Webanwendung "Modulhandbuch (mhb)"

Beschreibung der Anwendung

Sie sollen eine Webanwendung entwickeln, mit der die Beschreibungen der Studiengänge und Module abgerufen bzw. gepflegt werden können. Entwerfen und implementieren Sie die Webanwendung "Modulhandbuch". Berücksichtigen Sie die nachfolgenden Erläuterungen.

Datenmodell

Das vereinfachte Datenmodell der Webanwendung sieht so aus:

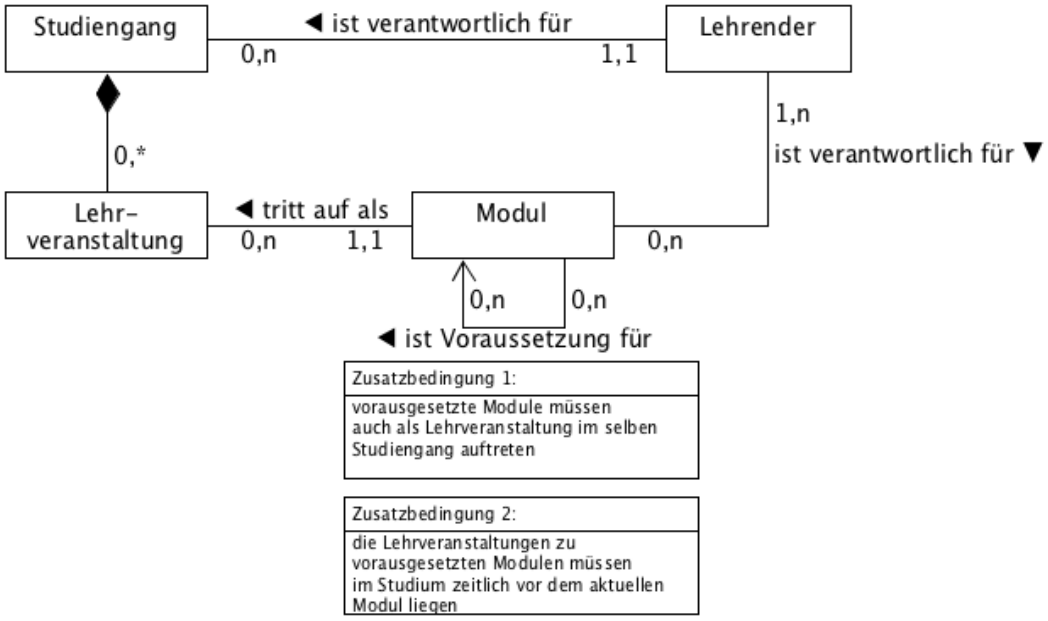

Abbildung: Datenmodell mhb

- bei jedem Studiengang werden angegeben:
  - Bezeichung: string
  - Kurzbezeichnung: string
  - Beschreibung: text
  - Anzahl Semester: number
- bei jeder Lehrveranstaltung werden angegeben:
  - Bezeichnung: string
  - Kurzbezeichnung: string
  - Lage (Semester): number
- bei jedem Modul werden angegeben:
  - Bezeichnung: string
  - Beschreibung: text
  - Anzahl Kreditpunkte: number
  - Anzahl SWS - Vorlesung: number
  - Anzahl SWS - Übung: number
  - Anzahl SWS - Praktikum: number

- Voraussetzungen (andere Module)

(Hinweis: der Datentyp `text` ist eine mehrzeilige Zeichenkette.)

Der Aufbau der Studiengänge ergibt sich aus:

- den einzelnen Lehrveranstaltungen in den einzelnen Semestern
- der Zuordnung von Modulen zu Lehrveranstaltungen
  - *beachten Sie, dass ein Modul Lehrveranstaltungen in unterschiedlichen Studiengänge zugeordnet werden kann*
  - Lehrveranstaltungen können anders benannt werden als das jeweils zugrunde liegende Modul.

Die Angabe von vorausgesetzten Modulen ist optional - beachten Sie die Zusatzbedingungen, die im Diagramm (oben) angegeben sind.

Weitere fachliche Anforderungen

Mit der Anwendung soll es möglich sein, neben der Datenpflege ein Modulhandbuch zu erstellen:

- das Modulhandbuch zu einem Studiengang enthält
  - die Zusammenstellung der Angaben zum Studiengang
  - eine Übersicht mit allen Lehrveranstaltungen, alphabetisch sortiert
    - auch die Modulverantwortlichen werden aufgeführt
  - einen detaillierten Semesterplan:
    - vom 1. bis zum letzten Semester des Studiengangs werden die Lehrveranstaltungen mit allen Daten ausgewiesen
    - die je Semester erzielbaren Kreditpunkte werden ausgewiesen
    - die insgesamt erzielbaren Kreditpunkte werden ausgewiesen
- es werden drei Rollen vorgesehen:
  - Rolle "Studierender"
    - kann alle Studiengänge einsehen
    - kann zu jedem Studiengang das Modulhandbuch abrufen
  - Rolle "Verantwortlicher Modul"
    - kann die Daten der ihm zugewiesenen Module bearbeiten (aber keine Module erstellen oder löschen!)
    - kann alle Studiengänge einsehen
    - kann zu jedem Studiengang das Modulhandbuch abrufen
  - Rolle "Verantwortlicher Studiengang"
    - erstellt und pflegt die Studiengänge
    - erstellt die Module (gibt dabei nur Bezeichnung an - Pflege der weiteren Inhalte siehe andere Rolle)
    - legt die Zuordnung der Benutzer der Rolle "Verantwortlicher Modul" zu Modulen fest
    - kann Module löschen
    - legt die Struktur eines Studiengangs anhand der Lehrveranstaltungen in den einzelnen Semestern fest
    - legt die Zuordnung der Module zu Lehrveranstaltungen fest
    - legt die Modulabhängigkeiten (Voraussetzungen) fest.

Aufbau der Benutzungsschnittstelle

Zur Unterscheidung der genannten Rollen wird ein Anmeldungsdialog vorgesehen:

- Studenten identifizieren sich mit ihrer EMail-Adresse (Muster: xxx@stud.hn.de), ein Passwort ist nicht erforderlich
  - eine Speicherung der Anmeldedaten der Studenten ist nicht erforderlich
- Lehrende identifizieren sich mit ihrer EMail-Adresse und ihrem Passwort
  - die Angaben sind im Datenbestand hinterlegt, eine spezielle Datenpflege ist nicht erforderlich
  - aus den gespeicherten Daten ergibt sich dann auch die Rolle "Verantwortlicher Modul" und/oder "Verantwortlicher Studiengang"

        - ein Lehrender kann beide Rollen gleichzeitig einnehmen

Die Folgeseiten sind abhängig von der Rolle und werden nachfolgend beschrieben.

Rolle Studierender

Es wird eine Seite mit der Liste der Studiengänge angeboten:

- bei jedem Studiengang wird die Liste der Lehrveranstaltungen in alphabetischer Folge angezeigt
- zu jedem Studiengang kann das Modulhandbuch abgerufen werden:
    - das Modulhandbuch wird dann in einer neuen Registerkarte oder einem neuen Fenster des Webbrowsers angezeigt
- es besteht die Möglichkeit, sich wieder abzumelden; damit wird wieder die Anmeldeseite angezeigt.

Rolle "Verantwortlicher Modul"

Es wird eine Seite mit der Liste der Studiengänge angeboten:

- bei jedem Studiengang wird die Liste der Lehrveranstaltungen und der zugrunde liegende Module in alphabetischer Folge (nach Bezeichnung Lehrveranstaktungen) angezeigt
    - alternativ kann eine Darstellung nach Semester und in einem Semester in alphabetischer Folge erfolgen
    - bei jeder Lehrveranstaltung wird gekennzeichnet, ob der angemeldete Lehrende die Daten des zugrunde liegenden Moduls bearbeiten kann:
        - dann kann ein Bearbeitungsformular für das entsprechende Modul aufgerufen werden
- zu jedem Studiengang kann das Modulhandbuch abgerufen werden:
    - das Modulhandbuch wird dann in einer neuen Registerkarte oder einem neuen Fenster des Webbrowsers angezeigt
- es besteht die Möglichkeit, sich wieder abzumelden; damit wird wieder die Anmeldeseite angezeigt.

Rolle "Verantwortlicher Studiengang"

Es wird eine Webseite mit der Liste der Studiengänge und der Liste der Module angeboten:

- Studiengänge können erfasst, bearbeitet und gelöscht werden
- zu jedem Studiengang kann das Modulhandbuch abgerufen werden:
    - das Modulhandbuch wird dann in einer neuen Registerkarte oder einem neuen Fenster des Webbrowsers angezeigt
- Module können erfasst, bearbeitet und gelöscht werden
- es besteht die Möglichkeit, sich wieder abzumelden; damit wird wieder die Anmeldeseite angezeigt.

Bei Erfassen bzw. Bearbeiten wird eine passende Webseite angezeigt, die die Bearbeitung der Detailangaben ermöglicht:

- Studiengang:
    - Attribute des Studiengangs
    - Anzeige der Liste der Lehrveranstaltungen mit den Möglichkeiten:
        - neue Lehrveranstaltungen zu erfassen
        - die Daten von Lehrveranstaltungen zu bearbeiten
        - Lehrveranstaltungen zu löschen
        - bei Erfassen und Bearbeiten wird zu einer Webseite gewechselt, die das passende Formular enthält und auch die Zuordnung des Moduls ermöglicht; von dieser Seite aus gelangt man wieder zur Anzeige des Studiengangs
- Modul:
    - Modulbezeichnung
    - Zuordnung Lehrende ("Verantwortliche Modul")
    - Anzeige der Liste der zugeordneten Lehrveranstaltungen
    - Anzeige der Liste der Module, die Voraussetzung für das aktuelle Modul sind
        - mit der Möglichkeit, die Liste zu erweitern (weitere Module als Voraussetzung) oder zu verkleinern (durch entfallende Voraussetzungen)
        - die beiden Zusatzbedingungen im Datenmodell sind zu beachten.

Zustandmodell

Im nachfolgenden Diagramm finden Sie ein Zustandsmodell der Webanwendung, das die Abfolge der einzelnen Webseiten verdeutlichen soll.

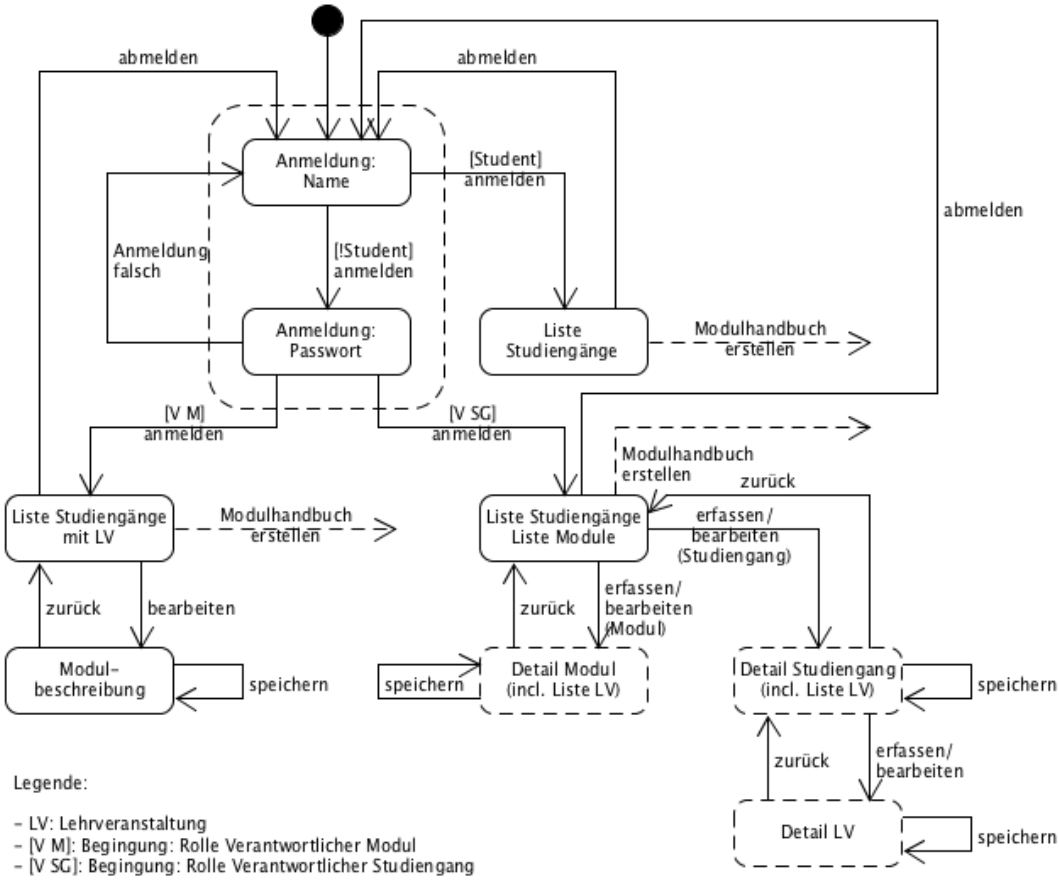

Abbildung: Zustandsmodell mbh

Das Diagramm verwendet folgende spezielle Darstellungsweisen:

- *"Modulhandbuch erstellen"*: die gestrichelte Kante soll verdeutlichen, dass ein paralleler Zustand erreicht wird (Darstellung des Modulhandbuchs in einem anderen Fenster / einer anderen Registerkarte des Webbrowsers); der andere Zustand ist hier nicht dargestellt
- die Umrahmung der beiden Zustände *"Anmeldung: Name"* und *"Anmeldung: Passwort"* soll verdeutlichen, dass die beiden Zustände zwei unterschiedliche Repräsentationen eines Objekts sind; es wird also nicht das Objekt gewechselt, sondern nur die Repräsentation: von einer Formularversion zur nächsten
- ebenso sollen die mit einer gestrichelten Umrahmung versehenen Dialoge *"Detail Modul (incl. Liste LV)"*, *"Detail Studiengang (incl. Liste LV)"* und *"Detail LV"* verdeutlichen, dass zwei Präsentationen verwendet werden (für die Erfassung neuer Daten, für die Bearbeitung bestehender Daten).

Anforderungen an die Umsetzung - Schritt 1 (Termin P2)

Die Webanwendung "Modulhandbuch" 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 wird zusätzlich zu den Standardmechanismen, die im User-Interface bei HTML5 direkt vorgesehen werden (Hyperlinks, einfache Formulare), auch javascript verwendet:

- Ereignisverarbeitung der Navigation
- Listen:
  - 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
- Bestätigung von Löschvorgängen
- Rückmeldungen an den Benutzer (modale Standard-Dialoge).

Anforderungen an die Umsetzung - Schritt 2 (Termin P3)

In Schritt 2 wird die Anwendung als *Single-Page-Application* implementiert:

- es wird einmalig eine HTML-Seite geladen, die Ausgangspunkt für alle weiteren Vorgänge ist
- weitere Inhalte werden im Hintergrund asynchron per Ajax (oder ggf. der neueren Fetch-API) beschafft und dargestellt.

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
    - die Gestaltung muss von Form und Farbe her so erfolgen, dass die Anwendung ästhetisch ansprechend und gut bedienbar ist
  - 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:

```
web
   /p2
       /mhb                  <--- server.py
           /app              <--- __init__.py, application.py, database.py, view.py, ggf. weitere
           /content          <--- html-Datei(en), css-Datei(en), js-Datei(en)
           /data             <--- Daten in JSON-formatierten Dateien
           /doc              <--- Dateien der Dokumentation
           /template         <--- Vorlagen für mako
```

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 bei Schritt 1 und Schritt 2 eine Dokumentation, die Ihre Lösung (Schritt 1 bzw. Schritt 2) beschreibt. Legen Sie dazu in einem Unterverzeichnis `doc` die Datei `mhb.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:

- Testat Schritt 1:
  - den von Ihnen erstellten Quellcode Ihrer Web-Anwendung (Variante Schritt 1)
  - die von Ihnen erstellte Dokumentation dazu
- Testat Schritt 2:
  - den von Ihnen erstellten Quellcode Ihrer Web-Anwendung (Variante Schritt 2)
  - die von Ihnen erstellte Dokumentation dazu

Sie müssen bei beiden Schritten 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_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 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.