Web2 Projekt

Diese Projektarbeit dient zur praktischen Vertiefung des Unterrichtsstoffes von *Web Technologien 2*. Neben der Projektarbeit existiert eine Serie von Laboraufgaben - die Laboraufgaben behandeln jeweils einzelne Aspekte zur Umsetzung des Projektes. Das Projekt kann in Gruppen von bis zu drei Mitgliedern bearbeitet werden. Das Projekt wird mit 20% Gewichtung zur Semesternote angerechnet.

Projektziel

Erstellen Sie eine Pendenzenlisten Applikation[1] mit:

- Responsive Layout
- 2-Tier Architektur
- RESTful Schnittstelle
- JSON Datenstrukturen
- Serverseitiger Persistenz

Meilensteine & Abgaben

Um die Komplexität zu reduzieren wird die Pendenzenlisten Applikation in drei Iterationen erstellt. Diese Meilensteine werden einzeln abgegeben und bewertet.

1. Pendenzenliste als Javascript Objekte
2. Dynamische HTML Seite
3. Pendenzenliste über Web-Schnittstelle lesen und persistieren

Alle Abgaben werden nach den folgenden Kriterien bewertet:

Validiert Valider HTML, CSS und Javascript Code *(1 Punkt)*

Deployed Zugreifbar auf dublin.zhaw.ch Lab-Server[2] *(1 Punkt)*

Getested Code ist unter Test *(1 Punkt)*

Funktional Code erfüllt funktionale Anforderungen des jeweiligen Meilensteines *(2 Punkte)*

---

[1] In Absprache mit den Dozenten kann auch an einer anderen Applikation gearbeitet werden. Wichtig ist, dass die Projektziele und Meilenstein Termine eingehalten werden. Für Studierende mit wenig Vorwissen wird jedoch davon abgeraten; die Pendenzenlisten Applikation ist durch die ergänzenden Übungen sehr gut dokumentiert.

[2] Für alle Übungen wird die Infrastruktur der ZHAW empfohlen. Natürlich können die Übungen auch auf andere Server deployed werden, solange die Seiten jederzeit zugänglich sind.

1. Meilenstein: Pendenzenliste als Javascript Objekte

Ziel des ersten Meilensteines ist eine simple Applikation zu erstellen, welche das Objektmodell der Pendenzenlisten Applikation abbildet. Die Applikation soll zwei Objekttypen unterstützen:

- Pendenzenlisten sollen über ein `TaskList` Objekt abgebildet werden
- Einzelne Pendenzen in den Listen sollen über `Task` Objekte abgebildet werden

Durch das Verwenden von `TaskList` Objekten kann die Applikation mehrere verschiedene Pendenzenlisten unterstützen. Sowohl `TaskList` als auch `Task` sollen ein Feld `title` besitzen. Die `TaskList` Objekte halten Referenzen auf ihre `Task`-Objekte in Form eines Arrays. Die `Task` Objekte enthalten zusätzlich ein Feld `done` um festzuhalten, ob sie bereits bearbeitet wurden.

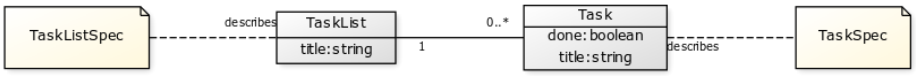

Figure 1: Objektdiagramm

Wenn die Programmierung mit Javascript für Sie neu ist, erstellen Sie die Objekte mittels einer Konstruktor-Funktion und nicht mit dem Schlüsselwort `class`.

Erstellen Sie die Applikation möglichst nach der Test-Driven-Development Methodik; versuchen Sie sich zuerst zu überlegen, welche Funktionen gebraucht werden. Implementieren Sie dann entsprechende Tests und zugehörigen Applikationscode Schritt für Schritt. Alle Tests sollen im Browser über eine Seite `SpecRunner.html` aufgerufen werden können.

Die App soll eine leere `index.html` Seite enthalten welche alle Javascript Dateien referenziert. Mittels Javascript Console können dann bereits Objekte angelegt und die Benutzerinteraktion simuliert werden.

Hilfreiche Labor Aufgaben

Ergänzend zur Bearbeitung des Meilensteines wird die Bearbeitung der folgenden Laborübungen empfohlen - sie enthalten detaillierte Anleitung zu Teilproblemen.

Labor	Übung
01	JavaScript mittels HTML aufrufen
02	Javascript Objekte
02	Javascript Funktion zum Constructor hinzufügen
03	Jasmine benutzen

Zusammenfassung erwartete Resultate

- Dateien `Task.js` und `TaskList.js` mit Objekt-Konstruktoren
- Dateien `TaskSpec.js`, `TaskListSpec.js` und `SpecRunner.html` mit Testcode
- Datei `index.html` zum 'ausporbieren' der Objekte in der Javascript Console

---

2. Meilenstein: Dynamische HTML Seite

Ziel dieses Meilensteines ist es zum Datenmodell eine grafische Benutzerschnittstelle zu programmieren. Der dazugehörige HTML Code soll nicht statisch hinterlegt werden sondern entsprechend den Daten generiert werden.

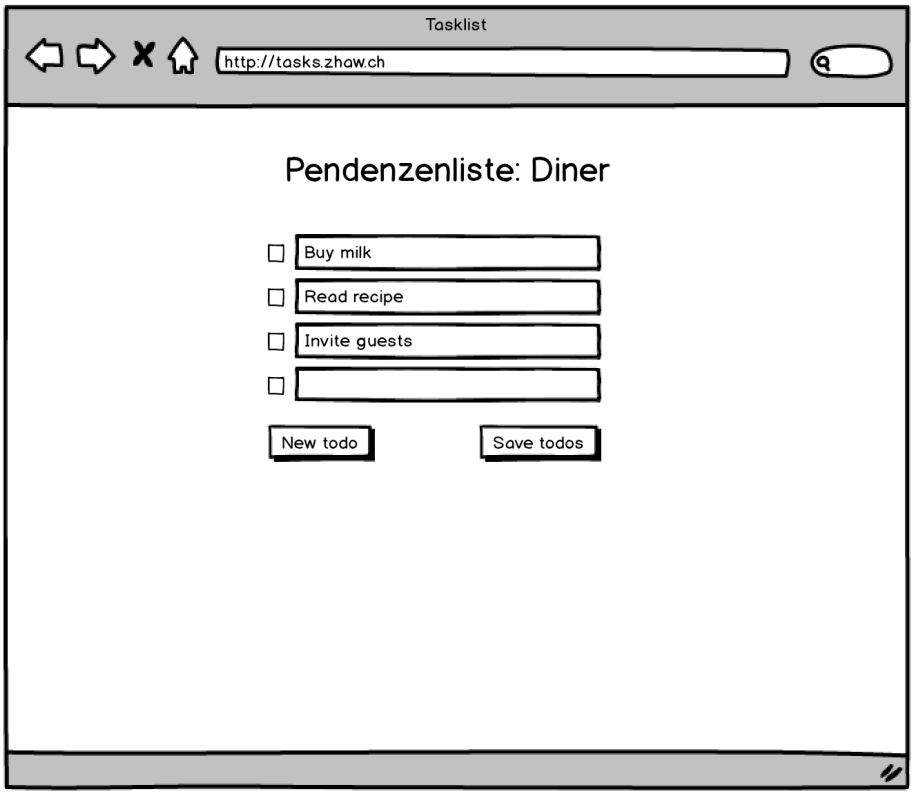

Figure 2: Mockup

Die einzelnen Arbeitsschritte sind (ebenfalls in den Laborübungen abgedeckt):

1. Erstellen eines Web-Designs für die Seite
2. Programmatisches Erstellen von Testdaten (Pendenzenliste mit Pendenzen) zur Demonstration
3. Programmatische Umwandlung der Daten in der Pendenzenliste in HTML Code (rendering)
4. Bei Änderung der Pendenzen durch die Benutzer (z.B. Klick auf Checkbox) programmatische Anpassung der verknüpften Daten

Für das Design der Seite wird Bootstrap empfohlen. Für das Rendering der HTML Elemente und das Event-Handling (Aktualisierung Daten) wird jQuery empfohlen. Die Funktionen von jQuery können gut mit Jasmine Specs getestet werden.

Hilfreiche Labor Aufgaben

Ergänzend zur Bearbeitung des Meilensteines wird die Bearbeitung der folgenden Laborübungen empfohlen - sie enthalten detaillierte Anleitung zu Teilproblemen

Labor	Übung
05	jQuery Funktionen unter Test bringen
06	Klassen und Attribute hinzufügen und entfernen
06	Animationen einbinden
07	Events verarbeiten

Zusammenfassung erwartete Resultate

- Grafische Benutzerschnittstelle zum Laden und Manipulieren der Pendenzenlisten
- Dateien `Task.js` und `TaskList.js` mit Funktionalität, insb. Rendering der Tasks
- Dateien `TaskSpec.js`, `TaskListSpec.js` und `SpecRunner.html` mit Testcode
- Datei `Demo.js` o.ä. zum Laden von Demo-Daten
- Datei `index.html` zum Laden der Benutzerschnittstelle

---

3. Meilenstein: Pendenzenliste über Web-Schnittstelle lesen und persistieren

Die Pendenzenlisten sollen von einem Webserver geladen werden und von einem Server gespeichert werden. Die Server Komponente der Applikation wird zur Verfügung gestellt - in dieser Übung geht es darum diese über eine REST-Schnittstelle anzubinden.

Der Server zhaw.herokuapp.com stellt einen einfachen Dienst zum Laden von Pendenzenlisten zur Verfügung. Von der URL http://zhaw.herokuapp.com/task_lists/`:id` kann eine Pendenzenliste im JSON Format geladen werden. `:id` ist dabei zu ersetzen mit der Identifizierung einer konkreten Pendenzenliste. Beispiel: Die Pendenzenliste mit der Identifizierung `demo` kann so geladen werden:

http://zhaw.herokuapp.com/task_lists/demo (Achtung bei copy & paste, task_lists ist mit einem underscore geschrieben)

Die Antwort des Servers geschieht im JSON Format:

```
{"id":"demo", "tasks":
 [{"title":"Buy milk","done":false},
{"title":"Invite friends","done":false},
{"title":"Call Bill","done":false},
{"title":"Write Task List","done":true}]}
```

Mittels POST-Request können neue Pendenzenlisten geschrieben werden oder bestehende Listen aktualisiert werden. Das JSON Format zum Schreiben der Listen ist dasselbe wie zum Lesen der Listen. Wird eine neue Liste angelegt, vergibt der Server in seiner Antwort automatisch eine eindeutige ID (Feld `id`). Diese kann danach zur Aktualisierung der Liste verwendet werden kann. Die URL um neue Listen anzulegen ist:

http://zhaw.herokuapp.com/task_lists/ (Slash `/` am Schluss wichtig)

Die URL um bestehende Listen anzupassen ist

http://zhaw.herokuapp.com/task_lists/`:id` (wobei `:id` ersetzt werden muss durch die konkrete Pendenzenlisten ID).

Bitte beachten Sie, dass in ihrem Programm die Erstellung und die Aktualisierung der Pendenzenlisten leicht unterschiedlich behandelt werden müssen. Im Erstellungsfall ist noch keine `:id` bekannt, im Aktualisierungsfall ist die `:id` Bestandteil der URL.

Die einzelnen Arbeitsschritte sind (ebenfalls in den Laborübungen abgedeckt):

- Lesen und Anzeigen von bestehenden Pendenzenlisten (z.B. `demo`) über die Schnittstelle. Dies beinhaltet die Umwandlung der Serverantwort im JSON Format in Javascript Objekte der Typen `TaskList` und `Task`.

- Anlegen von neuen Pendenzenlisten über einen POST Request. Dies beinhaltet die Umwandlung der Javascript Objekte ins JSON Format
- Abspeichern bestehender Pendenzenlisten auf dem Server (update).
- Hinterlegung der Pendenzenliste ID (falls bekannt) in der URL, z.B. als Hash-Parameter. Bei Aufruf dieser URL (Bookmark, Reload) Abruf der entsprechenden Daten vom Server.

Es wird empfohlen, dass Sie die jQuery Methoden zum Senden von AJAX Requests benutzen (`$.getJSON`, `$.post`). Die Testsuite Ihrer Applikation soll unabhängig von der Schnittstelle funktionieren (offline). Benutzen Sie dazu das Mock-Pattern mit Jasmine Spies.

Hilfreiche Labor Aufgaben

Ergänzend zur Bearbeitung des Meilensteines wird die Bearbeitung der folgenden Laborübungen empfohlen - sie enthalten detaillierte Anleitung zu Teilproblemen

Labor	Übung
04	Jasmine-Spies verwenden
08	GET Requests mit curl
08	POST Requests mit curl
09	JSON abrufen und verwerten
10	TaskList JSON Repräsentation
10	Pendenzenlisten über REST persistieren
10	Pendenzenlisten Bookmark

Zusammenfassung erwartete Resultate

- Grafische Benutzerschnittstelle zum Laden und Manipulieren der Pendenzenlisten über die REST Schnittstelle
- Dateien `Task.js` und `TaskList.js` mit Funktionalität, insb. Rendering der Tasks
- Dateien `TaskSpec.js`, `TaskListSpec.js` und `SpecRunner.html` mit Testcode
- Datei `index.html` zum Laden der Benutzerschnittstelle