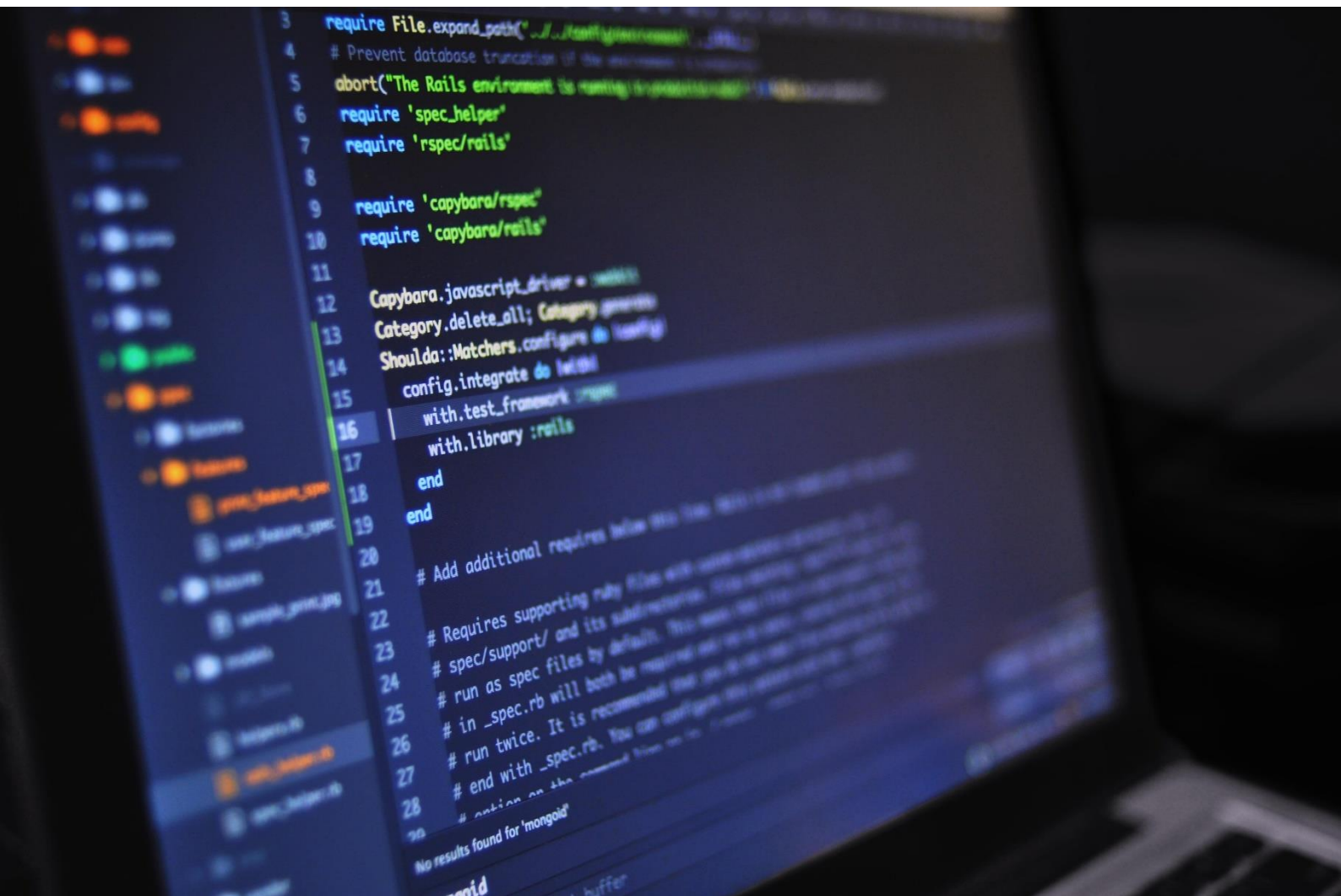


Dokumentation Webprogrammieren



Mitgliederverwaltung

Webprogrammieren Studienarbeit | 4. Semester

Dozent: Ulrich Hauser

Hochschule für Technik und Wirtschaft

Bachelorstudiengang: Digital Business Management, Teilzeit

Chur, 6. Juni 2019

Inhaltsverzeichnis

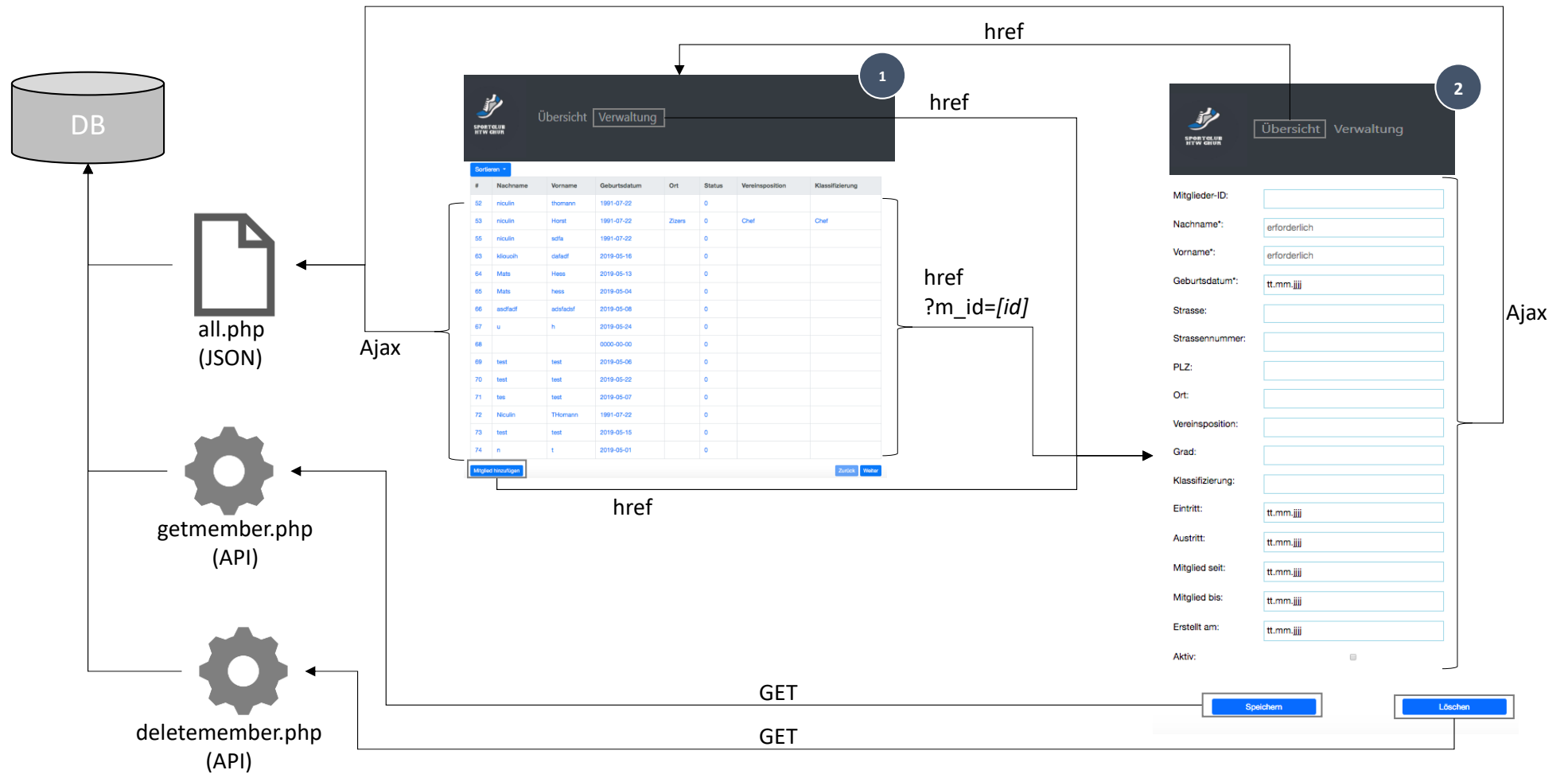
Struktur und Aufbau Datenbank.....	3
HTML-Konstrukt.....	5
View 1	
View 1 Auslesen der Daten - read_data().....	6
View 1 Tabellenerstellung - table_creation()	6
View 1 Sortieren Auswahl - sort	8
View 1 Weiter Button - Function forward_button	9
View 1 Zurück Button - Function backward_button	9
View 2	
View 2 Overview und Struktur.....	10
View 2 Validierung - xyzValidation()	11
View 2 Erstellen - createMember()	12
View 2 Ändern / Update - getSingleMember().....	13
View 2 Löschen - deleteMember()	14
Benutzerhandbuch	
Benutzerhandbuch 1. Übersichtsseite.....	15
Benutzerhandbuch 2. Verwaltungsseite	15

Struktur und Aufbau | Datenbank

Das Projekt wurde in zwei Views und in zwei entsprechende Gruppen unterteilt. Gruppe 1 mit View 1, welche die Übersicht der Mitglieder erstellt mit der gesamten Darstellung. Und Gruppe 2 mit View 2, welche die Verwaltungsseite erstellt, zur Erfassung neuer Mitglieder und Änderung oder Löschung bestehender Mitglieder. Die Datenbank wurde mit php eingerichtet, initial war dafür folgendes ERM v1 angedacht:



Dieses wurde auf Grund der Komplexität wieder verworfen und durch eine flache Datenbank v2 ersetzt.

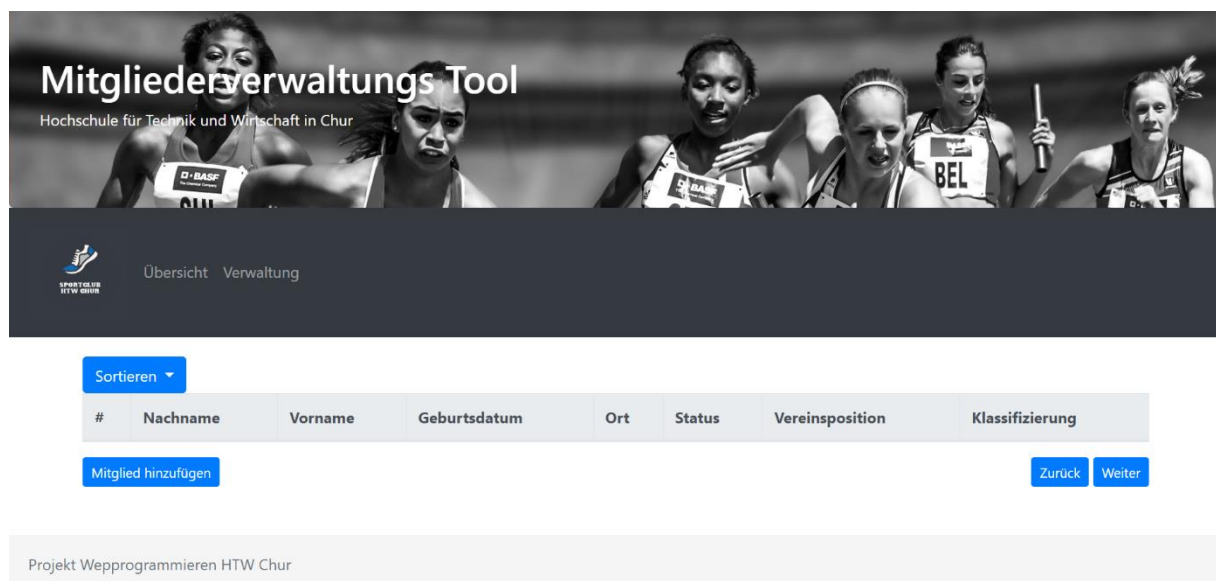


HTML-Konstrukt

Um ein responsives Design für die Seite zu erhalten wurde für den Aufbau des HTML-Konstrukts Bootstrap verwendet. Der Aufbau wurde folgendermassen organisiert:

- Jumbotron mit einem Hintergrundbild → In Datei *index.html* Zeile 32 bis 37
- Navbar mit Logo und den zwei Verlinkungen «Übersicht» (Indexseite) und «Verwaltung» (Mitglieder Erfassungsseite) → In Datei *index.html* Zeile 39 bis 58
- Body
- Footer → In Datei *index.html* Zeile 118 bis 122

View 1 Index



Die Indexseite beinhaltet die Mitgliedertabelle mit dazugehörigem «Sortieren» - Button, der Möglichkeit mit dem «Weiter» und «Zurück» - Button in den Tabelleninhalten zu blättern und mittels des Buttons «Mitglied hinzufügen» ein neues Mitglied zu erfassen.

Die Indexseite beinhaltet lediglich den fixen Tabellenheader mit den Spalten: #, Nachname, Vorname, Geburtsdatum, Ort, Status, Vereinspositionierung, Klassifizierung. Die Inhalte der Tabelle werden mittels der Java Script Funktion `table_creation` generiert. Diese Funktion ist im `<head>` auf Zeile 15 referenziert.

Oberhalb der Tabelle ist ein «sortieren» - Button enthalten. Dieser Button und das Drop-down sind fix auf der Indexseite von Zeile 62 bis 72 vermerkt. Die Sortierfunktion wird mittels der Java Script Funktion `sort`, welche im `<head>` auf Zeile 19 verlinkt ist ausgeführt.

Die beiden Buttons «Weiter» und «Zurück» sind fix auf der Indexseite von Zeile 108 bis 115 integriert. Die Funktionalität der Buttons werden mittels den Java Script Funktionen `backward_button` und `forward_button` ausgeführt, welche im `<head>` in Zeile 18 und 16 verlinkt sind.

Der Button «Mitglied hinzufügen» verlinkt auf die Seite *members.html*, auf dieser Seite ist es möglich, ein neues Mitglied hinzuzufügen.

View 1 | Auslesen - read_data()

Die Funktion dient dazu die Datensätze aus der Datenbank auszulesen.

Initialisierung:

Beim Laden der HTML Seite wird die Funktion initialisiert.

Input

Die PHP-Datei *All.php* liefert die Datensätze aus der Datenbank im JSON-Format.

Verarbeitung

Mittels `JSON.parse()` werden die JSON-Datensätze zerlegt und ins Objekt `memberObj` gespeichert.

Output

Die definierte Variabel `memberObj`, welche JSON Objekte enthält.

Zudem wird die Funktion `function_forward()` mit dem Parameter «initial» ausgeführt, welche durch den Parameter bekannt gibt, dass es sich um eine initiale Ausführung handelt.

Abhängigkeiten:

JSON File der Datenbank

Wird über das HTML aufgerufen. Im *index.html* wird mittels `EventListener()` die Funktion `read_data` initial aufgerufen.

View 1 | Tabellenerstellung - table_creation()

Die Funktion erzeugt eine dynamische Tabelle und füllt diese mit den acht definierten Datensätze der Mitglieder. Der Tabellenkopf bleibt dabei statisch.

Frontend

#	Nachname	Vorname	Geburtsdatum	Ort	Status	Vereinsposition	Klassifizierung
52	niculin	thomann	1991-07-22		0		
53	niculin	Horst	1991-07-22	Zizers	0	Chef	Chef
55	niculin	sdfa	1991-07-22		0		
63	kliouoih	dafadf	2019-05-16		0		
64	Mats	Hess	2019-05-13		0		
65	Mats	hess	2019-05-04		0		
66	asdfadf	adsfadsf	2019-05-08		0		
67	u	h	2019-05-24		0		
69	test	test	2019-05-06		0		
70	test	test	2019-05-22		0		
71	tes	test	2019-05-07		0		
72	Niculin	THomann	1991-07-22		0		
73	test	test	2019-05-15		0		
74	n	t	2019-05-01		0		
75	test	test	2019-05-15		0		

Initialisierung:

Beim Aufruf der HTML-Seite wird die dynamische Tabelle bis maximal 15 Zeilen und 8 Spalten erzeugt.

Input

Der Tabelleninhalt wird mit den Datensätze memberObj befüllt. Die Spalten werden über die Variabel member_attribute identifiziert.

Verarbeitung

Pro Datensatz wird eine neue Zeile mit dem member_attribute erzeugt.

Output

Es werden maximal 15 Mitglieder beziehungsweise Einträge angezeigt. Die Datensätze werden in der bestimmten Reihenfolge folglich dem Tabellenkopf aufgeführt.

Abhängigkeiten:

Der dynamische Teil der Tabelle wird dem Element tbody angefügt, welcher im *index.html* auf Zeile 93 aufzufinden ist. Die Daten werden aus der Variabel memberObj ausgelesen.

View 1 | Sortieren Auswahl - sort

Die Funktion dient dazu, die Einträge der Tabelle nach Mitglied ID, Nachname und Vorname zu sortieren.

Sortieren ▾		Vorname	Geburtsdatum	Ort	Status	Vereinsposition	Klassifizierung
Mitglied ID							
Nachname		thomann	1991-07-22		0		
Vorname							
53	niculin	Horst	1991-07-22	Zizers	0	Chef	Chef
55	niculin	sdfa	1991-07-22		0		
63	kliuoih	dafadf	2019-05-16		0		
64	Mats	Hess	2019-05-13		0		
65	Mats	hess	2019-05-04		0		
66	asdfadf	adsfadsf	2019-05-08		0		

Initialisierung:

Die Funktion wird bei einem «onclick» auf den jeweiligen Eintrag im Drop-down initialisiert. Dabei wird je nach Auswahl des Eintrages der jeweilige Parameter mitgegeben. → Siehe *index.html* Zeile 68 bis 70

Input

Die Daten, welche sortiert werden, werden aus dem Objekt memberObj gezogen.

Verarbeitung

Mittels der Funktion `compare` welche im Dokument *sort.js* in Zeile 11 ersichtlich ist, werden die einzelnen Einträge aus dem Objekt memberObj miteinander verglichen und je nach Wert wird der Funktion der Wert -1, 1 oder 0 zurückgegeben. Bei 1 wird der Eintrag oberhalb eingeordnet, bei -1 unterhalb und bei 0 auf gleicher Höhe.

Output

Nach Ausführung der Funktion wird die sortierte Tabelle angezeigt. Diese ist je nach Auswahl im Drop-down Menü nach BenutzerID, Vorname oder Nachname sortiert.

Abhängigkeiten:

Das Objekt memberObj liefert die Daten. Zudem besteht eine Abhängigkeit zum Button und Drop-down, welche im Dokument *index.html* von Zeile 62 bis 72 vermerkt sind.

View 1 | Weiter Button - Function forward_button

Die Funktion dient dazu weitere 15 Datensätze zu laden und anzuzeigen.

Initialisierung:

Beim Laden der Seite *index.html* wird ein «onclick» auf den forward_button gesetzt.

Input

Bei Klick auf den Weiter-Knopf wird die Funktion ausgeführt.

Verarbeitung

Die Idee des Weiter-Knopfs ist es, dass dieser aktiv ist, wenn es mehr als 15 Einträge zu zeigen gibt. Im Hintergrund liegt dazu eine mathematische Funktion, welche die Einträge auf der Datenbank liegend zählt, berechnet auf welcher Seite man sich befindet und auf wie viele Seiten die Einträge verteilt werden müssen. Wenn mehr als 15 Einträge vorhanden sind, ist der Knopf aktiv und beim Klick wird die Funktion aktiviert und für die nächsten 15 Einträge wird der Start- und Endwert berechnet.

Output

Mit den berechneten Start- und Endwerten wird die Funktion *table_creation* ausgeführt.

Abhängigkeiten:

Datensätze des Objekts memberObj werden ausgelesen.

Wird über das HTML aufgerufen. Im *index.html* wird mittels der id «forward» die Funktion *forward_button* initial aufgerufen.

View 1 | Zurück Button - Function backward_button

Die Funktion dient dazu die 15 vorherigen Datensätze zu laden und anzuzeigen.

Initialisierung:

Beim Laden der Seite *index.html* wird ein «onclick» auf den backward_button gesetzt.

Input

Beim Klick auf den Zurück-Knopf wird die Funktion ausgeführt.

Verarbeitung

Die Idee des Zurück-Knopfs ist es, dass dieser auf der ersten Ansicht inaktiv ist dh. wenn die ersten 15 Einträge angezeigt werden. Wenn man den Weiter-Knopf betätigt hat um weitere Einträge anzusehen wird der Zurück-Knopf aktiv. Im Hintergrund liegt dazu eine mathematische Funktion, welche die Einträge auf der Datenbank liegend zählt und berechnet auf welcher Seite man sich befindet, um entweder den Zurück-Knopf zu deaktivieren oder beim Klick wird die Funktion aktiviert und für die 15 vorherigen Einträge wird der Start- und Endwert berechnet.

Output

Mit den berechneten Start- und Endwerten wird die Funktion `table_creation` ausgeführt.

Abhängigkeiten:

Datensätze des Objekts `memberObj` werden ausgelesen.

Wird über das HTML aufgerufen. Im `index.html` wird mittels der id «backward» die Funktion `backward_button` initial aufgerufen.

View 2 | Overview und Struktur

In der View 2 wurde die Erstellung, Bearbeitung und Löschung von Mitgliedern erstellt. Dabei wurde zu einem späteren Zeitpunkt entschieden, den «Erstellen» und «Bearbeiten» Button durch ein und denselben Knopf «Speichern» abzubilden. Mit dem Button «Löschen» kann ein einzelner Datensatz aus der Datenbank unwiderruflich entfernt werden. Das Feld der Mitglieder-ID wird automatisch ausgefüllt bei bestehenden Mitgliedern, oder bleibt leer und in jedem Fall unveränderbar bei neuen Mitgliedern. Eine weitere Funktion besteht für die Validierung der Benutzereingaben.

Das Formular mit den verschiedenen Inputs wurde im HTML unter der id «memberform» erstellt.

Zeile 78 `<form id="memberform">`

Mitglieder-ID:	<input type="text"/>
Nachname*:	<input type="text" value="erforderlich"/>
Vorname*:	<input type="text" value="erforderlich"/>
Geburtsdatum*:	<input type="text" value="tt.mm.jjjj"/>
Strasse:	<input type="text"/>
Strassennummer:	<input type="text"/>
PLZ:	<input type="text"/>
Ort:	<input type="text"/>
Vereinsposition:	<input type="text"/>
Grad:	<input type="text"/>
Klassifizierung:	<input type="text"/>
Eintritt:	<input type="text" value="tt.mm.jjjj"/>
Austritt:	<input type="text" value="tt.mm.jjjj"/>
Mitglied seit:	<input type="text" value="tt.mm.jjjj"/>
Mitglied bis:	<input type="text" value="tt.mm.jjjj"/>
Erstellt am:	<input type="text" value="tt.mm.jjjj"/>
Aktiv:	<input type="checkbox"/>

View 2 | Validierung - xyzValidation()

Die Funktion dient dazu die Datensätze aus der Eingabe des Benutzers bei Änderung oder Neuerstellung eines Mitgliedes zu überprüfen, um unbrauchbare Datenleichen vermeiden zu können.

Initialisierung:

Die Einfärbung der Feldtypen und der Eintrag der m_id wird bei der Initialisierung aufgerufen.

Input

Der Benutzer gibt neue Daten ein oder ändert diese ab, welche durch die JS Validierung geprüft werden. Zudem ändert sich die Farbe des Randes bei der Eingabe in das Feld.

Zeile 218 `function enterFieldColor()`

Verarbeitung

Die Validierung mittels JS Funktion greift auf die Felder Vorname, Nachname, Strasse, PLZ und Ort. Dabei wird geprüft, ob tatsächlich Buchstaben und Nr. bei den entsprechenden Datenfeldern eingegeben wurden oder nicht. Dies geschieht in einer if Schleife, wobei der eingegebene Value mit einer in Regex definierten Variablen abgeglichen wird.

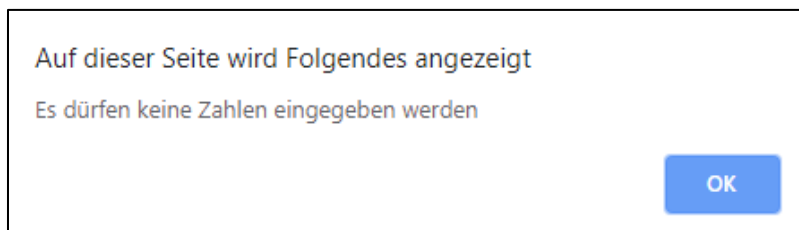
Zeile 230 `function nachNaValidation()`

Weitere kleine Validierungen wie maximale Wortlänge «`maxlength="30"`» oder obligatorische Felder «`required`» wurden direkt im HTML definiert, ohne weiteres Javascript.

Output

Wenn die Eingabe den Regex Definitionen entspricht, kann der Benutzer in das nächste Feld wechseln ohne weitere Aktion.

Wenn die Eingabe des Benutzers nicht mit den Regex Defintionen matched, so wird das Feld rot gefärbt und ein Alert mit weiteren Hinweisen ausgegeben.



Abhängigkeiten:

Die Javascript Validierungsfunktion wird aufgerufen durch onchange im HTML Code, das heisst bei jeglicher Änderung im Feld greift die Funktion. CSS wird dabei nicht tangiert, die Farbänderung wird im Javascript Code umgesetzt.

Zeile 78 `onchange="nachNaValidation()`

View 2 | Erstellen - createMember()

Die Funktion dient dazu ein neues Mitglied in der Datenbank zu erfassen und wird aufgerufen beim Klicken auf den Button «Mitglied hinzufügen».

Initialisierung:

Die Einfärbung der Feldtypen und der Eintrag der m_id wird bei der Initialisierung aufgerufen. Dabei wird das Feld Mitglieder-ID leer ausgegraut und für den Benutzer unveränderbar gemacht.

Zeile 100 `function createMember()`

Input

Sämtliche Inputfelder sind leer und können vom Benutzer ausgefüllt werden gemäss Validierungsregeln. Bei Inputs die ein Datum erfordern, kann der Benutzer einen bereitgestellten Kalender des Browsers nutzen, dieser wird als HTML Datentyp definiert `<type="date">`.

Verarbeitung

Drückt der Benutzer auf den «Speichern» Button, wird das Formular unter der Variablen `member` in ein neues Objekt gespeichert und in der `console.log` zur Prüfung ausgegeben. Die eingegebenen Daten werden anschliessend als URL an den PHP Server geschickt http://767727-7.web1.fh-htwchur.ch/19FS_DBM17TZ_WEBP_Mitglieder/db/setmember.php, geparsed – also die einzelnen Daten aus der URL ausgelesen zur richtigen Erfassung in der Datenbank. Dies unter der Bedingung der Funktion, dass die Member-ID leer ist: `if (member.m_id === "")`.

Zeile 121 - 174

Output

Der Benutzereintrag wurde somit als neues Mitglied in der Datenbank erfasst mit einer neu zugeordneten Mitglieder-ID und ist in der Übersicht der Mitgliederverwaltung unmittelbar ersichtlich. Nach Klicken auf den Button werden zudem die Inputfelder geleert für einen neuen Eintrag.

Abhängigkeiten:

Die Abhängigkeit dieser Funktion besteht mit dem HTML Code über den Button mit der `<id="buttonCreate">` und wird mittels `<onclick>` ausgeführt. Zudem wird mittels URL die Verbindung zur php Datenbank aufgebaut.

View 2 | Ändern / Update - `getSingleMember()`

Die Funktion dient dazu Daten eines bestehenden Mitgliedes aus der Datenbank ändern zu können. Sie wird aufgerufen beim Klicken auf einen bestehenden Datensatz aus der Mitgliederübersicht, unabhängig von der Art des Datensatzes und gespeichert über den Button «Speichern».

Initialisierung:

Die Einfärbung der Feldtypen und der Eintrag der `m_id` wird bei der Initialisierung aufgerufen. Dabei wird das Feld Mitglieder-ID mit der entsprechenden `m_id` ausgefüllt, ausgegraut und für den Benutzer unveränderbar gemacht. Die Daten werden als Single Record von der Datenbank via JSON URL geholt, geparsed und in das Formular abgefüllt zum Ändern.

Zeile 26 `function getSingleMember(m_id)`

Input

Sämtliche Inputfelder, sind gemäss Datensatz vorausgefüllt oder leerstehend und können vom Benutzer angepasst werden gemäss den vorgegebenen Validierungsregeln.

Verarbeitung

Drückt der Benutzer auf den «Speichern» Button, werden die Änderungen des Benutzers in der Datenbank übernommen und unmittelbar in der Benutzerübersicht angezeigt. Dabei wird wiederum das Paket an Datensätzen via JSON an die Datenbank gesendet. In diesem Fall mit dem Unterschied, dass die Bedingung der leeren Mitglieder-ID nicht erfüllt wird und somit der Datensatz der bestehenden `m_id` überschrieben wird. Siehe Funktion `createMember()`, *else Bedingung...*

Output

Der Benutzereintrag der entsprechenden `m_id` wurde somit überschrieben und die Felder des Formulars geleert. Der Benutzer kann die Änderung seines Eintrages in der Benutzerübersicht begutachten und kontrollieren.

Abhängigkeiten:

Die Abhängigkeit dieser Funktion besteht mit dem HTML Code über den Button mit der `«id="buttonCreate"»` und wird mittels `«onclick»` ausgeführt. Zudem wird mittels URL die Verbindung zur php Datenbank aufgebaut.

View 2 | Löschen - deleteMember()

Die Funktion dient dazu Daten eines bestehenden Mitgliedes aus der Datenbank zu löschen. Sie wird ermöglicht beim Klicken auf einen bestehenden Datensatz aus der Mitgliederübersicht, unabhängig von der Art des Datensatzes und ausgeführt nach Klicken auf den Button «Löschen».

Initialisierung:

Die Einfärbung der Feldtypen und der Eintrag der m_id wird bei der Initialisierung aufgerufen. Dabei wird das Feld Mitglieder-ID mit der entsprechenden m_id ausgefüllt, ausgegraut und für den Benutzer unveränderbar gemacht. Die Daten werden als Single Record von der Datenbank via JSON URL geholt, geparsed und in das Formular abgefüllt zum Ändern.

Zeile 26 `function getSingleMember(m_id)`

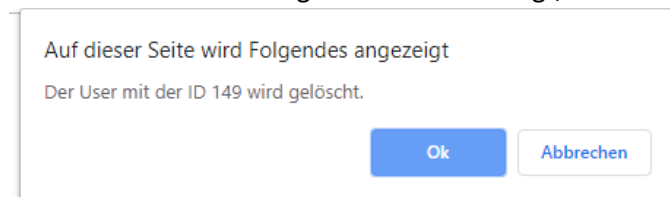
Input

Sämtliche Inputfelder, sind gemäss Datensatz vorausgefüllt oder leerstehend, wobei der Benutzer nun den gesamten Eintrag löschen kann mit Klick auf den Button «Löschen». Insofern es sich um einen leeren Datensatz handelt (m_id als Indikator), so wird ein Alert mit Hinweis ausgegeben.

Zeile 67 `function deleteMember()`

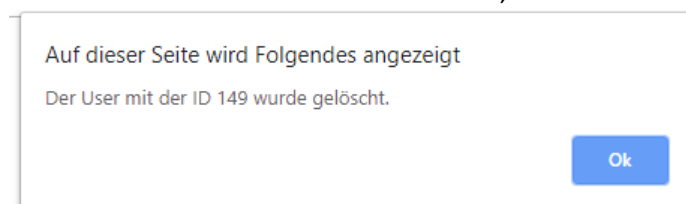
Verarbeitung

Drückt der Benutzer auf den «Löschen» Button, so wird vor der Ausführung der Funktion ein Alert ausgegeben zur Bestätigung des Benutzers, ob der Eintrag wirklich gelöscht werden soll. Insofern der Benutzer die Löschung mit Ok bestätigt, wird der Eintrag in der Datenbank geresetzt.



Output

Der Benutzereintrag der entsprechenden m_id wurde somit komplett aus der Datenbank gelöscht, zur Bestätigung wird dafür ein weiterer Alert ausgegeben. Die Input Felder werden geleert und in der Übersicht kann kontrolliert werden, dass der Benutzer nicht mehr aufzufinden ist.



Abhängigkeiten:


Die Abhängigkeit dieser Funktion besteht mit dem HTML Code über den Button mit der «id="buttonDelete"» und wird mittels «onclick» ausgeführt. Zudem wird mittels URL die Verbindung zur php Datenbank aufgebaut.

Benutzerhandbuch | 1. Übersichtsseite

Auf der Übersichtsseite werden 15 Mitgliedereinträge angezeigt. Man hat die Möglichkeit die Einträge nach Mitglied ID, Nachname oder Vorname zu sortieren. Weiter kann man mit Hilfe des Weiter-Knopfs auf der Mitgliederliste 15 weitere Einträge ansehen und dann wieder 15 Einträge zurück gehen mit dem Zurück-Knopf.

Benutzerhandbuch | 2. Verwaltungsseite

Mit dem Mitglied hinzufügen Knopf oder über die Navigation Verwaltung wird eine neue Verwaltungsseite geladen, auf welcher ein neues Mitglied hinzugefügt werden kann. Bei Doppelklick auf ein bestehendes Mitglied können die dazugehörigen Daten abgeändert und gespeichert werden oder das Mitglied gelöscht werden.


Übersicht Verwaltung

Sortieren -

#	Nachname	Vorname	Geburtsdatum	Ort	Status	Vereinsposition	Klassifizierung
52	niculin	thomann	1991-07-22		0		
53	niculin	Horst	1991-07-22	Zizers	0	Chief	Chief
55	niculin	sdfa	1991-07-22		0		
63	klouoih	dafadf	2019-05-16		0		
64	Mats	Hess	2019-05-13		0		
65	Mats	hess	2019-05-04		0		
66	asdfadf	adsfadsf	2019-05-08		0		
67	u	h	2019-05-24		0		
68			0000-00-00		0		
69	test	test	2019-05-06		0		
70	test	test	2019-05-22		0		
71	tes	test	2019-05-07		0		
72	Niculin	THomann	1991-07-22		0		
73	test	test	2019-05-15		0		
74	n	t	2019-05-01		0		

Mitglied hinzufügen

Zurück Weiter

Übersicht Verwaltung

Mitglieder-ID:

Nachname*:

erforderlich

Vorname*:

erforderlich

Geburtsdatum*:

tt.mm.jjjj

Strasse:

Strassennummer:

PLZ:

Ort:

Vereinsposition:

Grad:

Klassifizierung:

Eintritt:

tt.mm.jjjj

Austritt:

tt.mm.jjjj

Mitglied seit:

tt.mm.jjjj

Mitglied bis:

tt.mm.jjjj

Erstellt am:

tt.mm.jjjj

Aktiv:

☐

Speichern

Löschen

Mitglied hinzufügen

Zurück Weiter

Mitglieder-ID:

Nachname*:

erforderlich

Vorname*:

erforderlich

Geburtsdatum*:

tt.mm.jjjj

Strasse:

Strassennummer:

PLZ:

Ort:

Vereinsposition:

Grad:

Klassifizierung:

Eintritt:

tt.mm.jjjj

Austritt:

tt.mm.jjjj

Mitglied seit:

tt.mm.jjjj

Mitglied bis:

tt.mm.jjjj

Erstellt am:

tt.mm.jjjj

Aktiv:

☐

Speichern

Löschen