehören NICHT zur Aufgabe! Sie müssen auch nur Formularelemente vom Typ "text" berücksichtigen. Die Daten für das umgebende Formular (z.B. die Zieladresse des Formulars) stammen aus einer anderen Tabelle, die nicht Bestandteil der Aufgabe ist.

Lesen Sie zuerst die gesamte Aufgabenstellung. Schreiben Sie dann die fehlenden Methoden der bereits teilweise vorgegebenen Seitenklasse **FormularGenerator** (siehe unten). Die bekannte Klasse **Page** und die Datenbank sind ebenfalls vorgegeben. Der Code muss Exceptions einsetzen, vollständig, standardkonform und sicher sein.

```

abstract class Page
{
    protected $db;

    protected function __construct() {
        $this->db = new MySQLi(
            "localhost", "usr", "pwd", "formbuilder");
        if (mysqli_connect_errno())
            throw new Exception(mysqli_connect_error());
        if (!$this->db->set_charset("utf8"))
            throw new Exception($this->db->error);
    }

    protected function __destruct() {
        $this->db->close();
    }

    protected function generatePageHeader($headline = "") {
        header("Content-type: text/html; charset=UTF-8");
        $headline = htmlspecialchars($headline);
        echo <<<EOT
        <!DOCTYPE html>
        <html lang="de" >
        <head>
        <meta charset="UTF-8" />
        <title>$headline</title>
        <script src="FormularGenerator.js"></script>
        </head>
        <body onload="start();">
        EOT;
    }
}

```

```

protected function generatePageFooter() {
    echo "</body>\n";
    echo "</html>\n";
}

protected function processReceivedData() { }

//class FormularGenerator
require_once 'Page.php';
class FormularGenerator extends Page {
    public static function main()
    {
        try {
            $page = new FormularGenerator();
            $page->processReceivedData();
            $page->generateView();
        }
        catch (Exception $e) {
            header("Content-type:text/plain;
                charset=UTF-8");
            echo $e->getMessage();
        }
    }
}
FormularGenerator::main();

```

Aufgabe 1: Daten auslesen und für die Verarbeitung bereitstellen (12 Punkte)

Lesen Sie die Daten gemäß der Seitenklassenarchitektur aus der Datenbank aus und wandeln Sie die Datensätze in eine geeignete Datenstruktur für die anschließende Weiterverarbeitung um. Geben Sie NICHT das Datenbankobjekt nach außen, sondern verbergen Sie die Datenbankzugriffe innerhalb der Methode.

```
protected function _getViewData__{  
    // Ergänzen Sie hier Ihren Code  
    // ID beinhaltet implizit die Anzeigereihenfolge  
    $sql = "SELECT ID, Beschriftung, NameAttr, Typ FROM formelemente ORDER BY ID";  
  
    $recordset = $this->db->query ($sql);  
    if (!$recordset)  
        throw new Exception("Abfrage fehlgeschlagen: ".$this->db->error);  
  
    $elemente = array();  
    $data = array();  
    while ($record = $recordset->fetch_assoc()) {  
        $data["ID"] = $record["ID"];  
        $data["Beschriftung"] = $record["Beschriftung"];  
        $data["NameAttr"] = $record["NameAttr"];  
        $data["Typ"] = $record["Typ"];  
        $elemente[$record["ID"]] = $data; // Array of arrays  
    }  
    $recordset->free();  
    return $elemente;  
}
```

Aufgabe 2: HTML erzeugen mit PHP und MySQL (13 Punkte)

Generieren Sie das oben abgebildete Formular als vollständige HTML-Seite gemäß der Seitenklassenarchitektur.

- Das ausgefüllte Formular soll via POST mit den Daten an "<https://echo.fbi.h-da.de>" abgeschickt werden.
- Verwenden Sie die Standardelemente zum Beschriften von Formularelementen.
- Erzeugen Sie die im Screenshot dargestellten Werte der Eingabefelder vorerst als leere Einträge. (Die echten Werte werden später im AJAX-Teil der Aufgabe gesetzt.)
- Sie dürfen davon ausgehen, dass die Werte für "NameAttr" und "Typ" in der Datenbank keine ungültigen Werte oder Sonderzeichen enthalten. Aber im Feld "Beschriftung" sollten Sie mit Sonderzeichen rechnen.
- **Hinweis:** Sie müssen die Ausgabe nur für Formularelemente vom Typ "text" erzeugen.
- **Tipp:** Lagern Sie das Erstellen eines einzelnen Formularelements in eine separate Funktion aus. Dann können Sie die Hauptfunktion hier beschreiben und die ausgelagerte Funktion auf der nächsten Seite.

```
protected function generateView () {  
    if (!isset($_GET["key"])){ //No response-page required for AJAX-request (not expected)  
        $viewData = $this->getViewData();  
        $this->generatePageHeader('Formular');  
        // Ergänzen Sie hier Ihren Code  
        echo <<<HTML  
            <form action="https://echo.fbi.h-da.de" method="post" accept-charset="UTF-8">  
  
        HTML;  
        foreach($viewData as $elemente => $data) {  
            $this->generateInputElement($data);  
        }  
        echo <<<HTML  
            <input class = "submit" type="submit" value="Absenden"/>  
            </form>  
        HTML;  
        $this->generatePageFooter();  
    }  
}
```

// Ergänzen Sie hier weiteren Code zu Aufgabe 2 um die HTML-Seite zu erzeugen

```
private function generateInputElement($data) {
    $type=$data['Typ'];
    $name=$data['NameAttr'];
    // do not use htmlspecialchars before view is being generated!
    $label=htmlspecialchars($data["Beschriftung"]);
    $value="";
//***** Aufgabe 3d)
    if (isset($_SESSION[$name])) { // AJAX-Werte einbauen
        $value=$_SESSION[$name];
    }
//*****

echo <<<HTML
    <div class="row">
        <label> $label <!-- label needs to be defined (accessibility) -->
        <input type="$type" name="$name" value="$value"/>
    </label>

    </div>

HTML;
}
```

Aufgabe 3: ECMAScript / AJAX / PHP (33 Punkte)

Beim Ausfüllen eines Eingabefeldes sollen im Hintergrund die Daten dieses Formularelements (als Zwischenstand) an den Server übertragen werden, damit sie bei einem erneuten Seitenabruf (z.B. Reload) wiederhergestellt werden können.

- a) Schreiben Sie eine JavaScript-Funktion **sendData()**. Die Funktion liest das name-Attribut und den Wert des bearbeiteten Formularelements aus und sendet diese mit AJAX an den Server. Diese Funktion soll später (in Teil b) automatisch aufgerufen werden, wenn ein Eingabefeld bearbeitet wird.
- Erzeugen Sie aus den beiden ausgelesenen Werten X, Y einen Parameter-String zur Übertragung. Der Parameterstring sollte folgendes Format haben: **key=X&value=Y**
 - Senden Sie den Request mit GET an **FormularGenerator.php** und hängen Sie den Parameter-String durch ein '?' getrennt an.
 - Weil keine Daten empfangen, sondern nur gesendet werden, setzen Sie für den AJAX-Callback eine leere Funktion.
 - **Tipp:** Ihre JavaScript-Funktion kann mit event.target das DOM-Element abfragen, an dem das letzte Event (onxxx) aufgetreten ist.

```
//FormularGenerator.js
function sendData() {
    "use strict";
    let elem = event.target;          // DOM-Knoten, an dem das Event ausgelöst wurde

    let key = elem.name;
        console.log("Sendig AJAX Request... " + key);
    let value = elem.value;
    let param = "key=" + key + "&value=" + encodeURIComponent(value);      ; //encodeURIComponent not expected !
    let xhr = new XMLHttpRequest();
    xhr.open("GET", "FormularGenerator.php?" + param);
    xhr.onreadystatechange = function () {
        if (this.readyState == 4 && this.status == 200) { // not required - empty function ok!
            console.log("AJAX Request finished; Data stored in Session");
        }
    }
        console.log("sending AJAX Request: " + param);
    xhr.send();
}

//***** 3b) *****

window.onload = init; // not needed if function is called "start" (see page class: body onload...)
function init() {
    // eventlistener registrieren via JS
    // alternativ in HTML möglich <input ... onchange="sendData()" .../>;
    "use strict";
    let inputs = document.getElementsByTagName("input");
    console.log("Seite vollständig geladen... " + inputs.length);
    for (let i = 0; i < inputs.length; i++) {
        inputs[i].addEventListener("change", sendData, false);
    }
}
```

b) Implementieren Sie den Aufruf von **sendData()**!

Immer wenn ein Eingabefeld bearbeitet wird, soll die Funktion `sendData()` aufgerufen werden.

Passen Sie den in den vorherigen Aufgaben erstellten JavaScript- oder PHP-Code entsprechend an. Markieren Sie die Einfügung gut erkennbar mit **"3b"**.

c) Verarbeiten Sie die eingehenden AJAX-Daten mit PHP!

- Speichern Sie die eingehenden Werte so in der Session ab, dass Sie beim Generieren der HTML-Seite (in Aufgabe d) darauf zugreifen können. Verwenden Sie dafür ausschließlich eine Session! Schreiben Sie KEINE Daten in die Datenbank!
- Ein Hacker könnte Ihren Webaufttritt angreifen, wenn es ihm gelingt, bestimmte Sessionvariablen zu setzen (z.B. `DEBUG=true`). Vergleichen Sie das empfangene Name-Attribut mit den Datenbankeinträgen und setzen Sie die Sessionvariablen nur, wenn das übermittelte Eingabefeld auch in der Datenbank existiert.

// Ergänzen Sie hier Ihren Code

```
protected function __processReceivedData () {

    session_start(); // first thing - maybe in main, but better here because main is given
    parent::processReceivedData();
    if (isset($_GET["key"]) && isset($_GET["value"]) ) {
        $key=$_GET["key"];
        $value= $_GET["value"];
        //error_log ("$key = $value");    // for debugging
        if ($this->isValidKey($key)) {
            $_SESSION[$key] = $value;
        }
    }
}

protected function isValidKey($name){
    //Helper function - checks whether passed name-attribute exists in database
    $key=$this->db->real_escape_string($name);
    $sql = "SELECT NameAttr FROM formelemente WHERE NameAttr = '$key'";
    //error_log ("***** $sql *****");
    $recordset = $this->db->query ($sql);
    if (!$recordset)
        throw new Exception("Abfrage fehlgeschlagen: ".$this->db->error);

    $ret = false;
    if ($recordset->num_rows) {    $ret = true; }
    $recordset->free();
    return $ret;
}
```

- d) Verwenden Sie die mit AJAX übermittelten Werte beim Generieren der HTML-Seite!
Passen Sie die Generierung der HTML-Seite aus Aufgabe 2 so an, dass die Seite mit diesen Werten generiert wird. Schreiben Sie den entsprechend angepassten Code in das folgende Feld und markieren Sie in Aufgabe 2 die Stelle, die ersetzt werden soll, gut erkennbar mit "3d".

// Erzeugung der HTML-Seite mit AJAX-Werten

[siehe Aufgabe 2](#)

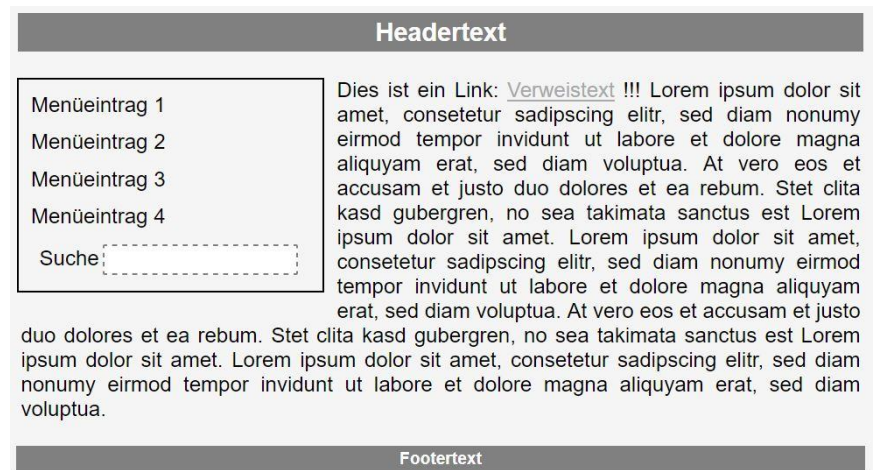
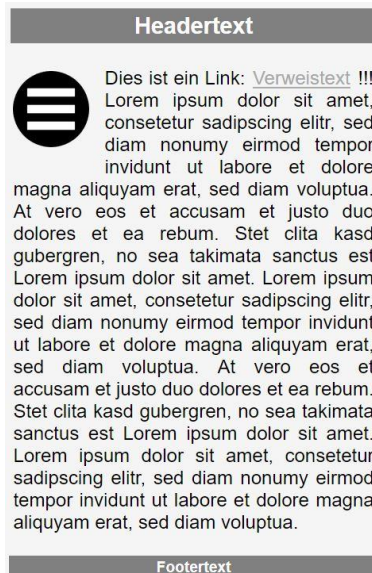
Platz für Notizen, Korrekturen oder weitere Lösungen... es kommt aber noch eine Aufgabe!

Aufgabe 4: CSS (14 Punkte)

Gegeben ist folgender HTML-Code (unabhängig von der Formular-Aufgabe). **Diesen Code dürfen Sie nicht ändern!**

```
<header><h1 id="header">Headertext</h1></header>
<nav>
  <ul class="nav_normal">
    <li>Menüeintrag 1</li>
    <li>Menüeintrag 2</li>
    <li>Menüeintrag 3</li>
    <li>Menüeintrag 4</li>
    <li><label>Suche<input type="text" value=""/></label></li>
  </ul>
  
</nav>
<section>
  <article>Dies ist ein Link:<a href="CSS_Aufgabe.html">Verweistext</a>!!! Lorem ipsum dolor....
  </article >
</section>
<footer ><h1 id="footer">Footertext</h1></footer>
```

Nun sollen Sie mit CSS dafür sorgen, dass das Layout und das Design für schmale bzw. breite Displays wie folgt aussehen:



Geben Sie den CSS-Code an, der die folgenden Aspekte des Designs umsetzt. Designteile nach denen nicht gefragt wird, müssen Sie nicht umsetzen!

a) Alle Texte sollen in einer schnörkel-freien Schrift (ohne Serifen), vorzugsweise 'Helvetica' dargestellt werden.	<pre>* { font-family: Helvetica, sans-serif}</pre>
b) Die Schriftgröße des Headertexts beträgt 120% der defaultmäßig festgelegten Browser-Basisschriftgröße; die Footertext-Schriftgröße 80%.	<pre>h1 { font-size: 1.2rem} #footer { font-size: 0.8rem}</pre>
c) Zeigen Sie alle bereits besuchten Verknüpfungen ("Verweistext") in dunkelgrau an.	<pre>a:visited{ color: DarkGray; }</pre>
d) Die Navigationsleiste (bzw. die "Hamburger-Grafik") soll an der linken Seite angezeigt und vom Text umflossen werden. Der Footer soll wieder die gesamte Bildschirmbreite ausnutzen (und von keinem anderen Element überdeckt werden).	<pre>/* Layout */ nav {float:left;} footer {clear: both;} /** Zuerst Handys als Default **/ .nav_normal {display: none } .nav_icon {display: block;}</pre>
e) Bei einer Bildschirmbreite von unter 600 Pixeln soll eine "Hamburgergrafik" statt dem Navigationselement angezeigt werden. Ab 600 Pixeln wird dann das eigentliche Navigationsmenü angezeigt.	<pre>@media only screen and (min-width: 600px){ .nav_normal { display: inline-block;} .nav_icon {display: none;} }</pre>
f) Das Suchfeld unten in der Navigationsleiste soll von einer grauen gestrichelten Linie umrahmt sein. Selektieren Sie das Element so, dass die Specificity möglichst hoch ist.	<pre>nav ul li input[type="text"] {border: 0.1em dashed grey !important}</pre>

Platz für Notizen oder weitere Lösungen