Webanwendung "Mitarbeiterqualifizierung (MiQu)"

Beschreibung der Anwendung

Eine Firma möchte die Qualifizierung ihrer Mitarbeiter im Rahmen von Weiterbildungsmaßnahmen verbessern. Mit einer Webanwendung sollen einerseits die Weiterbildungen besser organisiert und andererseits die vorhandenen und neuen Qualifikationen der Mitarbeiter erfasst werden. Neben freiwilligen Weiterbildungen sollen auch Weiterbildungen mit Abschlüssen / Zertifikaten aufgrund gesetzlicher Vorgaben berücksichtigt werden.

Entwerfen und implementieren Sie die Webanwendung "Mitarbeiterqualifizierung". Berücksichtigen Sie die nachfolgenden Erläuterungen.

Datenmodell

Das vereinfachte Datenmodell der Webanwendung sieht so aus:

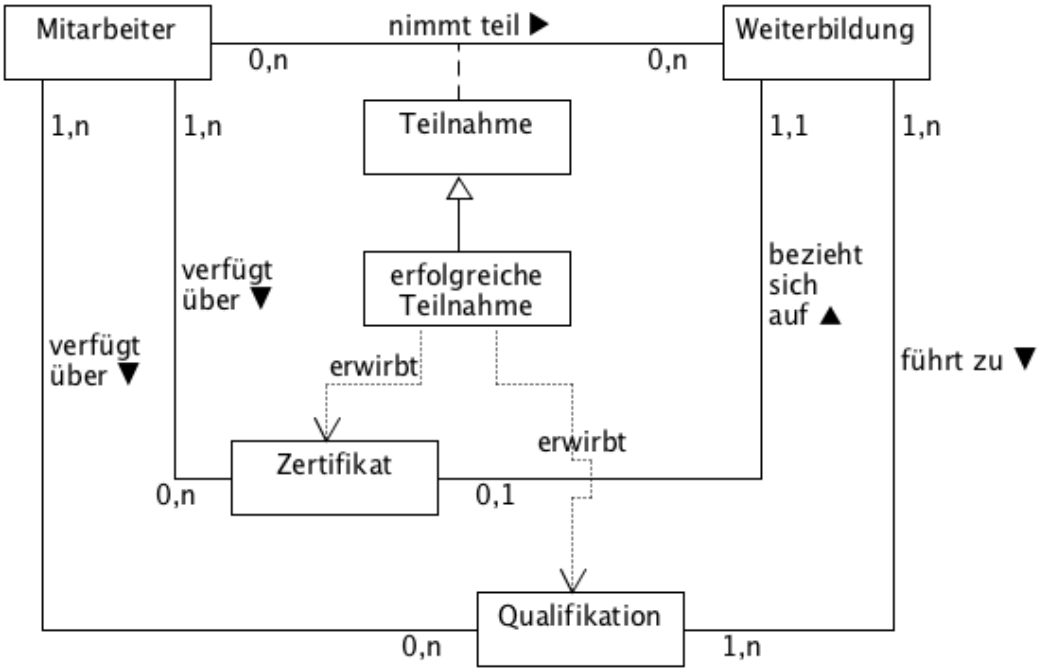

Abbildung: Datenmodell MiQu

Das Datenmodell ist folgendermaßen zu verstehen:

- ein Mitarbeiter kann über Qualifikationen verfügen
- ein Mitarbeiter kann über Zertifikate verfügen
- ein Mitarbeiter kann an Weiterbildungen teilnehmen
- eine Weiterbildung führt zu zumindest einer Qualifikation
- eine Weiterbildung kann zu einem Zertifikat führen
- im Falle einer erfolgreichen Teilnahme:
  - der Mitarbeiter erwirbt das Zertifikat, falls der Weiterbildung ein Zertifikat zugeordnet ist
  - der Mitarbeiter erwirbt die Qualifikationen, die der Weiterbildung zugeordnet sind.

Die Ableitung "erfolgreiche Teilnahme" verdeutlicht den Erfolgsfall bei der Teilnahme an einer Weiterbildung. Die bei der Weiterbildung erwerbbaren Qualifikationen und ggf. zugeordneten Zertifikate werden dann dem Mitarbeiter als weitere Qualifikation(en) und ggf. weitere Zertifikate zugewiesen.

Berücksichtigen Sie zumindest folgende Attribute:

- Mitarbeiter
  - Name: string
  - Vorname: string
  - akademische Grade: string
  - Tätigkeit: string
- Weiterbildung
  - Bezeichnung: string
  - Von: date
  - Bis: date
  - Beschreibung: text
  - maximale Teilnehmerzahl: integer
  - minimale Teilnehmerzahl: integer
- Teilnahme:
  - Status: Enumeration (angemeldet, nimmt teil, storniert, abgebrochen, nicht erfolgreich beendet, erfolgreich beendet)
- Zertifikat:
  - Bezeichnung: string
  - Beschreibung: text
  - berechtigt zu: string
- Qualifikation
  - Bezeichnung: string
  - Beschreibung: text.

Der Datentyp `text` ist eine mehrzeilige Zeichenkette.

Aufbau der Benutzungsschnittstelle

Im folgenden Wireframe (dt. "Drahtmodell") wird der Aufbau der Benutzungsschnittstelle und damit jeder Webseite der Anwendung angegeben.

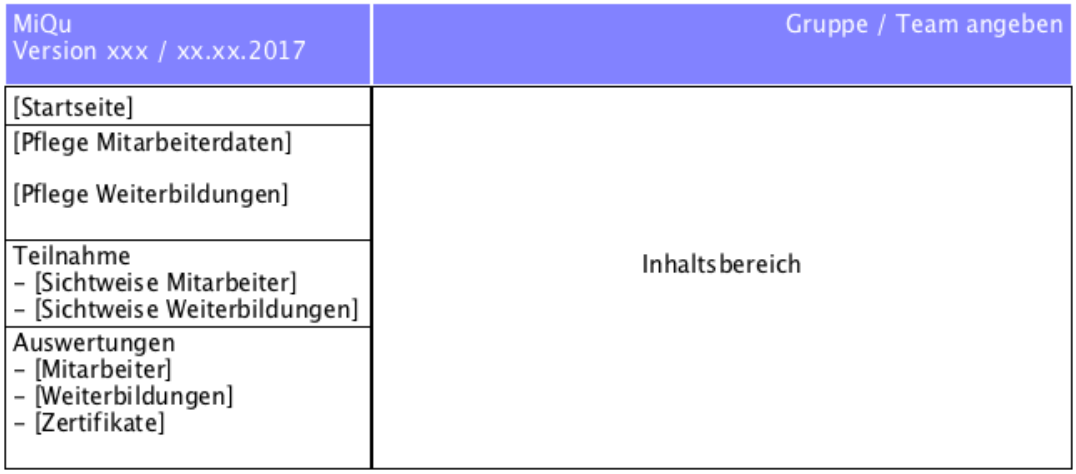

Abbildung: Wireframe MiQu

- der blau hinterlegte Bereich stellt den Kopfbereich (*Header*) dar:
  - links wird die Bezeichnung der Anwendung zusammen mit der aktuellen Versionsnummer und dem Gültigkeitsstand angezeigt
  - rechts tragen Sie Ihre Gruppenbezeichnung ein sowie die Namen und Matrikelnummern der Team-Mitglieder ein
- darunter befinden sich zwei Bereiche:
  - links befindet sich der Navigationsbereich (*Sidebar*), der bei allen Webseiten identisch aufgebaut ist
    - die eckigen Klammern stellen *Links* dar, d.h. diese Texte sind bedienbar und führen zur Anzeige einer neuen Webseite mit eigenem Inhaltsbereich (*die Klammern dienen nur der Verdeutlichung im Wireframe und sollen nicht in die Realisierung übernommen werden!*)
    - Texte, die nicht in Klammern gesetzt sind, sind Überschriften
  - rechts befindet sich der Inhaltsbereich, der je nach Webseite unterschiedlich aufgebaut ist.

Navigationsbereich

Die bedienbaren Einträge (siehe Abbildung) haben folgende Bedeutung:

- Startseite
  - im Inhaltsbereich wird die Startseite angezeigt mit folgenden Angaben:
    - Anzahl Mitarbeiter
    - Anzahl Weiterbildungen (in Planung, laufend, abgeschlossen)
    - Anzahl Teilnahmen
- Pflege Mitarbeiterdaten
  - im Inhaltsbereich wird eine Liste aller Mitarbeiterdaten angezeigt
  - es stehen die Aktionen "Erfassen", "Ändern" und "Löschen" zur Verfügung
    - bei "Erfassen" gelangt man in die Detailsicht zur Erfassung neuer Daten
    - bei "Ändern" gelangt man in die Detailsicht für die ausgewählten Mitarbeiterdaten
    - mit "Speichern" speichert man in der Detailsicht Änderungen
    - mit "zurück" gelangt man aus der Detailsicht wieder in die Listensicht
  - außerdem steht die Aktion "Anzeigen" zur Verfügung, mit der die Teilnahmen, Qualifikationen und Zertifikate des ausgewählten Mitarbeiters angezeigt werden
    - mit "zurück" gelangt man aus der Detailsicht wieder in die Listensicht
- Pflege Weiterbildungen
  - im Inhaltsbereich wird eine Liste aller Weiterbildungen angezeigt
  - es stehen die Aktionen "Erfassen", "Ändern" und "Löschen" zur Verfügung
    - bei "Erfassen" gelangt man in die Detailsicht zur Erfassung neuer Daten
    - bei "Ändern" gelangt man in die Detailsicht für die ausgewählten Weiterbildungsdaten
    - mit "Speichern" speichert man in der Detailsicht Änderungen
    - mit "zurück" gelangt man aus der Detailsicht wieder in die Listensicht
    - in der Detailsicht werden auch die Qualifikationen und Zertifikate erfasst / bearbeitet
  - außerdem steht die Aktion "Anzeigen" zur Verfügung, mit der die Teilnahmen sowie die zugeordneten Qualifikationen und Zertifikate angezeigt werden
    - mit "zurück" gelangt man aus der Detailsicht wieder in die Listensicht

(Teilnahme)

- Sichtweise Mitarbeiter:
  - im Inhaltsbereich wird die Liste der Mitarbeiter angezeigt
  - mit der Auswahl eines Mitarbeiters gelangt man in eine Anzeige
    - die den ausgewählten Mitarbeiter anzeigt
    - die Weiterbildungen anzeigt, an denen der Mitarbeiter teilnehmen kann
      - das sind die Weiterbildungen, die noch nicht begonnen haben und an denen der Mitarbeiter bisher nicht teilnimmt
    - die geplanten Weiterbildungen anzeigt, an denen der Mitarbeiter teilnehmen will; diese Teilnahme kann er hier stornieren

- Sichtweise Weiterbildungen:
  - im Inhaltsbereich wird die Liste der laufenden und geplanten (zukünftigen) Weiterbildungen angezeigt
  - die Auswahl einer Weiterbildung führt zu einer Detailsicht der Weiterbildung, die auch die Teilnehmer aufweist:
    - bei einer laufenden Weiterbildung kann eine Teilnahme angebrochen werden
    - bei einer abgeschlossenen Weiterbildung werden für die Teilnehmer Erfolg / Nicht-Erfolg erfasst.

(Auswertungen)

- Mitarbeiter:
  - im Inhaltsbereich wird eine alphabetisch sortierte Liste der Mitarbeiterdaten ausgegeben, die für jeden Mitarbeiter chronologisch dessen Teilnahme an Weiterbildungen ausweist:
    - entsprechend sind die Datumsangaben, die Bezeichnung der Weiterbildung sowie der Status der Teilnahme anzugeben
- Weiterbildungen
  - im Inhaltsbereich wird eine alphabetisch sortierte Liste der Weiterbildungen ausgegeben, die für jede Weiterbildung die erfolgreichen Teilnehmer ausweist
- Zertifikate
  - im Inhaltsbereich werden alphabetisch die vergebenen Zertifikate für die Mitarbeiter angezeigt.

Die Pflege der Mitarbeiterdaten und die Pflege der Weiterbildungen mit Listen- und Detailsichten erfolgt entsprechend dem einfachen Zustandsmodell der Aufgabenstellung 1 (ohne History-Funktion).

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 Webanwendung "MiQu" wird als Client-Server-Anwendung realisiert. In dieser Variante der Aufgabenstellung werden einzelne Webseiten, ggf. als Formulare, durch den Webserver erzeugt und ausgeliefert. Vermeiden Sie dabei grundsätzlich, Markup (HTML5) oder CSS direkt im Python-Code anzugeben: die serverseitige Erzeugung des Markup (HTML5) erfolgt deshalb mit Hilfe der **Template-Engine mako** (siehe Anhang).

Clientseitig werden nur die Standardmechanismen, die im User-Interface bei HTML5 direkt vorgesehen werden (Hyperlinks, einfache Formulare), verwendet. In Listen können Sie jeden Eintrag als Link implementieren und damit eine einfache Möglichkeit zum Übergang zu einem Detailformular schaffen.

Clientseitig kann ein einfacher Bestätigungsmechanismus bei löschenden Vorgängen entsprechend der Aufgabe 1 implementiert werden.

Eine weitergehende clientseitige Bearbeitung erfolgt in diesem Schritt nicht.

In Schritt 1 ist die Implementierung der *Auswertungsfunktionen noch nicht erforderlich*.

Anforderungen an die Umsetzung - Schritt 2 (Termin P3)

In Schritt 2 wird die Nutzung von Links in den Listensichten zum Wechsel in die Detailsichten ersetzt durch eine clientseitige Bearbeitung mit javascript:

- der zu bearbeitende Listeneintrag wird markiert
- die vorgesehene Funktion (z.B. "Ändern") wird mit einem Schalter realisiert
  - die Bedienung des Schalters wertet die Markierung in der Liste aus und führt daraufhin die entsprechende Anfrage zur Darstellung der nächsten Webseite durch.

In Schritt 2 werden die Auswertungsfunktionen implementiert.

Weitere Anforderungen an die Umsetzung

- Webclient:
  - Verwendung HTML5 (XML-konforme Notation)
  - überprüfung des Markup mit Hilfe der w3c-Validator-Dienste (siehe Anhang)
  - Präsentation mit CSS, ausgelagert in eine externe CSS-Datei
  - Verwendung javascript
- Webserver:
  - Verwendung Python (Version 3) (wie in Aufgabe 1)
  - Verwendung Framework "cherrypy" (wie in Aufgabe 1)
  - Verwendung der Template-Engine "mako" (siehe Anhang).

Die Verwendung von weiteren javascript-Bibliotheken / -Frameworks sowie CSS-Frameworks ist unzulässig!

Verwenden Sie folgende Verzeichnisstruktur und erstellen Sie die angegebenen Dateien (ggf. erst bei Schritt 2):

```
web
   /p2
       /bmf                  <--- server.py
           /app              <--- __init__.py, application.py, database.py, view.py
           /content          <--- html-Datei(en), css-Datei(en), js-Datei(en)
           /data             <--- Daten in JSON-formatierten Dateien
```

```
/doc              <--- Dateien der Dokumentation
/template         <--- Vorlagen für mako erst in Schritt 2
```

Sie können sich bei der Erstellung der Python-Module im Verzeichnis `app` *grundsätzlich* an der Aufgabenstellung 1 orientieren. Allerdings müssen Sie *wesentliche* Änderungen vornehmen! Beachten Sie die Hinweise im Anhang!

Anforderungen an die Dokumentation

Erstellen Sie eine Dokumentation, die Ihre Lösung (Schritt 1 und Schritt 2) beschreibt. Legen Sie dazu in einem Unterverzeichnis `doc` die Datei `bmf.md` an. Sehen Sie folgende Gliederung vor:

- einleitend: Ihre Gruppenzugehörigkeit, Aufbau Ihres Team, Gültigkeitsdatum der Dokumentation
- 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.

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 erstellte Dokumentation.

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)
  - 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

Datenbasis

Zur Vereinfachung werden die Daten serverseitig im Verzeichnis `data` als `JSON`-formatierte Dateien abgelegt.

Sie können sich bei Ihrer Implementierung an folgenden Überlegungen orientieren:

- jeder Instanz jeder im Datenmodell genannten Klasse wird durch einen Objekt-ID eindeutig identifiziert
- als Objekt-ID kann eine Ganzzahl verwendet werden, die zentral verwaltet wird
  - der aktuelle Wert wird z.B. in einer Datei abgelegt, damit er bei jedem Start des Webservers weiter verwendet werden kann
  - Werte werden nicht neu verwendet, es wird bei Anlegen neuer Instanzen einfach durch Inkrementieren eine neue Objekt-ID erzeugt
- für jede Klasse wird eine eigene JSON-Datei verwendet, die alle Instanzen der Klasse enthält
- für Beziehungen müssen ggf. ebenfalls eigene JSON-Datein verwendet werden
  - Beziehungen werden anhand der Objekt-IDs identiziert
- das von Ihnen zu implementierende Modul `database.py` kapselt die skizzierte Verwaltung der persistenten Daten und stellt den anderen Modulen eine geeignete Schnittstelle zur Bearbeitung zur Verfügung
  - direkte Zugriffe auf die (interne) Form der persistenten Daten durch andere Module sind unbedingt zu vermeiden.

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)* . 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 (z.B. Auszug aus einer Template-Datei `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_okey_s'name'}</td><td>${data_okey_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 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)
      # mit der Methode render wird das zuvor 'übersetzte' Template ausgeführt
      # data_o sind die im Template angegebenen Daten
      # data_opl die übergebenen Daten
```

```
29        markup_s = template_o.render(data_o = data_opl)
30        return  markup_s
31
32     #-------------------------------------------------------
33     def createList_px(self, data_opl):
34     #-------------------------------------------------------
35        return self.create_p('liste.tpl', data_opl)
36
37  # EOF
```

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.