Beschreibung und Dokumentation der WebApp mobiSenMood

Gregor Wixler

Dokumentation der im Verlauf der Bachelorarbeit erstellten WebApp mobiSenMood. Gezeigt wird der Aufbau, Installation und die Logik der WebApp.

Inhaltsverzeichnis

1. Über mobiSenMood

Die mobile WebApp mobiSenMood wurde im Sommersemester 2011 im Rahmen einer Bachelorarbeit an der FH Würzburg-Schweinfurt erstellt. Sie basiert auf dem Framework Sencha Touch und zeigt eine mögliche Lösung für die userseitige Bedienung der Lernumgebung Moodle via Smartphone. Der Name der WebApp setzt sich zusammen aus den Namen der bei der Entwicklung hinzugezogenen „Parteien“: das mobiLAB der FH Würzburg-Schweinfurt[1], das JavaScript-Framework Sencha Touch[2] und die Lernumgebung Moodle[3].

Die in JavaScript erstellte WebApp ahmt das Aussehen und Verhalten einer nativen App nach und benötigt außer ihren serverseitigen Dateien keine Installation auf dem Zielgerät.

Da zum Zeitpunkt der Erstellung Moodle-eiene Webservices noch nicht vollständig implementiert waren, benötigt mobiSenMood für den Datenzugriff das Moodle-externe Webservicepaket MoodleWS[4].

Als HTML5-basierte WebApp funktioniert mobiSenMood am besten in Browsern auf WebKit-Basis.

2. Struktur der WebApp mobiSenMood

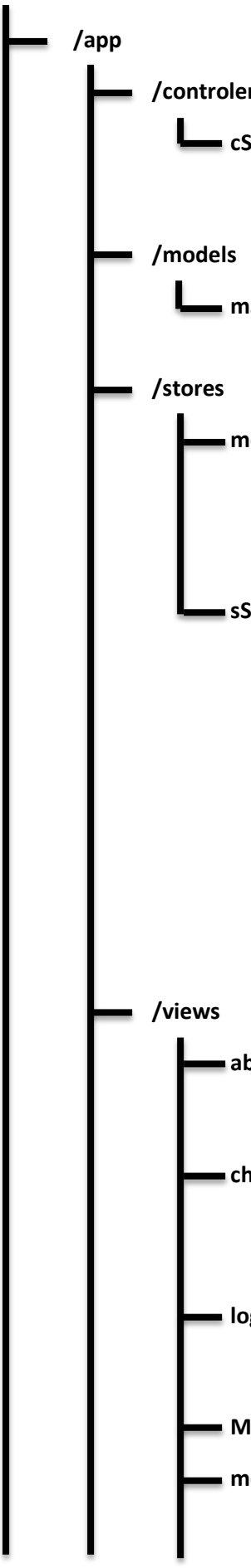
Nach dem MVC-Modell erstellt, spaltet die WebApp ihre Models, Views und Controlers in einzelne Dateien auf. Als ein zusätzliches Sencha-eigenes Element kommen die Stores hinzu. Diese sind für die Einstellung und Verwaltung der Datenzugriffe zuständig. Der gemeinsame Namensraum im Code ist „SenMood“. Der Code der App ist im Code selbst durchgehend mit Inline-Kommentaren versehen.

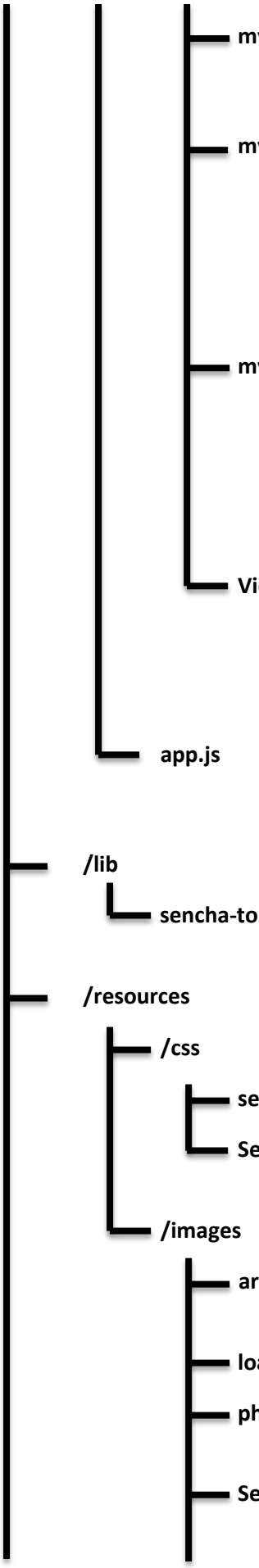
Die Ordnerstruktur der WebApp ist wie folgt:

./

index.html	Zentrale Startdatei der WebApp. Liest beim Aufruf die Scriptdateien der WebApp ein.
app.manifest	Offline-Manifest der WebApp. Listet die für den Offlinebetrieb notwendigen Dateien auf. Wird vom Browser beim Online-Aufruf der WebApp interpretiert.

[1] http://mobilab.fhws.de/
[2] http://www.sencha.com/products/touch/
[3] http://moodle.org/
[4] https://github.com/patrickpollet/moodlews

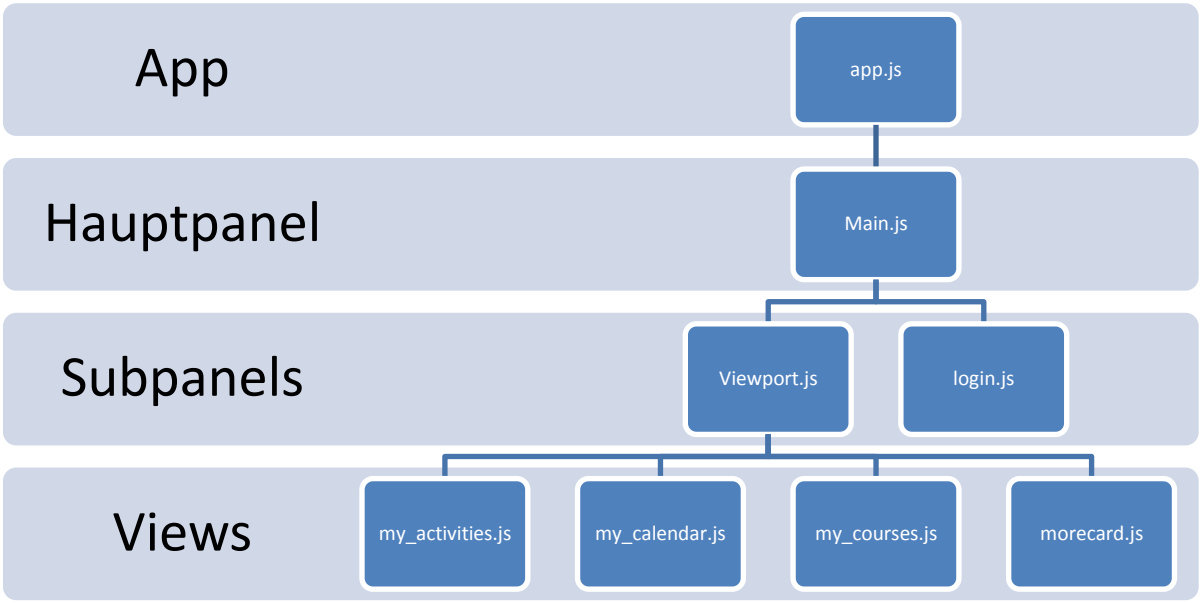

/app
/controlers
cSenMood.js
Sammlung aller zentralen Funktionsaufrufe der WebApp. Beim Aufrufen der WebApp wird hier die Funktion „goLogin“ gestartet.
/models
mSenMood.js
Datenmodelle der in der WebApp benutzen Daten
/stores
morestore.js
Definition für die Auflistung von auswählbaren Views im „Mehr“-Bereich der App. Explizit ausgelagert worden, um diesen Bereich bei Bedarf leicht erweitern zu können.
sSenMood.js
Definition der Datenstores und ihrer Zugriffsproxies.
Stores mit Moodledaten sind hier sowohl als Onlineversion (für den Zugriff auf soap_to_json.php) als auch als Offlineversion (für die lokale Speicherung der vom Server abgeholten Daten) vorhanden.
Onlinestores übergeben bei erfolgreichen Datenverbindungen ihre Daten an Offlinestores, welche wiederum von den Datendarstellenden Elementen der WebApp benutzt werden.
/views
aboutcard.js
View „Über mobiSenMood“. Zeigt das mobiSenMood-Logo und zusätzliche Informationen über die WebApp
changeusercard.js
View „Userwechsel“. Zeigt den aktuell benutzen Moodle-Usernamen an und kann via Knopfdruck das Login-Panel laden, um einen Userwechsel zu ermöglichen.
login.js
Login-Panel. Wurde abgetrennt, um sie nach Ablauf ihrer Nützlichkeit gezielt aus dem Speicher entfernen zu können.
Main.js
Hauptcontainer für die einzelnen App-Panels
morecard.js
View „Mehr“. Zeigt die in der morestore.js eingetragenen Views in einer auswählbaren Liste.



my_activties.js	View „Aktuelles“. Listet die zuletzt vorgenommenen Änderungen in den Kursen des Users auf
my_calendar.js	View „Termine“. Listet aktuell gültige Termine auf. Alle Termine, die älter als eine Stunde sind (zum Zeitpunkt der Viewbetrachtung), werden ausgeblendet. Einzelne Termine können zur näheren Betrachtung angeklickt werden.
my_courses.js	View „Aktuelles“. Listet die Kurse des Users auf. Informationen der Kursbereiche können hier durch ein Anklicken des Kurses erreicht werden. Die dabei benutze Carousel-View wird direkt im Code generiert.
Viewport.js	Zusammenfassendes Panel der App-Funktionen (Terminanzeige, Kursauflistung, etc.). Wurde abgetrennt, um diese Views gezielt und gemeinsam in den Speicher laden und sie aus demselben entfernen zu können.
app.js	Startskript der WebApp. Legt Namespace und die beim Appstart auszuführenden Aktionen fest
/lib	
sencha-touch.js	JavaScript-Bibliothek des Frameworks Sencha Touch
/resources	
/css	
sencha-touch.css	Stylesheets von Sencha Touch
SenMood.css	Zusätzliche Layoutstyles für die Listenfelder der WebApp
/images	
arrow_right.png	Pfeil-Symbol. Wird in den Listen verwendet um auf weiteren Inhalt hinzuweisen
loading.gif	Animiertes Kreissymbol für die Ladeprozesse
phone_startup.png	Ladebildschirm bei Offlineaufruf der WebApp. Funktioniert derzeit nur mit iOS.
SenMood_icon.png	mobiSenMood-Icon. Wird bei Ablage des Lesezeichens auf dem Screen angezeigt

SenMood_small.png	mobiSenMood-Logo zur Anzeige im Loginscreen
/webservice	
soap_to_json.php	Parser-Webservice in PHP. Dient der Umwandlung der SOAP-Antworten von MoodleWS in Sencha-lesbares JSON-Format

Folgende Grafik soll die Zusammenhänge der einzelnen Panels, Views und ihrer JavaScript-Dateien verdeutlichen:



Wie oben zu sehen, dient das Hauptpanel (Main.js) als Sammelbehälter für die Subpanels. Diese wiederum enthalten die einzelnen Views.

Grund für diese Trennung ist der bereits beschriebene Versuch speicherschonend zu arbeiten. Durch die explizite Gruppierung von logisch zusammengehörenden Views lassen sich die nicht mehr benötigten Views mit nur einem einzelnen Befehl aus dem Speicher entfernen (z.B. Loginpanel nach erfolgtem User-Login) oder in den selbigen neu laden (z.B. Loginpanel für einen erneuten Login mit einem anderen Usernamen).

3. Installation

3.1 Voraussetzungen

Zum funktionieren benötigt mobiSenMood eine eingerichtete Moodle-Umgebung samt Kursen und Benutzerkonten, die für diese Kurse eingetragen sind. Obwohl für Moodle Vers. 2.0+ entwickelt, funktioniert mobiSenMood aufgrund der Abwärtskompatibilität von MoodleWS auch mit Moodle-Versionen 1.7, 1.8 und 1.9.

Die URL der WebApp sollte nach Möglichkeit in derselben Domäne liegen, wie der Moodleserver, um mögliche Cross-Domain-Scripting-Fehler zu vermeiden.

3.2 Installation

Installation erfolgt in 3 Schritten:

1. **Einrichten MoodleWS**
2. **Kopieren (oder Entpacken) der WebApp-Dateien auf dem Zielserver**
3. **Anpassen der Serverzugriffs-Parameter in den WebApp-Dateien**

3.2.1 Einrichten MoodleWS

- **Einrichten MoodleWS[5] auf einem bereits laufenden Moodle-Server**

  Bei Moodle 2.0+:

  1. Die .zip-Datei mit dem Webservice in den Moodle-Rootordner entpacken. Dies erstellt den Ordner ./wspp/
  2. Ordner ./wspp/local/ in den Ordner ./local/ auf dem Moodleserver kopieren
  3. In Moodle als Administrator einloggen
  4. Den Menüpunkt „Mitteilungen“ in dem Bereich „Webseiten-Administration“ anwählen (siehe Abbildung 1 - Menüpunkt Mitteilungen) und den Installations-Anweisungen folgen

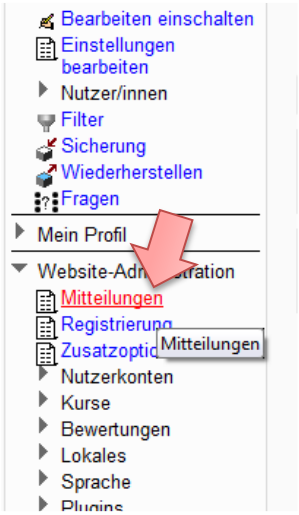


Abbildung 1 - Menüpunkt Mitteilungen

[5] https://github.com/patrickpollet/moodlews

3.2.2 Kopieren der WebApp-Dateien

- **.zip-Datei mit mobiSenMood auf dem Zielserver der WebApp entpacken.**

  Dies erstellt die Ordner ./app/, ./lib/, ./resources/ und ./webservice/ mit der oben bereits dargestellten inneren Struktur und legt die index.html und die app.manifest-Dateien in den Root-Ordner ab.

  Sollten bereits entpackte Dateien auf den Server kopiert werden, sollte die bisher beschriebene Ordnerstruktur zwecks Übersichtlichkeit weiterhin beibehalten werden.

  Zwar ist die WebApp so erstellt, dass sie aus dem Root-Ordner laufen soll – deswegen die index.html im Root - sie kann aber in einem beliebigen Ordner liegen und von einer beliebig benannten .html-Datei gestartet werden. Relevant sind hierbei jedoch die Serverzugriffsparameter, die bei einer veränderten Ordnerstruktur zusätzlich angepasst werden müssen (siehe 3.4 Anpassen von mobiSenMood an eine alternative Ordnerstruktur).

3.3.3 Anpassen der Serverzugriffs-Parameter

Damit die WebApp in der aktuellen Serverumgebung korrekt funktioniert, müssen folgende Zugriffsparameter im Code angepasst werden:

- **PHP-Parser-URL für den Store-Zugriff**

  Ordner ./app/stores/, Datei „sSenMood.js“, Zeile 3

  ```
  var proxy_url='http://YourWebAppDomain/webservice/soap_to_json.php';
  ```

  Diese Zeile wird von allen Oninestores benutzt und muss auf die tatsächliche URL der soap_to_json.php verweisen (z.B. http:/www.myMoodle.com/webservice/ soap_to_json.php). Die Adressenangabe ist bewusst als feste URL gelassen worden, da die PHP-Datei bei Bedarf (z.B. zur Vereinfachung der Zugriffsrechteadministration) überall innerhalb der selben Domäne abgelegt werden kann.

- **Webservice URL für den SOAP-Zugriff**

  Ordner ./webservice/, Datei „soap_to_json.php“, Zeile 5

  ```
  $SOAPclient = new
  SOAPClient('http://YourMoodleServer/wspp/wsdl_pp.php');
  ```

  Um die Daten korrekt abholen zu können, benötigt der PHP-Parser die Eingabe der MoodleWS-URL. Eingetragen wird diese direkt in der Zeile zur SOAP-Client-Definition.

  Diese URL verweist direkt auf die auf dem Moodle-Server liegende wdsl_pp.php-Datei.

3.4 Anpassen von mobiSenMood an eine alternative Ordnerstruktur

Sowohl die oben beschriebene Ordenrstruktur, als auch die Dateinamen sind keine zwingende Voraussetzung für die Funktionsfähigkeit der WebApp.
Soll die WebApp in einer anderen, als der beschriebenen Ordnerstruktur eingesetzt werden, müssen zuvor die Verweise auf die WebApp-Dateien in der index.html-Datei an die neuen Ordnernamen angepasst werden.
Die Reihfolge der zu ladenden JavaScript-Dateien sollte dabei jedoch nicht verändert werden, um bestehende Objektabhängigkeiten nicht zu zerstören.

Der Name der Start-HTML (index.html) ist variabel. Dieser kann ohne Behinderungen beliebig verändert werden.

Rechtliche Hinweise