Webanwendung "Praxisphasenmanager (PPM)"

Beschreibung der Anwendung

Der Fachbereich will zur Unterstützung der Betreuung von Praxisphasen eine Web-Anwendung erstellen lassen. Praxisphasen werden von externen Partnern (Firmen) angeboten, Studenten können diese Angebote nutzen, wenn sie einen Lehrenden als Betreuer finden.

Entwerfen und implementieren Sie die Web-Anwendung "Praxisphasenmanager (PPM)" mit folgendem Funktionsumfang:

- **Pflege der Daten der Studenten**
  - Anlegen, Bearbeiten und Löschen der Daten der Studenten
  - zu berücksichtigende Daten sind:
    - Name
    - Vorname
    - Matrikelnummer
- **Pflege der Daten der Lehrenden**
  - Anlegen, Bearbeiten und Löschen der Daten der Lehrenden
  - zu berücksichtigende Daten sind:
    - Titel
    - Name
    - Vorname
    - Lehrgebiet
- **Pflege des Firmenverzeichnisses**
  - Anlegen, Bearbeiten und Löschen der Firmendaten
  - zu berücksichtigende Firmendaten sind:
    - Firmenname
    - Branche
    - Tätigkeitsschwerpunkt
    - Sitz
    - Anzahl Mitarbeiter
- **Pflege des Praxisphasenangebots**
  - Anlegen, Bearbeiten und Löschen der Angebote
  - mit Bezug zu einer Firma
  - Beschreibung der Aufgabenstellung
  - Beschreibung der Voraussetzungen, die der Student erfüllen soll
  - Angaben zum Mitarbeiter, der den Studenten betreuen soll (Firmenbetreuer)
- **Auswahl eines angebotenen Praxisphasenplatzes durch einen Studenten**
  - mit der Auswahl muss der Lehrende, der den Studenten an der Hochschule betreut, eingetragen werden
  - der Praxisphasenplatz wird damit aus dem Angebot herausgenommen
    - Änderungen der Beschreibung etc. sind nicht mehr möglich
    - der Praxisphasenplatz kann auch nicht mehr aus dem Datenbestand entfernt werden
  - der Zeitraum der Praxisphase muss eingetragen werden
    - wenn die Praxisphase beendet ist, bleibt der Praxisphasenplatz mit diesen Angaben als "abgeschlossen" im Datenbestand
- **Auswertungen / Übersichten**
  - Firmenverzeichnis mit allen Firmen und allen Praxisphasenplätzen
    - alphabetisch sortiert nach Firmenname
    - Praxisphasenplätze gruppiert nach "Angebot", "aktuell" und "abgeschlossen"
    - innerhalb der Gruppen "aktuell" und "abgeschlossen" nach Anfangsdatum

- Verzeichnis der aktuellen und abgeschlossenen Praxisphasen nach Studenten
  - mit Angabe der Betreuer und der Firmen und den Angaben zur Praxisphase
- Verzeichnis der aktuellen und abgeschlossenen Praxisphasen nach Betreuern
  - mit Angabe der Studenten und der Firmen und den Angaben zur Praxisphase.

Beachten Sie:

- eine Firma kann mehrere Praxisphasenplätze anbieten
- ein Student kann genau einen Praxisphasenplatz nutzen
- ein Lehrender kann mehrere Praxisphasen betreuen.

Das Datenmodell der Anwendung sieht etwa so aus:

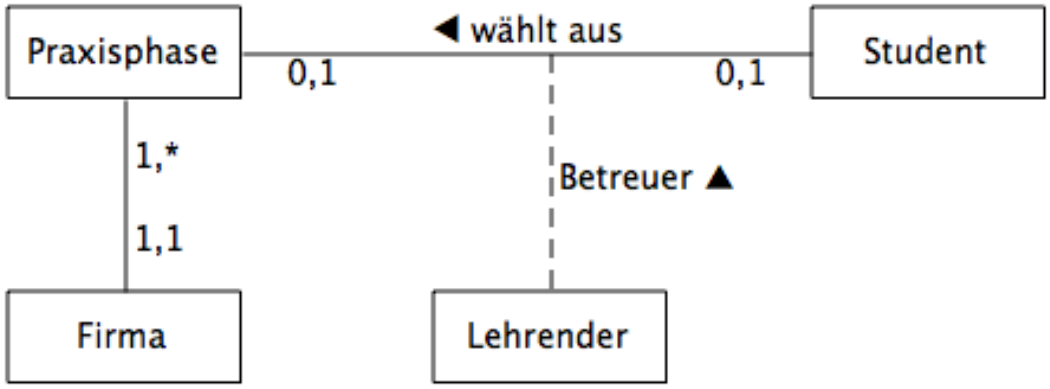

Abbildung: Datenmodell PPM

Sehen Sie folgende **Webseiten** als Sichten der Web-Anwendung vor:

- **Startseite** mit Verzweigung zu:
  - Pflege der Daten der Studenten
  - Pflege der Daten der Lehrenden
  - Firmenverzeichnis
  - Angebot Praxisphasen
  - Auswertungen (einzeln aufführen)
- für die Pflege der **Daten der Studenten**
  - **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 Studenten
  - **Detailformular**
    - zum Erfassen neuer Daten eines Studenten
    - zum Bearbeiten der Daten eines Studenten
- für die Pflege der **Daten der Lehrenden**
  - **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 Lehrenden
  - **Detailformular**

        - zum Erfassen neuer Daten eines Lehrenden
        - zum Bearbeiten der Daten eines Lehrenden
- für das **Firmenverzeichnis**
    - **Auswahlliste**
        - Auswahl einzelner Einträge zur weiteren Bearbeitung
        - Auswahl einzelner Einträge zum Löschen
            - dann müssen alle davon abhängigen Daten gelöscht werden
        - Übergang zum Erfassen neuer Firmendaten
    - **Detailformular**
        - zum Erfassen neuer Firmendaten
        - zum Bearbeiten der Daten einer Firma
- für das **Praxisphasenangebot**
    - **Auswahlliste**
        - Auswahl einzelner Einträge zur weiteren Bearbeitung
        - Auswahl einzelner Einträge zum Löschen
        - Übergang zum Erfassen eines neuen Angebots
        - man beachte den Bezug zur anbietenden Firma!
    - **Detailformular**
        - zum Erfassen eines neuen Angebots
        - zum Bearbeiten der Daten eines Angebots
- für die **Auswahl eines angebotenen Praxisphasenplatzes durch einen Studenten**
    - **Auswahlliste** Praxisphasenangebote
        - Auswahl eines Eintrags zur Annahme durch den Studenten
        - man beachte den Bezug zur anbietenden Firma!
    - **Detailformular**
        - zum Erfassen der Annahme des Angebots durch einen Studierenden
            - die Daten des Studierenden und des Betreuers sowie der Zeitraum müssen eingetragen werden
- für die **Auswertungen / Übersichten**
    - Firmenverzeichnis mit allen Firmen und allen Praxisphasenplätzen
    - Verzeichnis der aktuellen und abgeschlossenen Praxisphasen nach Studenten
    - Verzeichnis der aktuellen und abgeschlossenen Praxisphasen nach Betreuern.

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 "Praxisphasenmanager (PPM)" wird als Client-Server-Anwendung realisiert. In dieser Variante der Aufgabenstellung werden einzelne Webseiten, ggf. als Formulare, durch den Web-Server mit Hilfe einer Template-Engine erzeugt und ausgeliefert.

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 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 wird die Anwendung clientseitig als *Single-Page-Application* implementiert. Die eigentliche HTML-Seite wird einmal geladen, weitere Inhalte werden dynamisch im Hintergrund geladen (*Ajax-Ansatz*). Diese Inhalte werden vom Server als JSON-formatierte Daten bereitgestellt und *clientseitig* mit der Template-Engine `te/tm` (siehe Hinweis uten) ausgegeben.

Die clientseitige Implementierung erfolgt jetzt mit Hilfe der 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) (wie in Aufgabe 1)
  - Verwendung Framework "cherrypy" (wie in Aufgabe 1)
  - Verwendung der Template-Engine "mako" (siehe Anhang) (für Schritt 1).

Verwenden Sie folgende Verzeichnisstruktur (ppm1 - Schritt 1, ppm2 - Schritt 2) und erstellen Sie die angegebenen Dateien:

```
web
   /p2
       /ppm1                <--- 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 (ppm1.md + weitere)
           /templates       <--- Vorlagen für mako
       /ppm2                <--- 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 (ppm2.md + weitere)
           /templates       <--- Vorlagen für clientseitige Templates
```

Sie können sich bei der Erstellung der Python-Module im Verzeichnis `app` an der Aufgabenstellung 1 orientieren. Allerdings müssen Sie Änderungen vornehmen!

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 Dateien `ppm1.md` (für Schritt 1) bzw. `ppm2.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 Client
- 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 Quellcoder 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):

```python
# 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.