Otto-Friedrich-Universität Bamberg
Lehrstuhl für Medieninformatik
Prof. Dr. Andreas Henrich
Hans-Christian Sperker, M.Sc.

Web Technologien, SS 2013

Teilleistung 2

Hinweise

- Die vollständige und korrekte Bearbeitung einer Aufgabe ergibt die volle Punktzahl dieser Aufgabe; in Ausnahmefällen werden halbe Punkte verteilt.
- Pro Teilleistung erwarten wir einen **Bearbeitungsaufwand von etwa 30 Stunden**. Dies ist als Richtwert zu verstehen und keinesfalls eine Obergrenze.
- Eine Bearbeitung der Aufgabenstellung in der Gruppe ist zulässig. Die Gruppeneinteilung muss den Dozenten mitgeteilt werden.
- In Verdachtsfällen behalten wir uns vor, Plagiate von der Benotung auszunehmen. Wir werden in einem solchen Fall Kontakt mit den Betroffenen aufnehmen.
- Falls in der Aufgabenstellung gefordert, reichen Sie eventuelle *schriftliche Ausarbeitungen* als ein unkomprimiertes PDF-Dokument ein, das sämtliche schriftlichen Lösungen enthält. Bitte verteilen Sie die Lösungen zu einzelnen Aufgaben nicht auf verschiedene Dokumente. Sie können PDF-Dokumente bspw. mit dem PDFCreator (http://www.pdfforge.org/) oder auch FreePDF (http://freepdfxp.de/) erstellen.
- *Programmieraufgaben* sollen in gut kommentierter Form im Quelltext abgegeben werden. Verwenden Sie keine anderen als die angegebenen Bibliotheken, soweit sie nicht zum Standardsprachumfang gehören. Nicht kommentierter Quellcode oder nicht lauffähige Programme können nicht bewertet werden. Fügen Sie, falls notwendig, eine README-Datei hinzu, die beschreibt, wie man ihre Lösung kompilieren bzw. ausführen kann.

Abgabe der Lösungen

- *Abgabetermin* **25. Juni 2013, 23:55 CEST**
- *Abgabemodus* Online-Einreichung auf dem Virtuellen Campus im entsprechenden Kurs. Der Server ist erreichbar unter http://vc-neu.uni-bamberg.de/moodle/. Es ist ausreichend, wenn eines der Gruppenmitglieder die Abgabe einreicht. Bei mehreren Abgaben wird die letzte Abgabe gewertet.
- *Dateiformat der Abgabe* Die Abgabe erfolgt durch den Upload eines ZIP-Archivs, das alle notwendigen Dateien beinhaltet. Das ZIP-Archiv sollte das Namensschema **nachname_vorname_2.zip** bzw. **nachname1_nachname2_nachname3_2.zip** einhalten.
- *Update der Lösung* Bis zur oben angegebenen Deadline können Sie Ihre Lösung beliebig oft durch eine neue (eventuell korrigierte) Fassung ersetzen. Bitte beachten Sie, dass Sie in diesem Fall die bereits auf dem Server vorhandene alte Lösung überschreiben. Wir haben keine Möglichkeit, alte Fassungen wiederherzustellen!

Wir wünschen Ihnen viel Erfolg
bei der Bearbeitung der Aufgaben!

Aufgabe: Programmierung eine Veranstaltungsverwaltung (40 Punkte)

Entwickeln Sie unter Verwendung der vorgegebenen Server-Seite (siehe API Dokumentation des Servers) eine AJAX-Anwendung zur Verwaltung von Veranstaltungen sowie Aufgaben hierzu.

Folgende Funktionen und Abläufe sollen dabei bereitgestellt werden:

- Es soll einem Besucher möglich sein, sich zu registrieren (`users.php / register`) oder sich als schon registrierter Benutzer am System anzumelden (`users.php / login`).
- Ein Benutzer soll die Möglichkeit haben, sich vom System abzumelden (`users.php / logout`).
- Einem angemeldeten Benutzer sollen alle Veranstaltungen angezeigt werden, denen er zugesagt hat (`users.php / list_events_of_user`).
- Einem angemeldeten Benutzer sollen alle Aufgaben angezeigt werden, deren Erfüllung er zugesagt hat (`users.php / list_tasks_of_user`).
- Einem angemeldeten Benutzer sollen darüber hinaus alle im System eingetragenen Veranstaltungen angezeigt werden (`events.php / list_events`).
- Nach der Auswahl einer Veranstaltung sollen einem Benutzer die Beschreibung des Events angezeigt werden, sowie alle Besucher dieses Events (`events.php / list_users_of_event`). Zusätzlich werden alle Aufgaben des ausgewählten Events angezeigt (`events.php / list_tasks_of_event`), sowie die Anzahl der Personen die diese Aufgabe übernehmen sollen. Sollten schon Benutzer Aufgaben übernommen haben sollten die Name der Benutzer bei der jeweiligen Aufgabe angezeigt werden (`tasks.php / list_users_of_task`).
- Ein angemeldeter Benutzer soll die Möglichkeit haben, eine Veranstaltung zu erstellen (`events.php / create`).
- Ein angemeldeter Benutzer soll an einer Veranstaltung teilnehmen können (`events.php / join`).
- Hat ein Benutzer eine Veranstaltung angelegt, so soll er die Möglichkeit haben, Aufgaben für diese Veranstaltung zu definieren (`tasks.php / create`). Eine Aufgabe hat dabei eine definierte Anzahl an Slots die angibt, von wie vielen Teilnehmern eine Aufgabe erledigt werden kann.
- Nimmt ein Benutzer an einer Veranstaltung teil, so hat er die Möglichkeit Aufgaben dieser Veranstaltung zu übernehmen (`tasts.php / assign`).

Technische Umsetzung

Einige Details zur technischen Umsetzung:

- Es muss lediglich die Client-Seite unter Verwendung von HTML, CSS sowie JavaScript implementiert werden.
- Das Anlegen von Benutzern, Veranstaltungen sowie Aufgaben soll unter Verwendung der vorgegebenen Server-Seite realisiert werden. Somit ist eine Speicherung der Entitäten in der Datenbank gewährleistet.
- Voraussetzung für diese Teilleistung ist ein lauffähiger Web- sowie Datenbank-Server. Um dies auf einfache Weise zu ermöglichen wird empfohlen XAMPP zu verwenden (http://www.apachefriends.org/de/xampp.html). Nach der Installation sowie Konfiguration muss lediglich noch das Verzeichnis Events des im VC bereitgestellten Archiv Events.zip in das htdocs Verzeichnis innerhalb des XAMPP Ordners extrahiert werden. Nach dem Starten des Web- sowie MySQL-Server über das XAMPP Control Panel ist die Anwendung unter http://localhost/Events/ erreichbar.
- Die Client-Seite sollte direkt im Verzeichnis Events implementiert werden. Die unterschiedlichen Controller befinden sich im Verzeichnis ctrls.
- Um die Datenbank für diese Teilleistung anzulegen, muss zunächst die Datei _config.php im ctrls-Verzeichnis angepasst werden. Hier muss der Benutzer sowie das Passwort für die MySQL

Datenbank angegeben werden. Ist dies geschehen kann die Datei _setup.php über den Browser aufgerufen werden. Diese sollt nach einem kurzen Moment das erfolgreiche Anlegen der Datenbank sowie der benötigten Tabellen bestätigen.

- Anschließend sollten die unterschiedlichen Controller einsatzbereit sein.

Die Umsetzung muss mit Hilfe folgender Komponenten erfolgen:

Webseite	HTML5 + CSS3 + JavaScript
JavaScript-Bibliothek	jQuery 1.10.0
Client-/Serververbindung	AJAX-Requests
Serverseitige Implementierung	PHP (vorgegeben)
Datenbank	MySQL Community Server (siehe XAMPP)
Datenbankanbindung	mySQLi (vorgegeben)
Webserver	Apache (siehe XAMPP)

Die Verwendung aller Komponenten ist zwingend vorgeschrieben. Abgaben, die mit anderen Komponenten erstellt wurden, können nicht gewertet werden.

Der Schwerpunkt der Aufgabenstellung liegt in der Erstellung der JavaScript- sowie AJAX-Funktionalität. Selbstverständlich sind in vorherigen Studienleistungen überprüfte Komponenten aber ebenso Bestandteil dieser Studienleistung. Es wird ein valides und standardkonformes HTML-Dokument erwartet sowie CSS- und JavaScript-Dateien, die die gängigen Standards valide umsetzen.

API Dokumentation des Server

In diesem Abschnitt wird auf die unterschiedlichen Controller eingegangen die sich im Verzeichnis ctrls des Archiv Events.zip befinden. Die den Controllern zugrunde liegende Datenbankstruktur ist in Abbildung 1 dargestellt.

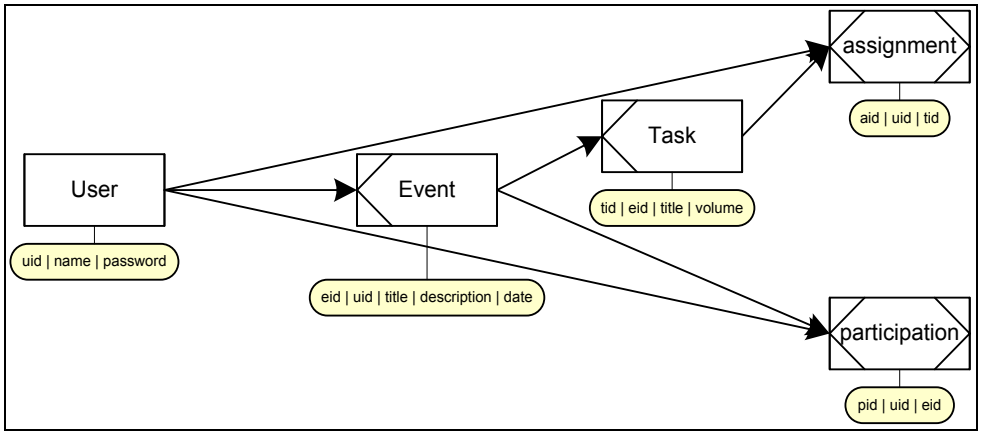

Abbildung 1 – SERM Struktur der Datenbank.

Genereller Aufruf einer Methode eines Controllers auf der Server-Seite, hier am Beispiel der Methode `register` des `user`-Controllers.

```
$.getJSON('ctrls/users.php', {
        operation: 'register',
        name: 'Nutzername',
        password: 'Passwort'
    },
    function(data) {
        // hier kann die Antwort des Servers verarbeitet werden!
    }
);
```

Die unten stehende Tabelle listet die verschiedenen Controller mit den jeweils implementierten Methoden. Der Controller bestimmt die Datei welche per `$.getJSON(…)` angesprochen wird. Die Methode sowie die benötigten Parameter bilden den `data`-Teil des Requests. Die Rückgabe definiert das JSON Objekt welches vom Server als Antwort auf eine Anfrage zurückgesendet wird. Generell besteht diese Antwort mindestens aus den Teilen `msg` sowie `error`. Ist ein Fehler während der Operation aufgetreten nimmt `error` den Wert `true` an. Eine allgemeine Beschreibung der Operation befindet sich in `msg`.

Controller	Methode	Parameter	Rückgabe	Beschreibung
users.php	Übernimmt generelle Tätigkeiten die den Benutzer betreffen. Die Methode login ist Voraussetzung für das Arbeiten mit Events und Tasks.			
	operation: „register“	name: „Klaus“, password: „geheim“	msg: „...“, error: „false|true“	Registriert einen Benutzer im System.
	operation: „login“	name: „Klaus“, password: „geheim“	msg: „...“, error: „false|true“	Ein zuvor registrierter Nutzer wird am System angemeldet.
	operation: „logout“		msg: „...“, error: „false|true“	Ein angemeldeter Benutzer wird abgemeldet.
	operation: „list_events_of_user“		msg: „...“, error: „false|true“, events: [{ eid: „...“, title: „...“, description: „...“, date: „...“ }, {...}]	Es werden alle Events angezeigt denen der aktuell angemeldete Benutzer beigetreten ist.
	operation: „list_tasks_of_user“		msg: „...“, error: „false|true“, tasks: [{ eid: „...“, event: „...“, task: „...“, due: „...“ }, {...}]	Es werden alle Tasks angezeigt die der aktuell angemeldete Benutzer übernommen hat.
event.php events	Übernimmt generelle Tätigkeiten die Events betreffen. Die Anmeldung eines Benutzers ist Voraussetzung.			
	operation: „list_events“		msg: „...“, error: „false|true“,	Gibt alle im System

			events: [{ eid: „...“, title: „...“, description: „...“, date: „...“ }, {...}]	eingetragen Events zurück.
	operation: „list_users_of_event“	eid: 1	msg: „...“, error: „false|true“, users: [{ name: „...“ }, {...}]	Gibt alle Teilnehmer eines Events zurück.
	operation: „list_tasks_of_event“	eid: 1	msg: „...“, error: „false|true“, tasks: [{ tid: „...“, title: „...“, volume: „...“ }, {...}]	Gibt alle Tasks für einen Event zurück.
	operation: „create“	title: „...“, description: „...“, date: „YYYY-MM-DD“	msg: „...“, error: „false|true“, eid: „...“	Legt einen neuen Event an. Der aktuell eingetragene Benutzer ist gleichzeitig Admin und Teilnehmer dieses Events.
	operation: „join“	eid: 1	msg: „...“, error: „false|true“	Der aktuell angemeldete Benutzer tritt dem angegebenen Event bei.
tasks.php				
	operation: „create“	title: „...“, volume: „...“, eid: 1	msg: „...“, error: „false|true“	Ein Task für einen angegebenen Event wird angelegt. Der angemeldete Nutzer muss Admin des Events sein. Volume gibt dabei an, wie viele Personen einer Task zugeteilt werden können.
	operation: „assign“	tid: 1	msg: „...“, error: „false|true“	Der angemeldete Benutzer übernimmt eine Aufgabe. Dies setzt voraus, dass er Benutzer dem Event beigetreten ist und der Task noch nicht komplett belegt ist (siehe Volume).
	operation:	tid: 1	msg: „...“,	Alle Benutzer die

	„list_users_of_task“		error: „false|true“, users: [{ name: „...“ }, {...}]	eine Aufgabe übernommen haben werden angezeigt.

Hinweise zur Abgabe

Als Abgabe wird ein Archiv des Ordners Events erwartet, der das gesamte, lauffähige Projekt beinhalten muss (ohne die entsprechende Datenbank). In diesem Ordner sollte neben dem vorgegebenen Server-Teil auch der implementierte Client-Teil befindlich sein.