Meilenstein 3 – AJAX & Webservices

Ziel ist die weitere Dynamisierung der Oberfläche sowie die Bereitstellung von Artikeldaten in Abalo über einen Webservice. Die Daten sollen via AJAX abgeholt und dargestellt werden können.
Damit ermöglichen wir die Verteilung der Daten in Abalo über eine generelle Schnittstelle. Beliebige weitere Anwendungen, wie Apps (mobilen Applikationen) oder Drittsysteme (wie Werbe- oder Produktportalen) können diese neuen Datenschnittstellen anbinden und die Daten konsumieren.

Aufgabe 1

Übung. Ziel ist das weitere Einüben von JavaScript zur Erprobung von Fähigkeiten der Sprache, um diese bei der Erweiterung von Abalo einsetzen zu können.
Diesmal setzen wir den Fokus auf AJAX, womit wir im Hintergrund aus JavaScript heraus Daten an den Webserver schicken und vom Webserver laden können.

Schreiben Sie ein JavaScript-Programm…

- **3-ajax1-static.html**, das eine statische JSON Datei mit dem Namen „message.json“ mit dem Inhalt

  ```
  { "message": "Willkommen bei Abalo" }
  ```

  über AJAX abfragt und die Nachricht „message“ in einem HTML-Element darstellt. Legen Sie die Datei message.json unter public/static/message.json ab.
- **3-ajax2-periodic.html**, das die Abfrage aus 3-1-ajax.html zyklisch alle 3 Sekunden durchführt. Ändern Sie zum Test den gespeicherten Text in der message.json und überzeugen Sie sich davon, dass das laufende Skript einen veränderten Text überträgt und dargestellt.

Aufgabe 2

Artikeleingabe. Erweitern Sie Abalo. Verändern Sie die Artikeleingabe unter /newarticle so, dass diese nun nicht mehr `submit()` verwendet, sondern die Daten via AJAX zum Webserver sendet.
Kommen Daten via AJAX beim Webserver an, so soll dieser kein HTML, sondern ausschließlich eine Textnachricht, wie „Erfolgreich“ oder „Fehler: …“ zurückgeben. Der Webbrowser soll die Textnachricht unter dem abgeschickten Formular darstellen.

Aufgabe 3

Refactoring: Navigationsmenü als Objekt. Führen Sie ein Refactoring des umgesetzten Menüs durch. Die Funktionalität soll in einem Objekt in JavaScript gekapselt werden. Finden Sie passende Methoden, die das Objekt anbieten kann.

Aufgabe 4 (optional)

Dynamisches Menü. Entwickeln Sie das bereits umgesetzte statische Menü zu einem dynamischen Menü weiter. Bei einem dynamischen Menü können alle Kindknoten durch den/die Nutzer:in ein- und ausgeblendet werden. Die Ein- und Ausblendung geschieht in der Regel auf einen „Klick“ oder eine Mausüberfahrt/-ausfahrt des gewünschten Menüeintrags.

Aufgabe 5

Recherche. Bei der Entwicklung neuer Webanwendungen (wie Abalo) wollen wir (eigentlich gerne) immer die neueste Technologie verwenden. Recherchieren Sie, inwieweit die drei JavaScript-Engines JavaScriptCore, V8 und SpiderMonkey die folgenden Sprachkonstrukte unterstützen:

- Set.prototype.* (wie intersection) – Mengenoperationen.
- Static Blocks in Klassen zur Initialisierung statischer Variablen. Beispiel: `class CL { static { /* … */ } }`
- Array.prototype.flat(depth)
- Array.prototype.group zur Gruppierung von Elementen eines Arrays
- Neues Top-Level-Namespace Objekt “Temporal” als Weiterentwicklung von Date.

Verwenden Sie diese Sprachkonstrukte ab jetzt in der Implementierung? Begründen Sie Ihre Entscheidung. Welche Quellen haben Sie verwendet?

Woche 2

Aufgabe 6

Analyse Webservices. Ziel ist das Kennenlernen existierender REST APIs. Analysieren Sie zwei bestehende REST APIs unterschiedlicher Anbieter, wie Google, Facebook, Twitter, Amazon oder von anderen Anbietern zu einem angebotenen Dienst, wobei Sie die folgenden Fragen zu beiden APIs beantworten:

- Was ist der Zweck der API? Welcher Daten können z.B. ausgetauscht werden?
- Welche REST Prinzipien sind umgesetzt?

- Auf welchem Level (nach dem Richardson Maturity Model) befindet sich die API?
- Wie ist eine Versionierung implementiert?

Hinweis: Die vorgeschlagenen Anbieter besitzen mehrere APIs für unterschiedliche Produkte. Wähen Sie eine aus. Sie sind frei in der Auswahl der API.

Aufgabe 7

Erweitern Sie Abalo. Implementieren Sie eine Artikelsuche als Webservice unter /api/articles über GET mit dem URL-Parameter (?search=), der das Attribut ab_article.ab_name durchsucht. Geben Sie das Ergebnis in einem geeigneten JSON Format zurück.

Aufgabe 8

Erweitern Sie Abalo. Entwickeln Sie einen Webservice, über den ein Artikel angelegt werden kann über POST unter /api/articles. Es sollen die Parameter name, price und description angenommen werden. Der Preis muss größer 0 sein, der Name darf nicht leer sein. Im Erfolgsfall soll die vergebene Id unter dem Attribut „id“ als JSON zurückgegeben werden.

Aufgabe 9 (optional)

Erweitern Sie Abalo. Implementieren Sie das Löschen eines Artikels als Webservice unter /api/articles/{id} über die Methode DELETE mit dem Parameter id. Bieten Sie die Funktionalität unter Berücksichtigung möglicher Fehler mit entsprechenden Rückmeldungen für die Nutzer:innen an, wie „Artikel nicht gefunden“.

Aufgabe 10

Serverseitiger Warenkorb. Wir erweitern den bestehenden clientseitigen Warenkorb um die noch fehlende Serverseite als Webservice. Wiederverwenden Sie Ihre bereits entwickelte Lösung in JavaScript (aus M2).

Erweitern Sie die Struktur der Datenbank.

Hinweis: uint8 steht für einen 8-Byte Unsigned (Big-)Integer. TIMESTAMP steht für einen Zeitstempel ohne Zeitzone.

ab_shoppingcart

Attribut	Typ	Kommentar
id	uint8	Primärschlüssel
ab_creator_id	uint8, not null	Referenz auf den/die Benutzer:in, dem der Warenkorb gehört

Attribut	Typ	Kommentar
ab_createdate	TIMESTAMP, not null	Zeitpunkt der Erstellung

ab_shoppingcart_item

Attribut	Typ	Kommentar
id	uint8	Primärschlüssel
ab_shoppingcart_id	uint8, not null	Referenz auf den Warenkorb
ab_article_id	uint8, not null	Referenz auf den Artikel
ab_createdate	TIMESTAMP, not null	Zeitpunkt der Erstellung

a) Berücksichtigen Sie bei der Datenbankstruktur die folgende Nebenbedingung: Wird ein shoppingcart entfernt, sollen auch alle zugehörigen shoppingcard_items entfernt werden.
b) Erweitern Sie die Artikelübersicht unter /articles/. Fügen Sie für jeden Artikel ein + (Plus) hinzu, über welches der Artikel in den Warenkorb über ein POST nach /api/shoppingcart mit dem Parameter articleid eingetragen wird.
c) Ermöglichen Sie das Entfernen von Artikeln im Warenkorb über ein – (Minus) pro Eintrag über ein DELETE nach /api/shoppingcart/{shoppingcartid}/articles/{articleId}.

Aufgabe

Abgabe. Laden Sie Ihre Ergebnisse als ZIP in ILIAS hoch. Der Name des ZIPs soll sein:

<TeamNr>_<Meilensteinnummer>.zip

Die TeamNr finden Sie in der Teamzuordnung. TeamNr und Meilensteinnummer schreiben Sie bitte ohne führende Null. Also z.B. 4_2.zip für Team 4 Meilenstein 2.

Das ZIP soll alle Ergebnisse des Meilensteins (Antworten zu Freitextaufgaben, Grafiken, Quelltexte, …) enthalten. Fügen Sie zusätzlich ein README ein, wo Sie Vor-, Nachname und Matrikelnummer der beteiligten Teammitglieder hinterlegen.