

Klausur "Entwicklung Webbasierter Anwendungen", Sommersemester 2022

Prof. Dr. Ralf Hahn, Prof. Dr. Ute Trapp, Thomas Hofmann

Hinweise:

- Bearbeitungszeit: 90 Minuten + 10 Minuten für Down- und Upload. Insgesamt gibt es 100 Punkte in 5 Aufgaben.
- **Abgabe bis 8.8.22, 14:10 Uhr über Git in Ihrem persönlichen Repository für die Prüfung.**
- Erlaubte Hilfsmittel: Ein Notebook pro Person 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. Committen 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. Verwenden Sie Exceptions zur Behandlung von ungewöhnlichen Fehlern. Entwickeln Sie eine barrierefreie, sichere Anwendung.
- In den PHP-Seitenklassen ist die strenge Typüberprüfung aktiviert. Diese Überprüfung darf nicht abgeschaltet oder umgangen werden. Ändern Sie nicht die Signaturen der vordefinierten Methoden.
- Sie können sowohl in Deutsch als auch in Englisch antworten (You may also answer in English).
- Diese Aufgabenblätter werden nicht abgegeben. Schreiben Sie also keine Lösungsteile darauf. Sie dürfen die Seiten natürlich auch trennen.

Wichtig! Schreiben Sie zuerst den Code und versuchen Sie erst am Schluss den Code zum Laufen zu bringen!

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.
- Page.php: Die bekannte Basisklasse der Seitenklassen (Sie müssen diese Datei nicht mehr bearbeiten!)
- Exam22.html - siehe Aufgabe 1
- Exam22API.php - siehe Aufgabe 2
- Exam22.js - siehe Aufgabe 3
- Exam22.css - siehe Aufgabe 4

Die Aufgabe allgemein: "Leichte Sprache"

Schreiben Sie einen Dienst, der eine Erläuterung in Leichter Sprache für komplizierte Wörter in eine Webseite einfügt. Wer den Dienst nutzen möchte, muss nur Ihr Skript einbinden und einen Container gemäß Aufgabe 3 in seine HTML-Seite einfügen. Da Sie nur wenig Zeit zur Verfügung haben, ist die Aufgabenstellung vereinfacht.

Aufgabe 1: HTML (18 Punkte)

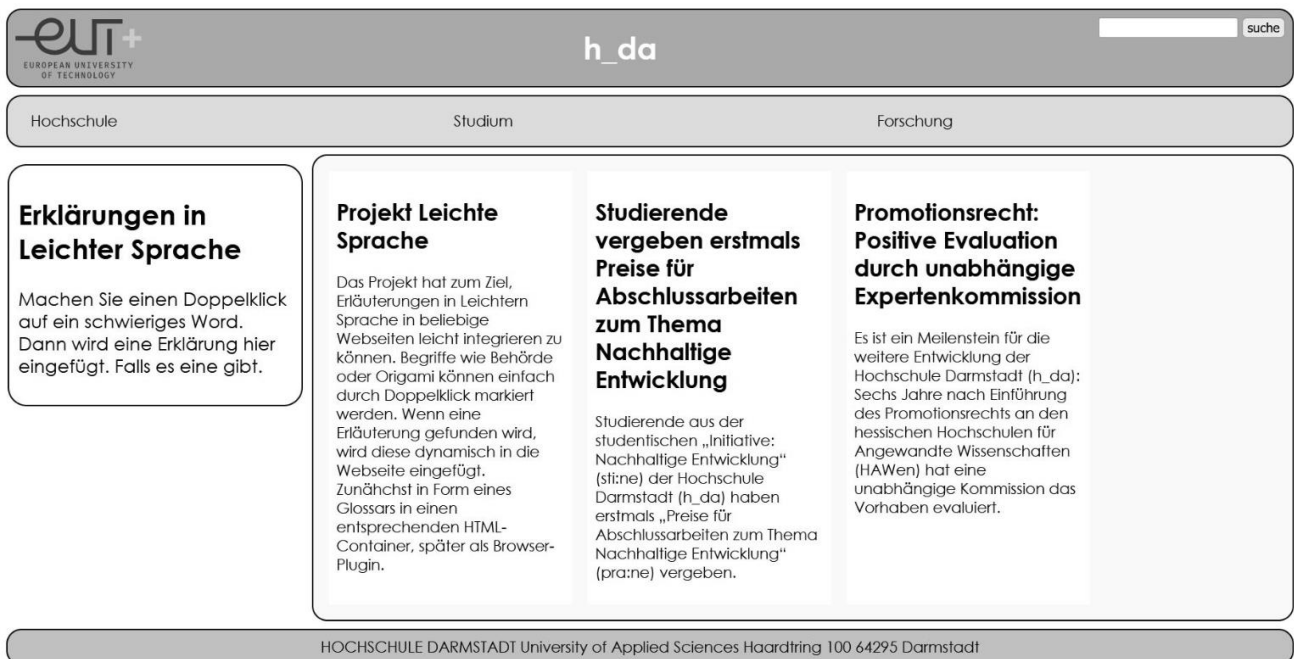


Abbildung 1: Basisseite

Erstellen Sie eine statische und unformatierte HTML-Seite (keine PHP-Seite, keine Seitenklasse, (noch) kein Design und kein Layout!) mit den Inhalten aus Abbildung 1. Die Texte sind bereits in der zugelieferten HTML-Datei enthalten (Exam22.html). Fügen Sie in dieser Datei entsprechende Tags und Attribute ein.

Hinweise:

- Sie sollen für diese Seite wirklich nicht die Seitenklassen verwenden
- Der Pfad zum EUT+-Logo ist ebenfalls in der HTML-Datei gegeben
- Das Styling erfolgt erst in Aufgabe 4. Sie dürfen die Größe des Logos aber auch jetzt schon setzen.
- Die Links, Form- und Navigationselemente sollen ohne echte Funktion auf sich selbst bzw. die Seite Exam22.html verweisen

Aufgabe 2: PHP (25 Punkte)

Implementieren Sie den serverseitigen Teil Ihres Dienstes als Seitenklasse in PHP in der Datei Exam22API.php. Die zugelieferte Datei enthält das bekannte PageTemplate der Seitenklassen. Die zugelieferte Page.php enthält bereits die üblichen, benötigten Methoden. Realisieren Sie den Service so, dass die Get-Anfrage <http://localhost/Exam/Exam22API.php?search=origami> sinnvoll beantwortet wird.

Suchen Sie das übergebene Wort in der Datenbank-Tabelle *Dictionary*.

2022_Easy2Read Dictionary	
id	int(11)
word	varchar(100)
explanation	varchar(1024)

Abbildung 2: Datenbank-Schema

Die Suche soll case-insensitive und getrimmt (Leerzeichen am Anfang und Ende werden ignoriert) erfolgen (Tipp: Es gibt strtolower() und trim() in PHP und LOWER bzw. TRIM auch für SQL-Queries). Wenn Sie den gesuchten Text gefunden haben, geben Sie ein JSON-Objekt des gefundenen Datensatzes mit den drei Datenbankwerten für *id*, *word* und *explanation* zurück. Wenn es keine Übersetzung gibt, geben Sie den Status-Code 404 zurück.

Testen Sie Ihren Dienst direkt über den Browser, z.B. mit dem Aufruf <http://localhost/Exam/Exam22API.php?search=origami> oder <http://localhost/Exam/Exam22API.php?search=xxx> – dieser Aufruf zeigt in der Netzwerkanalyse den Status 404 als Antwort an.

Klausur "Entwicklung Webbasierter Anwendungen", Sommersemester 2022

Prof. Dr. Ralf Hahn, Prof. Dr. Ute Trapp, Thomas Hofmann

Aufgabe 3: JavaScript (27 Punkte)

Nachdem ein Wort mit Doppelklick ausgewählt wurde, soll eine Erläuterung des Wortes dynamisch in die Seite eingefügt werden (siehe Abbildung 3). Dazu wird per Ajax nachgefragt, ob es eine Erklärung zu diesem Wort gibt. Es soll immer nur eine Erläuterung angezeigt werden (d.h. das zuletzt angeklickte Wort).

Hierzu sind folgende Teilschritte zu implementieren:

- a) Implementieren Sie eine Funktion *wordClickHandler*, welche das angeklickte Wort ausliest und einen entsprechenden Ajax-Request erzeugt. Das ausgewählte Wort bekommen Sie über *window.getSelection().toString()*. Die Ajax-Anfrage soll nur ausgelöst werden, wenn die Auswahl mindestens 2 Buchstaben enthält.
Hinweis: Zu Testzwecken können Sie den Funktionsaufruf temporär in der HTML-Datei vornehmen (vgl. Teil b). Beachten Sie, dass in der Datenbank nur Erläuterungen für die Worte Origami, Behörde und HTML existieren!
- b) Registrieren Sie die Methode *wordClickHandler* mit JavaScript für das Ereignis *dblclick* (dieses feuert bei einem Doppelklick auf ein Wort) für das body-Tag (So kann Ihr Service alleine durch das Einbinden der JavaScript-Datei hinzugefügt werden).
Tipp: Verwenden Sie *window.onload*.
- c) Passen Sie den Container für den Erklärungstext (links in Abbildung 1) so an, dass Sie mit JavaScript darauf zugreifen können. Zeigen Sie die zuletzt erhaltene Erläuterung in diesen Container an (vgl. Abbildung 3). Beim Status 404 geben Sie eine entsprechende Meldung als *console.log* aus. Beachten Sie, dass die Erklärungen auch Sonderzeichen wie <, > oder & enthalten können.

Aufgabe 4: CSS (20 Punkte)

Setzen Sie Design und Layout gemäß Abbildung 3 grob um. Abstände dürfen Sie im Detail ignorieren. Beachten Sie insbesondere folgende Anforderungen:

- Verwenden Sie neben Weiß und Schwarz die Farbcodes, die in Abbildung 3 auf der rechten Seite vorgegeben sind.
- Die Basisschrift soll möglichst *Century Gothic* sein, alternativ eine andere schnörkelfreie Schrift.
- Beachten Sie die runden Ecken an vielen Elementen
- Vergrößern Sie die Schrift der Erläuterungen in einfacher Sprache etwas.
- Machen Sie Ihr Layout responsive
 - Die Beiträge in der Mitte (Projekt Leichte Sprache usw.) können in der Anzahl variieren. Verwenden Sie Flexbox, um diese Blöcke anzuordnen.
 - Zeigen Sie die Einträge in der Leiste (Hochschule / Studium / Forschung) für schmale Bildschirme (bis zu 768px Auflösung) untereinander an.



Abbildung 3: Basisseite mit beispielhafter Erläuterung für "Origami"

Aufgabe 5: Lauffähigkeit / Vollständigkeit (10 Punkte)

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

- Prüfen Sie den HTML- und CSS-Teil (auch ohne Docker), indem Sie die HTML-Datei im Browser öffnen. Vergleichen Sie die angezeigten Informationen mit Abbildung 1 und Abbildung 3. Überprüfen Sie, ob der Code valide und barrierefrei ist.
- Damit der JavaScript- und Ajax-Code laufen kann, müssen Sie die Docker-Container starten und die zugelieferte Datenbank inkl. Tabellen und Beispieldaten importieren. Melden Sie sich dazu als root in der phpMyAdmin-Oberfläche an und führen Sie im Reiter SQL das mit der Aufgabenstellung zugelieferte SQL-Skript aus. Starten Sie nun Ihre Webanwendung und gehen Sie folgende Checkliste durch:
 - Führen Sie einen Doppelclick auf dem Wort Origami aus – wird im Debugger ein entsprechender Request angezeigt?
 - Wird die zurückgelieferte Erklärung in die Seite eingefügt?
 - Führen Sie einen Doppelclick auf ein unbekanntes Wort aus - erscheint in der Konsole des Debuggers eine entsprechende Meldung?

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.