Webanwendung "Seminarverwaltung"

Beschreibung der Anwendung

Ein Betrieb möchte die Teilnahme seiner Mitarbeiter an Seminaren und Fortbildungsveranstaltungen besser organisieren. Dazu sollen Sie eine passende Web-Anwendung - "Seminarverwaltung (SV)" genannt - erstellen.

Das Datenmodell der Web-Anwendung sieht vereinfacht so aus:

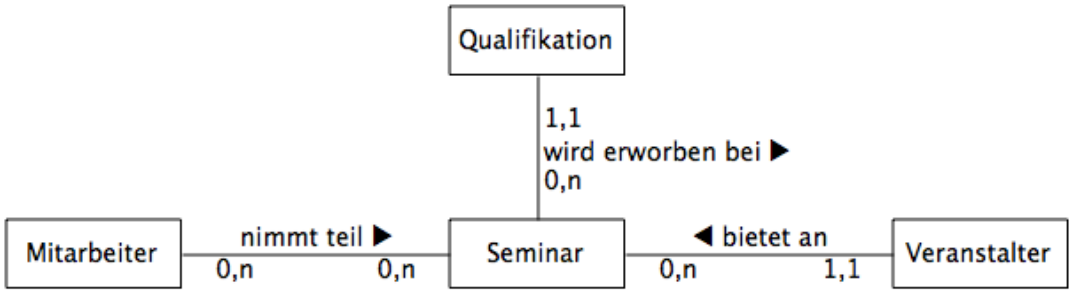

Abbildung: Datenmodell SV

Es ist folgendermaßen zu verstehen:

- ein Mitarbeiter kann an keinem, einem oder vielen Seminaren teilnehmen
- ein Seminar kann von keinem, einem oder vielen Mitarbeitern besucht werden
- einem Seminar ist genau eine Qualifikation zugeordnet, die durch einen teilnehmenden Mitarbeiter erworben werden kann
- eine Qualifikation kann keinem, einem oder mehreren Seminaren zugeordnet werden
- ein Seminar wird durch genau einen Veranstalter angeboten
- ein Veranstalter kann kein, ein oder viele Seminare anbieten.

Berücksichtigen Sie folgenden Funktionsumfang:

- **Pflege der Daten der Mitarbeiter**
  - Daten der Mitarbeiter anlegen, bearbeiten und löschen
  - zu berücksichtigende Daten sind:
    - Name
    - Vorname
    - Funktion
- **Pflege der Daten der Seminare**
  - Daten der Seminare anlegen, bearbeiten und löschen
  - zu berücksichtigende Daten sind:
    - Bezeichnung
    - Zeitraum (von / bis)
    - Ort
  - Qualifikation zuordnen
  - Veranstalter zuordnen
- **Pflege der Daten der Qualifikationen**
  - Daten der Qualifikation anlegen, bearbeiten und löschen
    - eine Qualifikation kann nur gelöscht werden, wenn sie keinem Seminar mehr zugeordnet ist
  - zu berücksichtigende Daten sind:

        - Bezeichnung
        - Beschreibung (längerer Text)
        - ausstellende Organisation
- **Pflege der Daten der Veranstalter**
    - Daten der Veranstalter anlegen, bearbeiten und löschen
        - die Daten eines Veranstalters können nur gelöscht werden, wenn sie keinem Seminar mehr zugeordnet sind
    - zu berücksichtigende Daten sind:
        - Name
        - Gesellschaftsform
        - Sitz (Ort)
- **Teilnahme der Mitarbeiter verwalten**
    - Mitarbeiter anmelden:
        - Auswahl des Mitarbeiters
        - Auswahl des Seminars
            - der Mitarbeiter darf noch nicht als Teilnehmer eingetragen sein
            - das Seminar darf noch nicht begonnen haben
        - anmelden
    - Mitarbeiter abmelden:
        - Auswahl des Mitarbeiters
        - Auswahl des Seminars
            - nur Seminare, für die der Mitarbeiter angemeldet ist und die noch nicht begonnen haben
        - abmelden
- **Auswertungen**
    - Teilnehmerverzeichnis
        - alphabetisch sortiert nach Namen der Mitarbeiter
        - bei einem Mitarbeiter die Seminare, an denen er teilgenommen hat (Sortierung nach Bezeichnung der Seminare)
    - Seminarverzeichnis
        - alphabetisch sortiert nach Bezeichnung der Seminare
        - mit Angabe der Daten des Veranstalters
    - Qualifikationsverzeichnis
        - alphabetisch sortiert nach Bezeichnung der Qualifikation
        - bei einer Qualifikation die Seminare, die zugeordnet sind, sortiert nach Bezeichnung des Seminars
        - bei einem Seminar die Mitarbeiter, die teilgenommen haben, sortiert nach den Namen der Mitarbeiter.

Sehen Sie folgende Sichten der Web-Anwendung vor:

- **Startseite** mit Verzweigung zu:
    - Pflege der Daten der Mitarbeiter
    - Pflege der Daten der Seminare
    - Pflege der Daten der Qualifikationen
    - Pflege der Daten der Veranstalter
    - Anmeldung eines Mitarbeiters
    - Abmeldung eines Mitarbeiters
    - Teilnehmerverzeichnis
    - Seminarverzeichnis
    - Qualifikationsverzeichnis
- für die Pflege der **Daten der Mitarbeiter**
    - **Auswahlliste**
        - Auswahl einzelner Einträge zur weiteren Bearbeitung
        - Auswahl einzelner Einträge zum Löschen
            - dann müssen alle davon abhängigen Daten gelöscht werden

        - Übergang zum Erfassen neuer Daten eines Mitarbeiters
    - **Detailformular**
        - zum Erfassen neuer Daten eines Mitarbeiters
        - zum Bearbeiten der Daten eines Mitarbeiters
- für die Pflege der **Daten der Seminare**
    - **Auswahlliste**
        - Auswahl einzelner Einträge zur weiteren Bearbeitung
        - Auswahl einzelner Einträge zum Löschen
            - dann müssen alle davon abhängigen Daten gelöscht werden
        - Übergang zum Erfassen neuer Daten eines Seminars
    - **Detailformular**
        - zum Erfassen neuer Daten eines Seminars
        - zum Bearbeiten der Daten eines Seminars
- für die Pflege der **Daten der Qualifikation**
    - **Auswahlliste**
        - Auswahl einzelner Einträge zur weiteren Bearbeitung
        - Auswahl einzelner Einträge zum Löschen
            - dann müssen alle davon abhängigen Daten gelöscht werden
        - Übergang zum Erfassen neuer Daten einer Qualifikation
    - **Detailformular**
        - zum Erfassen neuer Daten einer Qualifikation
        - zum Bearbeiten der Daten einer Qualifikation
- für die Pflege der **Daten der Veranstalter**
    - **Auswahlliste**
        - Auswahl einzelner Einträge zur weiteren Bearbeitung
        - Auswahl einzelner Einträge zum Löschen
            - dann müssen alle davon abhängigen Daten gelöscht werden
        - Übergang zum Erfassen neuer Daten eines Veranstalters
    - **Detailformular**
        - zum Erfassen neuer Daten eines Veranstalters
        - zum Bearbeiten der Daten eines Veranstalters
- für die **Anmeldung eines Mitarbeiters**
    - Auswahl Mitarbeiter
    - Auswahl Seminar
        - nur Seminare, die noch nicht begonnen haben
        - nur Seminare, für die der Mitarbeiter noch nicht angemeldet ist
    - anmelden
- für die **Abmeldung eines Mitarbeiters**
    - Auswahl Mitarbeiter
    - Auswahl Seminar
        - nur Seminare, die noch nicht begonnen haben
        - nur Seminare, für die der Mitarbeiter angemeldet ist
    - abmelden
- für jede **Auswertungen**:
    - Teilnehmerverzeichnis
    - Seminarverzeichnis
    - Qualifikationsverzeichnis.

Beim Löschen von Daten muss grundsätzlich eine Bestätigung erfolgen und auf die Integrität des Datenbestands geachtet werden.

Anforderungen an die Umsetzung - Schritt 1 (Termin P2)

Die Web-Anwendung "Seminarverwaltung (SV)" wird als *Single-Page-Application* implementiert. Die eigentliche HTML-Seite wird einmal geladen, weitere Inhalte werden dynamisch im Hintergrund geladen (*Ajax-Ansatz*). Die Implementierung erfolgt mit Hilfe des `XMLHttpRequest()`-Objekts. Weitere Hinweise finden Sie im Anhang.

Die einzelnen Inhalte - Listen (Tabellen), Formulare tec. - werden durch den Web-Server mit Hilfe einer Template-Engine erzeugt, ausgeliefert und auf dem Client dynamisch in die bereits geladene HTML-Seite eingetragen.

Die Einträge in den Listen werden **nicht** als Verweise (Links) ausgeführt. Vielmehr erfolgt die Auswahl erfolgt durch Klick auf den Eintrag, die Bearbeitung und Auswertung des Ereignisses erfolgt clientseitig mit Javascript. Zum Wechsel in das Bearbeitungsformular muss ein zusätzlicher Verweis (Link) oder Schalter vorgesehen werden, der die Auswahl des Eintrags berücksichtigt.

Die Auswertung und Übertragung der Formulardaten erfolgt mit dem `FormData` - Object und der oben genannten Hintergrundübertragung (Ajax). Weitere Hinweise finden Sie im Anhang.

Die clientseitige Implementierung erfolgt mit Hilfe der Standard-DOM-Methoden und -eigenschaften. Die Verwendung von zusätzlichen Javascript-Bibliotheken ist in diesem Schritt nicht zugelassen.

Anforderungen an die Umsetzung - Schritt 2 (Termin P3)

In Schritt 2 stellt der Server keine HTML-Abschnitte mehr zur Verfügung, sondern liefert die Inhalte als Daten im JSON-Format aus. Die Aufbereitung der Daten zu HTML-Markup erfolgt *clientseitig* mit der Template-Engine `te/tm` (siehe Hinweis unten).

Die clientseitige Implementierung nutzt jetzt die Javascript-Bibliothek *jquery*. Auf *jquery* aufbauende Erweiterungen (Plugins, Skripte, Bibliotheken etc.) dürfen **nicht** verwendet werden.

Weitere Anforderungen an die Umsetzung

- Webclient:
  - Verwendung HTML5 (XML-konforme Notation)
  - überprüfung des Markup mit Hilfe der w3c-Validator-Dienste (siehe Anhang)
    - überprüfen Sie 2 Listen- und 2 Detailsichten
  - Präsentation mit CSS, ausgelagert in eine externe CSS-Datei
  - Verwendung der Template-Engine "te/tm" (siehe Anhang) (für Schritt 2).
- Webserver:
  - Verwendung Python (Version 3)
  - Verwendung Framework "cherrypy"
  - Verwendung der Template-Engine "mako" (siehe Anhang) (für Schritt 1).

Verwenden Sie folgende Verzeichnisstruktur (sv1 - Schritt 1, sv2 - Schritt 2) und erstellen Sie die angegebenen Dateien:

```
web
   /p2
       /sv1                 <--- server.py
           /app             <--- __init__.py, application.py, database.py, view.py
           /static          <--- html-Datei(en), css-Datei(en), js-Datei(en)
           /data            <--- Daten in JSON-formatierten Dateien
           /doc             <--- Dateien der Dokumentation (sv1.md + weitere)
           /templates       <--- Vorlagen für mako
```

```
/sv2                <--- server.py
    /app            <--- __init__.py, application.py, database.py, view.py
    /static         <--- html-Datei(en), css-Datei(en), js-Datei(en)
    /data           <--- Daten in JSON-formatierten Dateien
    /doc            <--- Dateien der Dokumentation (sv2.md + weitere)
    /templates      <--- Vorlagen für clientseitige Templates
```

Anforderungen an die Dokumentation

Erstellen Sie eine Dokumentation, die Ihre Lösung (Schritt 1 bzw. Schritt 2) beschreibt. Legen Sie dazu in einem Unterverzeichnis `doc` die Dateien `sv1.md` (für Schritt 1) bzw. `sv2.md` (für Schritt 2) an.

Sehen Sie folgende Gliederung für den Schritt 1 vor:

- einleitend: Ihre Gruppenzugehörigkeit, Aufbau Ihres Team, Gültigkeitsdatum der Dokumentation an
- allgemeine Beschreibung Ihrer Lösung
  - Aufgabe der Anwendung
  - Übersicht der fachlichen Funktionen
- Beschreibung der Komponenten des Servers
  - für jede Komponente:
    - Zweck
    - Aufbau (Bestandteile der Komponente)
    - Zusammenwirken mit anderen Komponenten
    - API (Programmierschnittstellen), die die Leistungen der Komponente anbieten
- Datenablage
- Konfiguration
- Durchführung und Ergebnis der geforderten Prüfungen.

Sehen Sie folgende Gliederung für den Schritt 2 vor:

- einleitend: Ihre Gruppenzugehörigkeit, Aufbau Ihres Team, Gültigkeitsdatum der Dokumentation an
- allgemeine Beschreibung Ihrer Lösung
  - Aufgabe der Anwendung
  - Übersicht der fachlichen Funktionen
- Beschreibung des Aufbaus und der Komponenten des Clients
- Beschreibung der Komponenten des Servers
  - für jede Komponente:
    - Zweck
    - Aufbau (Bestandteile der Komponente)
    - Zusammenwirken mit anderen Komponenten
    - API (Programmierschnittstellen), die die Leistungen der Komponente anbieten
- Datenablage
- Konfiguration
- Durchführung und Ergebnis der geforderten Prüfungen.

Die Dokumentation wird als utf-8 kodierter Text mit der einfachen Auszeichnungssprache `Markdown` erstellt. Mit Hilfe des Werkzeugs `pandoc` (siehe https://pandoc.org) kann eine Umsetzung in eine HTML-Datei erfolgen:

```
pandoc -f markdown -t html5 -s <IhreDatei> -o <IhreHTML5Datei>
```

Die in `pandoc` verfügbaren Erweiterungen der Auszeichnungssprache `Markdown` sollen genutzt werden.

Bewertung / Testat

Zur Bewertung Ihrer Lösung im Hinblick auf die mögliche Erteilung des Testats müssen Sie vorlegen und erläutern:

- den von Ihnen erstellten Quellcode Ihrer Web-Anwendung in den Varianten für Schritt 1 und Schritt 2
- die von Ihnen erstellten Dokumentationen.

Sie müssen die Lauffähigkeit Ihrer Lösungen und die Durchführung der Validierungen nachweisen.

Anhang

Nutzung der W3C-Validatordienste

Mit den W3C-Validatordiensten können Sie überprüfen, ob das von Ihnen verwendete Markup korrekt ist und Sie gültige CSS-Anweisungen verwendet haben.

Sie erreichen die Validatordienste so:

- **w3c-Validator-Dienst (Markup):** http://validator.w3.org/
  - Überprüfung der Korrektheit des Markup
  - Zeigen Sie den Quelltext der zu prüfenden Webseite an (z.B. Kontextmenü Webseite, dort etwa "Seitenquelltext" auswählen)
    - im Falle nachgeladener Inhalte müssen Sie ggf. auf "generierten Inhalt" zugreifen
  - Den angezeigten Quelltext markieren und in die Zwischenablage kopieren
  - Inhalt der Zwischenablage in der Registerkarte "Direct Input" einfügen und Überprüfung starten
- **w3c-Validator-Dienste (CSS):** http://jigsaw.w3.org/css-validator/
  - Überprüfung von CSS-Stilregeln
  - weitere Vorgehensweise wie vor

Javascript-Bibliothek jquery

Die Javascript-Bibliothek *jquery* vereinfacht den Zugriff auf das DOM und bietet eine erweiterte API an. Damit werden auch Unterschiede in der Javascript-Implementierung der einzelnen Web-Browser ausgeglichen und komplexere Abläufe, z.B. beim Ajax-Ansatz, vereinfacht.

Sie finden den Quellcode und die Dokumentation unter https://jquery.com und zwar unter:

- Quellcode: https://jquery.com/download/
- API-Dokumentation: https://api.jquery.com/

Template-Engine te/tm für clientseitige Templates

Es handelt sich um eine einfache, von mir (HDB) entwickelte Template-Engine, die Templates in Javascript-Funktionen umsetzt. Die Templates werden mit Hilfe des Template-Managers bei Start des Clients vom Webserver ausgeliefert. Sie werden bei der ersten Verwendung kompiliert (= Erzeugen der Javascript-Funktionen, die clientseitig vorgehalten werden). Weitere Verwendungen benutzen dann direkt diese Javascript-Funktionen.

Weitere Informationen und den Quellcode erhalten Sie rechtzeitig.

Mako-Template-Engine

Mit der Mako-Template-Engine können Sie insbesondere HTML-Seiten und -abschnitte einfach erzeugen. Mako ist in Python geschrieben, verfügt über eine Programmierschnittstelle in Python und erzeugt (intern) Python-Code zur Ausführung des übersetzten Templates.

Quelle und Installation

Sie finden die `mako template library` unter http://www.makotemplates.org/, dort insbesondere die zwar ausführliche, für den Anfänger aber manchmal etwas unübersichtliche Dokumentation.

Der Download-Bereich verweist auf den *Python Package Index (pypi)*: https://pypi.python.org/pypi/Mako/?. Verwenden Sie das angebotene Archiv, entpacken Sie es in ein temporäres Verzeichnis und installieren Sie es mit dem Befehl `python setup.py install` (falls Sie zwei Python-Versionen haben, geben Sie `python3` an).

Verwendung

Sie erstellen die Template-Dateien wiederum als UTF-8 kodierte Texte mit einem Texteditor (Ablage im Verzeichnis `template`).

Beispiel eines `mako`-Templates (Auszug aus `liste.tpl`):

```
## coding: utf-8
            <table id="idList">
               <tr><th>Name</th><th>Typ</th></tr>

               ## man verwendet hier Zugriff auf das Dictionary "data_o"

               % for key_s in data_o:
               <tr id="r${key_s}"><td>${data_o[key_s]['name']}</td><td>${data_o[key_s]['typ']}</td></tr>
               % endfor
            </table>
## Mako-Kommentare verwenden zwei #-Zeichen
```

Die Kontrollflussanweisungen werden durch ein Prozentzeichen gekennzeichnet und sind ähnlich den Anweisungen in Python aufgebaut. Platzhalter, die durch den Wert des jeweiligen Ausdrucks ersetzt werden, beginnen mit einem Dollar-Zeichen. Der eigentliche Ausdruck wird durch geschweifte Klammern begrenzt.

Auf Einrückung des Quellcodes im Template muss nicht ausdrücklich geachtet werden, sehr wohl aber auf eine übersichtliche Schreibweise!

Das Modul `view.py` sieht dann etwa so aus (Ausschnitt):

```
# coding: utf-8
import os.path
from mako.template import Template
from mako.lookup import TemplateLookup

#-------------------------------------------------------
class View_cl(object):
#-------------------------------------------------------

   #-------------------------------------------------------
   def __init__(self, path_spl):
   #-------------------------------------------------------
      # Pfad hier zur Vereinfachung fest vorgeben
      self.path_s = os.path.join(path_spl, "template")
      self.lookup_o = TemplateLookup(directories=[self.path_s])

   # ... weitere Methoden

   #-------------------------------------------------------
   def create_p(self, template_spl, data_opl):
   #-------------------------------------------------------
      # Auswertung mit templates
      template_o = self.lookup_o.get_template(template_spl)
      return template_o.render(data_o = data_opl) # hier wird da Template ausgeführt für die übergebenen Daten

   #-------------------------------------------------------
   def createList_px(self, data_opl):
   #-------------------------------------------------------
      return self.create_p('liste.tpl', data_opl)
```

In der Variable `data_opl` wird hier im Beispiel ein Dictionary übergeben, das dann in der Templatedatei zur Erzeugung des Markups einer Tabelle verwendet wird.

Ajax: Datenaustausch Client-Server im Hintergrund

Nutzung XMLHttpRequest

Die Implementierung der Datenübertragung vom Client zum Server erfolgt in Schritt 1 mit Hilfe des `XMLHttpRequest`-Objekts. Eine Beschreibung finden Sie unter:

- https://developer.mozilla.org/de/docs/Web/API/XMLHTTPRequest und
- https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest/Using_XMLHttpRequest

Beachten Sie, dass alle Anfragen an den Server *asynchron* erfolgen! D.h., Sie können nicht direkt nach dem Senden bei der Ausführung der nächsten Anweisungen mit dem Vorliegen der Ergebnisse rechnen!

Beispiel

Die Anforderung eines neuen Inhalts, z.B. einer neuen Liste, könnte etwa so aussehen:

```
var request_o = new XMLHttpRequest();
request_o.open("GET", "/mitarbeiter");
// Rückmeldungen als Ereignis verarbeiten
request_o.onload = function() {
   if (request_o.status === 200) { // 200 = ok
      // hier nur zur Verdeutlichung
      alert(request_o.responseText); // enthält z.B. den HTML-Ausschnitt mit der Liste etc.
      // wird per Eigenschaft innerHTML dem Container-Element zugewiesen
   } else {
      alert('Request failed.  Returned status of ' + request_o.status);
   }
};
request_o.send();
```

Verarbeitung von Formulardaten

Die Verarbeitung von Formulardaten insbesondere im Zusammenhang mit dem Versand per `XMLHttpRequest`-Objekts wird durch die Verwendung des `FormData`-Objekts vereinfacht. Eine Beschreibung finden Sie unter:

- https://developer.mozilla.org/de/docs/Web/API/FormData
- https://developer.mozilla.org/en-US/docs/Web/API/FormData/Using_FormData_Objects (1)

Durch den Bezug auf ein Formular, z.B. durch eine Id eindeutig gekennzeichnet, können die Formulardaten gesammelt und leicht übertragen werden.

Ein gegenüber der oben genannten Quelle (1) abgewandeltes Beispiel sieht etwa so aus:

```
// auf Formular per id zugreifen
var formElement_o = document.getElementById("f123");
// Daten im Formular sammeln im FormData-Objekt
var formData_o = new FormData(formElement_o);
// jetzt im Hintergrund an Server übertragen
var request_o = new XMLHttpRequest();
request_o.open("POST", "/mitarbeiter/save");
request_o.send(formData_o);
```

Das Beispiel enthält keine Angaben zur Reaktion auf die Rückmeldungen durch das `XMLHttpRequest`-Objekt!