Klausur „Entwicklung webbasierter Anwendungen, Wintersemester 2023/24

Thomas Hofmann

Hinweise:

- Bearbeitungszeit: 90 Minuten. Insgesamt gibt es 100 Punkte in 6 Aufgaben.
- **Abgabe bis 26.02.24 um 14:00 über GIT in Ihrem persönlichen Repository für die Prüfung**
- Erlaubte Hilfsmittel: Ein Notebook pro Person, das Internet, sowie alle auf dem Notebook mitgebrachten Informationen und jede auf Papier gedruckte Quelle.
- **Kommunikation mit anderen Menschen, außer der Prüfungsaufsicht, ist nicht erlaubt! Die Verwendung von Chats o.ä. wird als Täuschungsversuch gewertet.**
- Lesen Sie zuerst die gesamte Aufgabenstellung bevor Sie mit der Lösung der Aufgaben beginnen.
- Bearbeiten Sie die Aufgabe im Ordner **src/Exam** des Repositories, das ihnen zugewiesen wurde. Legen Sie keine neuen Dateien an und benennen Sie die Dateien nicht um. Commiten bzw. pushen Sie ihren Code via Git regelmäßig (ca. alle 10 Minuten) in Ihr zugeteiltes Repository. Die Abgabe erfolgt ausschließlich über Git. Commits mit einem Zeitstempel nach Ablauf der Bearbeitungszeit können nicht berücksichtigt werden.
- **Es wird nur bewertet, was sich in dem für Sie angelegten Gitlab-Repository in den vorbereiteten Dateien bis zum Abgabetermin unter src/Exam befindet.**
- Schreiben Sie standardkonformen Code, der die Regeln der Veranstaltung für ordentlichen Code und die Vorgaben durch die Seitenklassen berücksichtigt.
- Diese Aufgabenblätter werden nicht abgegeben. Schreiben Sie also keine Lösungsteile darauf. Sie dürfen die Seiten natürlich auch trennen.

Zulieferung

In Ihrem GitLab-Projekt sind die Dateien, die Sie bearbeiten sollen, schon angelegt:

- Im Moodlekurs der Veranstaltung finden Sie eine ZIP-Datei mit vorbereiteten Inhalten für diese Dateien sowie ein SQL-Skript zum Erzeugen und Füllen der Datenbank. Kopieren Sie zunächst diese Dateien in das Verzeichnis src/Exam und überschreiben Sie die vorhandenen Dateien.
- Wenn Sie die Datenbank mit dem SQL-Skript importieren wollen melden Sie sich im phpmyadmin mit dem **Benutzer: root** und **Passwort: ganzGeheim** an. Anschließend nutzen Sie die „Importieren“ Funktionalität, wie wir es kennengelernt haben.
- Page.php: Die bekannte Basisklasse der Seitenklassen
- Exam.php – siehe Aufgabe 1 und Aufgabe 4
- ExamAPI.php – siehe Aufgabe 2
- ExamService.php – siehe Aufgabe 4
- Exam.js – siehe Aufgabe 3 und 4
- Exam.css – siehe Aufgabe 4 und 5
- Exam.txt – siehe Aufgabe 4

Die Aufgabe allgemein: „HDA - Electronis“

In dieser Klausur sollen Sie eine Webseite für einen Elektronikfachhandel entwickeln. Die Seite umfasst eine Liste der zum Verkauf stehenden Artikel sowie ein Suchfeld, um mithilfe der Artikelnummer schnell den gewünschten Gegenstand zu finden. Außerdem gibt es noch eine „Zu den Favoriten hinzufügen“ Funktionalität.

Visuelle Darstellung

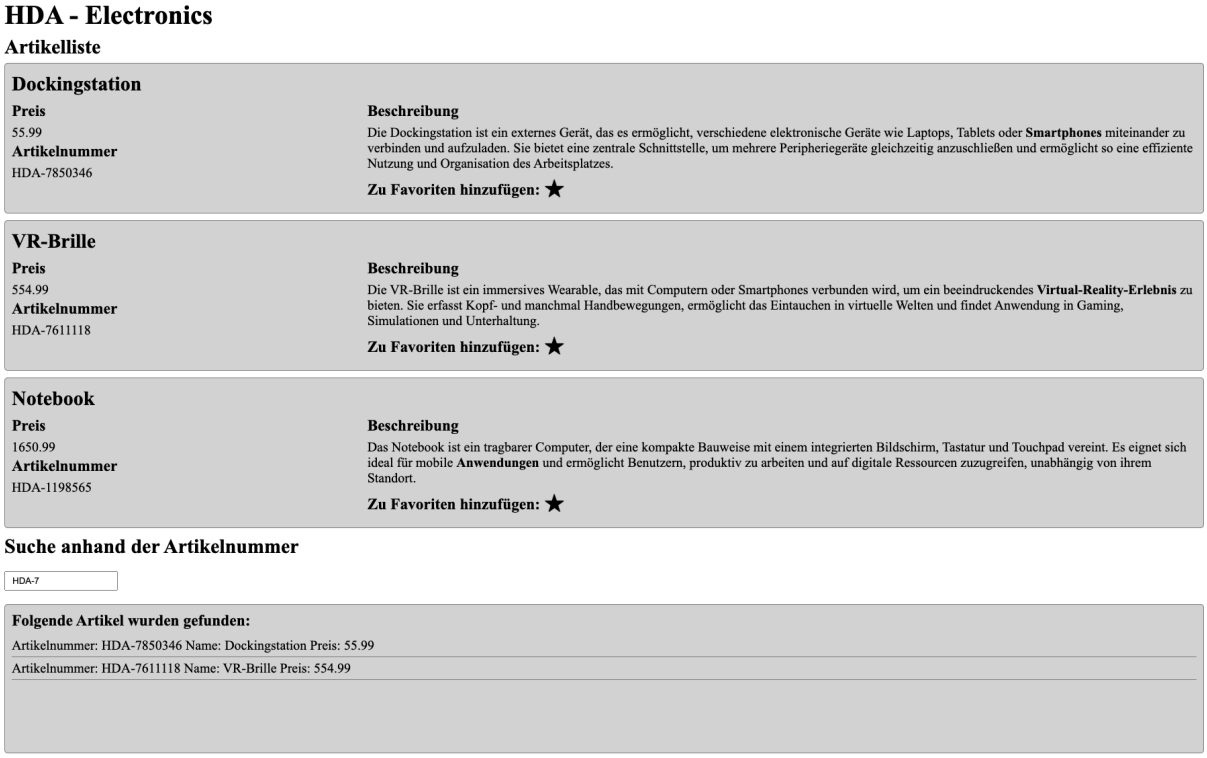
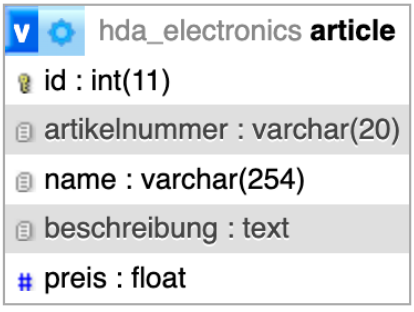


Abbildung 1: Interface samt Artikelliste und beispielhafter Ausgabe der Artikelsuche

Datenbankstruktur

Die Datenbank heißt hda_electronics und besteht aus der Tabelle: article die wie folgt aufgebaut ist:

id	artikelnummer	name	beschreibung	preis
1	HDA-7850346	Dockingstation	Die Dockingstation ist ein externes Gerät, das es ...	55.99
2	HDA-7611118	VR-Brille	Die VR-Brille ist ein immersives Wearable, das mit...	554.99
3	HDA-1198565	Notebook	Das Notebook ist ein tragbarer Computer, der eine ...	1650.99



Aufgabe 1: HTML/PHP (Exam.php) (20 Punkte)

Implementieren Sie in der Datei **Exam.php** unter Verwendungen unserer **Seitenklassenstruktur,** sowie **PHP** und **semantischem HTML,** die in der **Abbildung 1** dargestellte Webseite. Die Seite hat folgende Struktur:

- Einen Kopfteil mit einer Hauptüberschrift.
- Eine Artikelliste, welche dynamisch aus den Daten der Datenbanktabelle: article generiert wird.
    - Diese beinhaltet alle Artikel, welche man kaufen kann samt folgender Eigenschaften: Name, Artikelnummer, Preis, Beschreibung und ein „Zu Favoriten hinzufügen“-Element.
    - Die Erzeugung der Liste sollte gegen das Cross-Site-Scripting abgesichert sein. Ausnahme hier ist die Beschreibung des Artikels, dort möchten wir die Ausführung von HTML-Code erlauben.
    - Der Favorit-Stern soll später via JavaScript angesprochen werden können. Dieser Unicode Character kann mittels HTML-Code ★ oder ★ realisiert werden und verhält sich wie ganz normaler Text.
    - Die Artikelliste soll später mittels CSS, Flexbox und Mediaqueries gestylt und responsive gemacht werden. Überlegen Sie sich jetzt schon eine sinnvolle HTML-Struktur. (Siehe Abbildung 1)
- Ein Eingabefeld für die Suche via Artikelnummer
    - Dieses Eingabefeld soll später via JavaScript angesprochen werden können.
    - Als Platzhalter hat es den Wert „Artikelnummer eingeben“
- Ein leeres Element, welche später als Ausgabe für die Artikelsuche dienen soll.
    - Dieses Element soll via JavaScript angesprochen werden können.

Anmerkungen zur Aufgabe 1

- Die Artikelliste ist kein <ul> oder <ol> Listen-Element.
- Wenn Ihnen das mit dem Stern HTML Code zu kompliziert ist, nutzen sie den Buchstaben X statt dem Stern.
- Wie oben bereits erwähnt wird das Aussehen später via CSS realisiert, also machen Sie sich keine Sorgen, wenn Ihre Ausgabe erst mal nicht so aussieht wie in der Abbildung.

Aufgabe 2: PHP (ExamAPI.php) (15 Punkte)

Implementieren Sie in der Datei **ExamAPI.php** unter Verwendung unserer **Seitenklassenstruktur** ein **PHP-Skript** welches als **Response** ein **JSON-String** zurückliefert. Folgendes ist zu beachten:

- Im späteren Verlauf (Aufgabe 3) soll via JavaScript ein Ajax-Request an dieses PHP-Skript geschickt werden. Dieser Request beinhaltet die Artikelnummer als GET-Parameter. Anhand des GET-Parameters sollen Sie alle passenden Artikel in der Datenbank suchen und als JSON-String zurückliefern.
- Achtung: Es wird entweder keiner, einer oder alle Artikel zurückgeliefert, die mit Wert vom GET-Parameter anfangen. (Anmerkungen beachten)
- Sichern Sie das Skript gegen jegliche SQL-Injection Angriffe ab.

Anmerkungen zu Aufgabe 2

- Nutzen Sie den LIKE-Operator für ihre SQL-Abfrage.
  Zum Beispiel: **SELECT * FROM article WHERE artikelnummer LIKE 'HDA-7%'**;
  würde alle Items die mit der Artikelnummer „HDA-7“ anfangen ausgeben.
- Denke Sie dran in diesem Beispiel ist das „HDA-7“ in dem Fall ihr GET-Parameter welchen Sie wie gelernt in die Query integrieren müssen.
- Zum Testen ihrer Lösung können Sie die URL mit verschiedenen GET-Parametern aufrufen und schauen ob Sie die richtigen Daten im JSON-String zurückbekommen.
  Zum Beispiel: **ExamAPI.php?MeinGetParameterName=HDA-7611118** usw.

Aufgabe 3: JavaScript (Exam.js) (20 Punkte)

Implementieren Sie in der **Exam.js** Datei via **JavaScript** ein **Ajax-Request**, welcher an **ExamAPI.php** abgeschickt wird. Fortan sollen die angefragten Daten verarbeitet und mittels **HTML** in dem leeren Ausgabe-Element visualisiert werden. Folgendes ist zu realisieren:

- Es soll bei jedem Eingabe-Event auf das Such-Eingabefeld ein Request abgeschickt und das Ausgabe-Element aktualisiert werden.
- Mit dem Request zusammen soll als GET-Parameter das aktuelle Value des Suchfeldes mitgeschickt werden.
- Wurden Daten zurückgeliefert werden diese im Ausgabe-Element samt Artikelnummer, Name und Preis visualisiert. Denken Sie dran, dass Sie auch mehrere Artikel zurückbekommen können.
- Die gefundenen Artikel sollen eine CSS-Klasse bekommen, welche ihnen unten einen schwarzen Rand gibt. Das dazugehörige CSS machen Sie später in Aufgabe 5.
- Wurden keine Daten gefunden steht dort „Es wurden keine Artikel gefunden“.
- Denke Sie dran, dass wir bei jeder Eingabe ein Request abschicken und das Ausgabe-Element manipulieren also muss auch der alte Inhalt immer gelöscht werden bevor neuer eingefügt wird.

Anmerkungen zur Aufgabe 3

- Für eine bessere Verständnis schauen Sie sich in Ruhe die Abbildung 1 an sowie die Testeingabe im Suchfeld und was diese im Ausgabe-Element ausgibt. Vergleichen Sie das mit dem Inhalt der Datenbank.
- Fallbeispiel: GET-Parameter „HDA-7“ liefert zwei Artikel, der „HDA-78“ würde nur einen eindeutigen Artikel zurückliefern.

Aufgabe 4: Alles zusammen (25 Punkte)

Implementieren Sie eine Funktionalität welche es erlaubt, durch einen Klick auf den Stern in der Artikelliste einen Artikel zu den Favoriten zuzufügen und wieder zu entfernen. Dieser Status soll auch bestehen, wenn Sie die Seite neu laden.
Folgendes ist zu beachten:

- Es steht Ihnen frei die **Exam.php, Exam.js, Exam.css** entsprechend anzupassen.
- Sollten Sie für ihre Lösung eine zusätzliche serverseitige Datei benötigen, nutzen Sie die **ExamService.php**
- Zum Zwischenspeichern vom Status soll nicht die Datenbank genutzt werden.
- Der Stern bekommt beim Klick eine gelbe Farbe und behält diese auch solange er serverseitig als Favorit markiert ist. Wird er nochmal angeklickt bekommt er wieder seine ursprüngliche schwarze Farbe und wird serverseitig von den Favoriten entfernt.
- Es können auch mehrere Artikel zu den Favoriten zugefügt werden.
- Dieser Status und die Farbe sollen auch nachdem die Seite neu geladen wurde bestehen und erst „zerstört“ werden, wenn der Browser geschlossen wurde.
- **Wichtig: Bevor Sie loslegen, beschreiben Sie in der Exam.txt ihr Vorgehen in eigenen Worten so detailliert wie möglich. (Die Beschreibung gibt auch Punkte)**
    - Wie wollen Sie die Aufgabe lösen?
    - Welche Dateien nutzen Sie dazu?
    - Welche Funktionalität wird in welcher Datei realisiert?
    - In welchen Dateien führen Sie Anpassungen am bestehenden Code durch?

Anmerkungen zur Aufgabe 4

- Die Aufgabe ist extra freizügiger formuliert um ihr Wissen zu testen. Sie können alle Werkzeuge nutzen, welche wir in der Vorlesung kennengelernt haben. Auch eventuell diejenigen, welche in dieser Klausur bereits Verwendung gefunden haben.

Aufgabe 5: CSS (Exam.css) (10 Punkte)

Implementieren Sie in der **Exam.css** Datei via **CSS** das Styling der Webseite. Die **Abbildung 1** sollte Ihnen als Vorlage dienen. Folgendes soll realisiert werden:

- Die ganze Seite hat die Schrift 'Times New Roman', serif
- Die Artikelelemente haben die Hintergrundfarbe lightgray und einen abgerundeten schwarzen Rand der 1px breit ist.
- Der Inhalt der Artikelelemente ist in zwei Flex-Items unterteilt. Das linke Item beinhaltet den Preis und Artikelnummer und das rechte Item die Beschreibung und das Favorit-Feld. Die beiden Items sollen bis zu einer Bildschirmbreite von 550px untereinander sein und danach nebeneinander.
- Das Ausgabefeld für die Suche soll eine Mindesthöhe von 200px haben, einen lightgray Hintergrund und einen 1px schwarzen abgerundeten Rand.
- Der ganze Inhalt der Seite soll ab 600px Bildbreite nicht breiter wie 70% vom Viewport sein.
- Versuchen Sie alle Abstände ungefähr so zu realisieren wie in der Abbildung 1.
- Die in Aufgabe 3 gefragte CSS-Klasse hat einen 1px schwarzen Rand unten.

Aufgabe 6: Lauffähigkeit/Vollständigkeit (10 Punkte)

Gehen Sie diese Aufgabe erst an, wenn Sie die übrigen Aufgaben nach bestem Wissen gelöst haben.

Prüfen Sie die Lauffähigkeit ihrer Anwendung indem Sie das Docker-Setup starten und im **localhost** ihre Anwendung begutachten. Folgendes soll geprüft werden:

- Prüfen Sie ob Ihr erzeugter HTML-Code valide und barrierefrei ist mit dem Validator den wir in der Vorlesung kennengelernt haben. (https://validator.w3.org/#validate_by_input)
- Sieht das Styling ihre Lösung ungefähr so aus wie in der Abbildung 1?
- Vergewissern Sie sich ob die Anwendung funktionsfähig ist indem Sie nach Artikeln via dem Suchfeld suchen. Werden alle Artikel wie gewünscht gefunden und ausgegeben?
- Läuft die Favorit-Funktionalität wie gewünscht? Laden Sie die Seite neu und prüfen Sie ob alles so sein soll wie gewünscht.

Abgabe

Laden Sie Ihr Endergebnis in **Git** hoch und prüfen Sie über **GitLab (https://code.fbi.h-da.de)**, ob die Dateien auch wirklich angekommen sind.

Sollte wirklich was schief gehen beim hochladen, sagen Sie ihrem Betreuer Bescheid. Wir können notfalls ihre Abgabe im Zip-Dateiformat auf ein USB-Stick laden oder schicken Sie eine Mail mit ihrer Lösung und dem Betreff „EWA Klausurabgabe Nachname Vorname“ sowie Ihrer Matrikelnummer an thomas.hofmann@h-da.de