Ziele

Beim ersten Praktikumstermin sollen Sie einerseits die Arbeitsumgebung einrichten und andererseits durch Ergänzung einer vorgegebenen Lösung für eine Webanwendung einen ersten Eindruck der verschiedenen softwaretechnischen Aspekte erhalten.

Teil 1: Arbeitsumgebung einrichten

Schritt 1.1: Einrichtung Werkzeuge / Komponenten auf Ihrem System

Sie benötigen verschiedene Werkzeuge und Software-Komponenten, um die Aufgabenstellungen zu bearbeiten, die in der Veranstaltung **Web-Engineering** behandelt werden:

- Laufzeitsystem für die Programmiersprache Python
  - lizenzkostenfrei erhältlich bei www.python.org, Version 3.6 verwenden!
  - `http://www.python.org/downloads` (“looking for a specific release?”")
- Entwicklungsumgebung “Visual Studio Code” (VSCode oder VSC) mit der Erweiterung “Python” verwenden
- und / oder Quelltexteditor, lizenzkostenfreie / ohne Kosten verwendbare Beispiele (falls Sie einen zusätzlichen Editor verwenden wollen):
  - für MS-Windows-Nutzer: notepad++, Sublime 3, Atom, VS Code
  - für Linux / Mac OS -Nutzer: SciTe oder Sublime 3, Atom, VS Code
    - Hinweis: Sublime 3 kann kostenfrei benutzt werden, es erscheint dann von Zeit zu Zeit ein Hinweis
- Webbrowser FireFox
  - installieren Sie die Erweiterung “Web Developer Toolbar”
- oder Webbrowser Chrome / Chromium
- Python Framework zur Webserver-Programmierung “cherrypy”
  - lizenzkostenfrei erhältlich bei `www.cherrypy.org` bzw. `https://pypi.python.org/pypi/CherryPy`
  - ggf. müssen Sie vor der Installation die “setuptools” installieren
  - nach Download des Archivs und Entpacken kann die Installation so erfolgen (Konsole/Kommandozeile):

```
python setup.py install
```

bitte beachten Sie dabei:

- entpacken Sie das Archiv nicht auf den Desktop!
- entpacken Sie das Archiv in ein temporäres Verzeichnis, das sie nach der Installation vollständig (samt Inhalt) entfernen können

Schritt 1.2: Einrichtung prüfen

Legen Sie folgende Verzeichnisstruktur in einem Verzeichnis Ihrer Wahl an (MS-Windows-Nutzer: das sollte nicht der Desktop sein!):

```
web
   /p1
       /test
             /app
             /content
```

Erstellen Sie im Verzeichnis `web/p1/test` die Datei `testserver.py` (Inhalt: siehe Anlage 1 / 1).

Erstellen Sie im Verzeichnis `web/p1/test/app` die Dateien `__init__.py` und `application.py` (Inhalt: siehe Anlage 1 / 2 und Anlage 1 / 3).

Erstellen Sie im Verzeichnis `web/p1/test/content` die Datei `index.html` (Inhalt: siehe Anlage 1 / 4).

Achten Sie darauf, bei allen Textdateien die Zeichenkodierung UTF-8 (ohne BOM [Byte Order Mark]]) zu verwenden!

öffnen Sie eine Konsole und wechseln Sie in das Verzeichnis `web/p1/test` und starten Sie dort den Webserver durch die Eingabe:

```
python testserver.py
```

Starten Sie den Webbrowser und geben Sie die Adresse `http://localhost:8080/` an. Betätigen Sie den Link. Als Ergebnis sollte Ihnen die Version des installierten cherrypy-Frameworks angezeigt werden.

Welche Anzeige erhalten Sie, wenn Sie die Adresse `http://localhost:8080/home.html` angeben?

Teil 2: Webanwendung “WEBTeams”

Beschreibung der Anwendung

Im zweiten Teil sollen Sie die Webanwendung “WEBTeams” vervollständigen. Mit dieser Anwendung sollen Sie die Namen und Matrikelnummern aller Teams Ihrer Praktikumsgruppe erfassen.

Die Webanwendung besteht aus den beiden folgenden Seiten:

- Startseite: Auflistung der Daten der erfassten Teams
- Formular: Erfassung neuer Daten oder Bearbeitung vorhandener Daten.

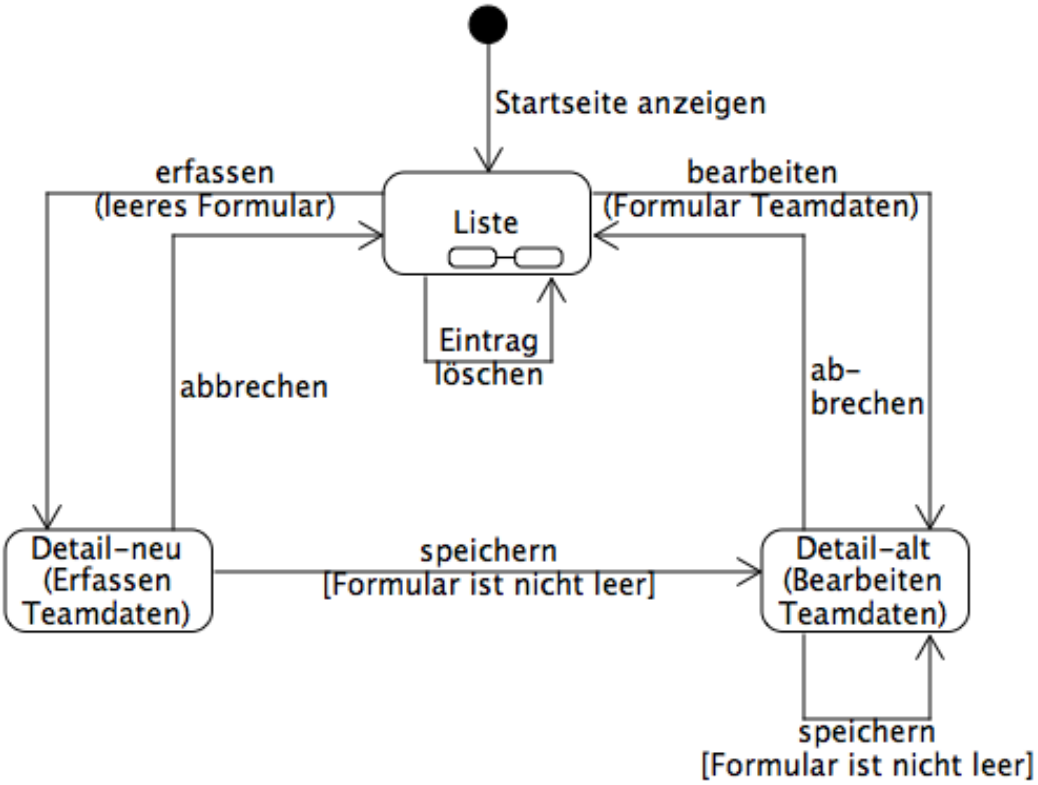

Abbildung 1: Zustandsmodell Webanwendung

In der Abbildung sind die drei Zustände der Webanwendung dargestellt:

- Zustand “Liste”:

    - wird beim Aufruf der Webanwendung eingenommen
    - zeigt die Liste der Teams an
    - mögliche Aktionen sind:
        - erfassen: führt in den Zustand “Detail-neu”
        - bearbeiten: führt in den Zustand “Detail-alt”
        - löschen: Daten eines Web-Teams entfernen und Liste wieder anzeigen; d.h. der Zustand wird nicht verlassen
- Zustand “Detail-neu”:
    - es wird ein leeres Formular angezeigt
    - mögliche Aktionen sind:
        - abbrechen: führt (ohne Speichern) in den Zustand “Liste” zurück
        - speichern: der Formularinhalt wird gespeichert, führt in den Zustand “Detail-alt”
            - kann nur ausgeführt werden, wenn der Formularinhalt nicht leer ist
- Zustand “Detail-alt”:
    - das Formular zeigt die Daten des ausgewählten Teams an
    - mögliche Aktionen sind:
        - abbrechen: führt (ohne Speichern) in den Zustand “Liste” zurück
        - speichern: der Formularinhalt wird gespeichert, der Zustand wird nicht verlassen.
            - kann nur ausgeführt werden, wenn der Formularinhalt nicht leer ist

Die Darstellung des Zustands “Liste” zeigt an, dass dieser Zustand weiter verfeinert ist. Diese Verfeinerung ist im folgenden Diagramm dargestellt.

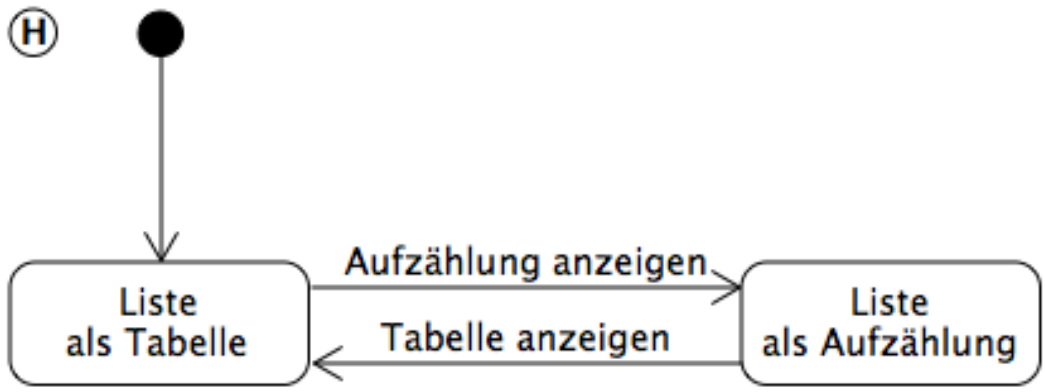

Abbildung 2: Verfeinerung Zustand Liste

Wenn der Zustand “Liste” zum ersten Mal eingenommen wird (siehe Abbildung 1), dann wird der Unterzustand “Liste als Tabelle” eingenommen. Das Symbol “H” (umrandet) bedeutet, dass sich der Automat anschließend den zuletzt eingenommenen Zustand merkt. Die Umschaltung der Darstellung der Liste bleibt also auch dann erhalten, wenn zwischenzeitlich der Zustand “Detail-Neu” oder “Detail-Alt” (siehe Abbildung 1) angenommen wurde.

In beiden Unterzuständen sind die Bedienungen und Zustandswechsel, die im übergeordneten Automaten erfolgen (siehe Abbildung 1), identisch.

Schritt 2.1: Verzeichnisse und Dateien erstellen

Ergänzen Sie die in Teil 1 eingerichtete Verzeichnisstruktur:

```
web
    /p1
```

```
/webteams
        /app
        /content
        /data
        /doc
        /templates
```

Erstellen Sie im Verzeichnis `web/p1/webteams` die Datei `server.py` (siehe Anlage 2 / 1). Diese Datei ist nahezu identisch mit der Datei `testserver.py`, beachten Sie die Änderung bei der Definition der Variablen `static_config`!

Erstellen Sie im Verzeichnis `web/p1/webteams/app` die Dateien

- `__init__.py` (wie in Teil 1)
- `application.py` (Inhalt: siehe Anlage 2 / 2)
  - nimmt die Anfragen des Webclient entgegen (*Requests*) und erzeugt die Antworten des Webservers (*Responses*)
  - verwendet dazu die Methoden aus den beiden anderen Modulen
- `database.py` (Inhalt: siehe Anlage 2 / 3)
  - implementiert eine sehr einfache Datenhaltung
- `dataid.py` (Inhalt: siehe Anlage 2 / 4)
  - implementiert die Erzeugung einer eindeutigen Identifikation
- `view.py` (Inhalt: siehe Anlage 2 / 5)
  - erzeugt das Markup, das ausgeliefert werden soll.

Erstellen Sie im Verzeichnis `web/p1/webteams/content`

- die zunächst leere Datei `webteams.css`
- die Datei `webteams.js` (Inhalt: siehe Anlage 2 / 7).

Erstellen Sie im Verzeichnis `web/p1/webteams/templates`

- die Vorlagen für das Markup der Liste und des Formulars (Inhalt: siehe Anlage 2 / 6).

Achten Sie darauf, bei allen Textdateien die Zeichenkodierung UTF-8 (ohne BOM [Byte Order Mark]]) zu verwenden!

Schritt 2.2: aktuellen Stand überprüfen

Verwenden Sie den Python-Debugger in der Entwicklungsumgebung “VSCode”, um den Bearbeitungsablauf im Webserver zu überprüfen. Überprüfen Sie z.B. durch Setzen von Breakpoints, ob die im Zustandsmodell angegebenen Aktionen serverseitig bearbeitet werden können.

Nutzen Sie im Menü “Entwicklerwerkzeuge” des Webbrowser (z.B. FireFox) das Werkzeug “Netzwerkanalyse”, um den Datenverkehr zwischen dem Webclient und dem Webserver zu analysieren.

Schritt 2.3: Ergänzungen vornehmen

Die in Schritt 2.1 erstellte Implementierung ist nicht vollständig. Ergänzen Sie die nachfolgenden Eigenschaften.

Ergänzung: “step 1”

Berücksichtigen Sie die Dateneingabe für die Daten des 2. Team-Mitglieds im Formular (zu bearbeitende Dateien: `form.tpl`, `application.py`, `database.py`).

Ergänzung: “step 2”

Berücksichtigen Sie die Daten des 2. Team-Mitglieds in der Liste (zu bearbeitende Datei: `liste.tpl`)

Ergänzung: “step 3”

Berücksichtigen Sie das zusätzliche Attribut “Semesteranzahl” (für beide Teammitglieder) im Formular sowie in der Datenbasis (zu bearbeitende Dateien: `form.tpl`, `application.py`, `database.py`).

Ergänzung: “step 4”

Implementieren Sie die Aktion “Abbrechen” im Formular (zu bearbeitende Datei: `form.tpl`).

Ergänzung: “step 5”

Implementieren Sie das Löschen von Einträgen in der Liste mit Rückfrage, *redirect* statt direkter Listenerzeugung (zu bearbeitende Dateien: `list.tpl`, `application.py`, `database.py`, `webteams.js`).

Ergänzung: “step 6”

Gestalten Sie die beiden Seiten mit Hilfe von CSS: tragen Sie die CSS-Stilregeln dazu in die Datei `webteams.css` ein.

Ergänzung: “step 7”

- implementieren Sie die Darstellung der Liste als Aufzählung und ergänzen Sie die Möglichkeit zur Umschaltung der beiden Darstellungsweisen
- implementieren Sie die “History”-Funktion der Darstellung der Liste (siehe oben, Abbildung 2)
- zu bearbeitende Dateien: `list.tpl`, `list2.tpl`, `form.tpl`, `application.py`, `view.py`.

Hinweis zur Implementierung der “History”-Funktion

Die “History”-Funktion bedeutet, dass beim Übergang zum Zustand “Liste” (Aufruf mit der URI “/”) eine Information vorhanden sein muss, die dem Webserver mitteilt, welche Darstellungsform geliefert werden soll. Dazu kann man beim Zustandswechsel vom Zustand “Liste” zu einem der beiden Detail-Zustände mitgeben, welche Darstellungsform gerade aktiv ist. Diese Angabe kann verwendet werden, um beim Formular den Verweis auf die Liste um einen entsprechenden Hinweis zu ergänzen (etwa so: `href=“/?listform=tabelle”).

Sie müssen dazu sowohl die Templates als auch die Verarbeitung des Webservers weiterentwickeln.

Schritt 2.4: Dokumentation erstellen

Erstellen Sie eine Dokumentation. Legen Sie dazu im Verzeichnis `web/p1/webteams/doc` die Datei `webteams.md` an. Schreiben Sie die Dokumentation als `Markdown`-Dokument und sehen Sie folgende Gliederung vor:

- Aufbau der Webanwendung
- Durchgeführte Ergänzungen
- Beschreibung des HTTP-Datenverkehrs
  - beim Start der Anwendung
  - beim Speichern von Formulardaten
  - verwenden Sie Screenshots der “Netzwerkanalyse” des Webbrowser und geben Sie an
    - welche Anfragen an den den Webserver geschickt werden (HTTP-Methode, URI, Inhalt der Anfrage)

- welche Antworten der Webserver liefert (Inhalt beschreiben).

Geben Sie einleitend Ihre Gruppenzugehörigkeit, den Aufbau Ihres Teams und das Gültigkeitsdatum der Dokumentation an.

Die Dokumentation wird als utf-8 kodierter Text mit der einfachen Auszeichnungssprache “markdown” erstellt. Mit Hilfe des Werkzeugs “pandoc” (siehe http://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.

Testat

Sie erhalten das Testat, wenn Sie Teil 1 und Teil 2 erfolgreich bearbeiten.

Anlagen Teil 1

1 / 1: Datei testserver.py

```
#coding: utf-8
import os
import cherrypy
from app import application

#--------------------------------------
def main():
#--------------------------------------
   # Get current directory
   try:
      current_dir = os.path.dirname(os.path.abspath(__file__))
   except:
      current_dir = os.path.dirname(os.path.abspath(sys.executable))
   # disable autoreload and timeout_monitor
   cherrypy.engine.autoreload.unsubscribe()
   cherrypy.engine.timeout_monitor.unsubscribe()
   # Static content config
   static_config = {
      '/': {
            'tools.staticdir.root': current_dir,
            'tools.staticdir.on': True,
            'tools.staticdir.dir': './content',
            'tools.staticdir.index': 'index.html'
      }
   }
   # Mount static content handler
   root_o = cherrypy.tree.mount(application.Application_cl(), '/', static_config)
   # suppress traceback-info
   cherrypy.config.update({'request.show_tracebacks': False})
   # Start server
   cherrypy.engine.start()
   cherrypy.engine.block()

#--------------------------------------
if __name__ == '__main__':
#--------------------------------------
   main()
# EOF
```

1 / 2: Datei __init__.py

```
# kennzeichnet ein Verzeichnis als Python-Package
```

1 / 3: Datei application.py

```
# coding: utf-8

import cherrypy

#-------------------------------------
class Application_cl(object):
#-------------------------------------
   #---------------------------------
   def __init__(self):
   #-------------------------------------
      # constructor
      pass

   @cherrypy.expose
   #-------------------------------------
   def greeting(self):
   #-------------------------------------
      return "Cherrypy-Server, Version %s" % cherrypy.__version__

   @cherrypy.expose
   #-------------------------------------
   def default(self, *arglist, **kwargs):
   #-------------------------------------
      msg_s = "unbekannte Anforderung: " + \
              str(arglist) + \
              ''+\
              str(kwargs)
      raise cherrypy.HTTPError(404, msg_s)
# EOF
```

1 / 4: Datei index.html

```
<!DOCTYPE html>
<html>
   <head>
      <title>Titel</title>
      <meta charset="UTF-8" />
   </head>
   <body>
      <p>Stellen Sie eine Anfrage an den Testserver: <a href="greeting">Anfrage</a></p>
   </body>
</html>
```

Anlagen Teil 2

2 / 1: Datei server.py

```
#coding: utf-8

import sys
import os
import cherrypy
from app import application

#--------------------------------------
def main():
#--------------------------------------
   # Get current directory
   try:
       current_dir = os.path.dirname(os.path.abspath(__file__))
   except:
       current_dir = os.path.dirname(os.path.abspath(sys.executable))
   # disable autoreload
   cherrypy.engine.autoreload.unsubscribe()
   # Static content config
   static_config = {
      '/': {
         'tools.staticdir.root': current_dir,
         'tools.staticdir.on': True,
         'tools.staticdir.dir': './content'
      }
   }
   # Mount static content handler
   cherrypy.tree.mount(application.Application_cl(), '/', static_config)
   # suppress traceback-info
   cherrypy.config.update({'request.show_tracebacks': False})
   # Start server
   cherrypy.engine.start()
   cherrypy.engine.block()
#--------------------------------------
if __name__ == '__main__':
#--------------------------------------
   main()
# EOF
```

2 / 2: Datei application.py

```
# coding: utf-8
import cherrypy
from .database import Database_cl
from .view import View_cl

#-------------------------------------------------------
class Application_cl(object):
#-------------------------------------------------------

   #-----------------------------------------------------
   def __init__(self):
   #-----------------------------------------------------
```

```
13          self.db_o = Database_cl()
14          self.view_o = View_cl()
15
16       @cherrypy.expose
17       #-----------------------------------------------------
18       def index(self):
19       #-----------------------------------------------------
20          return self.createList_p()
21
22       @cherrypy.expose
23       #-----------------------------------------------------
24       def add(self):
25       #-----------------------------------------------------
26          return self.createForm_p()
27
28       @cherrypy.expose
29       #-----------------------------------------------------
30       def edit(self, id_spl):
31       #-----------------------------------------------------
32          return self.createForm_p(id_spl)
33
34       @cherrypy.expose
35       #-----------------------------------------------------
36       def save(self, id_spa, name1_spa, vorname1_spa, matrnr1_spa):
37       #-----------------------------------------------------
38          id_s = id_spa
39          data_a = [ name1_spa, vorname1_spa, matrnr1_spa ]
40          if id_s != "None":
41             self.db_o.update_px(id_s, data_a)
42          else:
43             self.db_o.create_px(data_a)
44          return self.createList_p()
45
46       @cherrypy.expose
47       #-----------------------------------------------------
48       def default(self, *arguments, **kwargs):
49       #-----------------------------------------------------
50          msg_s = "unbekannte Anforderung: " + \
51          str(arguments) + \
52          ' ' + \
53          str(kwargs)
54          raise cherrypy.HTTPError(404, msg_s)
55       default.exposed= True
56
57       #-----------------------------------------------------
58       def createList_p(self):
59       #-----------------------------------------------------
60          data_o = self.db_o.read_px()
61          return self.view_o.createList_px(data_o)
62
63       #-----------------------------------------------------
64       def createForm_p(self, id_spl = None):
65       #-----------------------------------------------------
66          if id_spl != None:
67             data_o = self.db_o.read_px(id_spl)
68          else:
69             data_o = self.db_o.getDefault_px()
70          return self.view_o.createForm_px(id_spl, data_o)
71    # EOF
```

2 / 3: Datei database.py

```
1   # coding: utf-8
2
3   import os
4   import os.path
5   import codecs
6   import json
7
8   from . import dataid
9
10  #-------------------------------------------------------
11  class Database_cl(object):
12  #-------------------------------------------------------
13
14     #-------------------------------------------------------
15     def __init__(self):
16     #-------------------------------------------------------
17        self.data_o = None
18        self.maxId_o = dataid.DataId_cl()
19        self.readData_p()
20
21     #-------------------------------------------------------
22     def create_px(self, data_opl):
23     #-------------------------------------------------------
24        id_s = self.maxId_o.create_px()
25        self.data_o[str(id_s)] = data_opl
26        self.saveData_p()
27        return str(id_s)
28
29     #-------------------------------------------------------
30     def read_px(self, id_spl = None):
31     #-------------------------------------------------------
32        data_o = None
33        if id_spl == None:
34           data_o = self.data_o
35        else:
36           if id_spl in self.data_o:
37                 data_o = self.data_o[id_spl]
38        return data_o
39
40     #-------------------------------------------------------
41     def update_px(self, id_spl, data_opl):
42     #-------------------------------------------------------
43        status_b = False
44        if id_spl in self.data_o:
45           self.data_o[id_spl] = data_opl
46           self.saveData_p()
47           status_b = True
48        return status_b
49
50     #-------------------------------------------------------
51     def delete_px(self, id_spl):
52     #-------------------------------------------------------
53        status_b = False
54        if self.data_o.pop(id_spl, None) != None:
55           self.saveData_p()
56           status_b = True
57        return status_b
```

```
58
59       #-----------------------------------------------------
60       def getDefault_px(self):
61       #-----------------------------------------------------
62          return ['', '', ''] # hier später Ergänzung!
63
64       #-----------------------------------------------------
65       def readData_p(self):
66       #-----------------------------------------------------
67          try:
68             fp_o = codecs.open(os.path.join('data', 'webteams.json'), 'r', 'utf-8')
69          except:
70             self.data_o = {}
71             self.saveData_p()
72          else:
73             with fp_o:
74                self.data_o = json.load(fp_o)
75          return
76
77       #-----------------------------------------------------
78       def saveData_p(self):
79       #-----------------------------------------------------
80          with codecs.open(os.path.join('data', 'webteams.json'), 'w', 'utf-8') as fp_o:
81             json.dump(self.data_o, fp_o, indent=3)
82
83    # EOF
```

2 / 4: Datei dataid.py

```
1     # coding: utf-8
2
3     import os
4     import os.path
5     import codecs
6     import json
7
8     #-----------------------------------------------------
9     class DataId_cl(object):
10    #-----------------------------------------------------
11
12       #-----------------------------------------------------
13       def __init__(self):
14       #-----------------------------------------------------
15          self.maxId_i = 0
16          self.readMaxId_p()
17
18       #-----------------------------------------------------
19       def create_px(self):
20       #-----------------------------------------------------
21          self.maxId_i += 1
22          self.saveMaxId_p()
23          return str(self.maxId_i)
24
25       #-----------------------------------------------------
26       def read_px(self):
27       #-----------------------------------------------------
28          return str(self.maxId_i)
29
```

```
30        #-----------------------------------------------------
31        def readMaxId_p(self):
32        #-----------------------------------------------------
33           try:
34              fp_o = codecs.open(os.path.join('data', 'maxid.json'), 'r', 'utf-8')
35           except:
36              self.maxId_i = 0
37              self.saveMaxId_p()
38           else:
39              with fp_o:
40                 self.maxId_i = json.load(fp_o)
41           return
42
43        #-----------------------------------------------------
44        def saveMaxId_p(self):
45        #-----------------------------------------------------
46           with codecs.open(os.path.join('data', 'maxid.json'), 'w', 'utf-8') as fp_o:
47              json.dump(self.maxId_i, fp_o)
48     # EOF
```

2 / 5: Datei view.py

```
1     # coding: utf-8
2
3     import codecs
4     import os.path
5     import string
6
7     from mako.template import Template
8     from mako.lookup import TemplateLookup
9
10    #-----------------------------------------------------
11    class View_cl(object):
12    #-----------------------------------------------------
13
14       #-----------------------------------------------------
15       def __init__(self):
16       #-----------------------------------------------------
17          self.lookup_o = TemplateLookup('./templates')
18
19       #-----------------------------------------------------
20       def createList_px(self, data_opl):
21       #-----------------------------------------------------
22          template_o = self.lookup_o.get_template('list.tpl')
23          markup_s = template_o.render(data_o = data_opl)
24          return markup_s
25
26       #-----------------------------------------------------
27       def createForm_px(self, id_spl, data_opl):
28       #-----------------------------------------------------
29          template_o = self.lookup_o.get_template('form.tpl')
30          markup_s = template_o.render(data_o = data_opl, key_s = id_spl)
31          return markup_s
32
33       #-----------------------------------------------------
34       def readFile_p(self, fileName_spl):
35       #-----------------------------------------------------
36          content_s = ''
```

```
37          with codecs.open(os.path.join('templates', fileName_spl), 'r', 'utf-8') as fp_o:
38             content_s = fp_o.read()
39          return content_s
40      # EOF
```

2 / 6: Vorlagen

Datei list.tpl

```
1   ## coding: utf-8
2   <!DOCTYPE html>
3   <html>
4      <head>
5         <title>Web-Teams</title>
6         <meta charset="UTF-8" />
7      </head>
8      <body>
9         <table>
10           <tr>
11              <th>Name (1)</th><th>Vorname (1)</th><th>Matr.-Nr. (1)</th><th>Aktion</th>
12           </tr>
13           % for key_s in data_o:
14           <tr>
15              <td>${data_o[key_s][0]}</td>
16              <td>${data_o[key_s][1]}</td>
17              <td>${data_o[key_s][2]}</td>
18              <td><a href="/edit/${key_s}">bearbeiten</a></td>
19           </tr>
20           % endfor
21        </table>
22        <div>
23           <a href="/add">erfassen</a>
24        </div>
25     </body>
26  </html>
```

Datei form.tpl

```
1   ## coding: utf-8
2   <!DOCTYPE html>
3   <html>
4   <head>
5      <title>Web-Teams</title>
6      <meta charset="UTF-8" />
7   </head>
8   <body>
9      <form id="idWTForm" action="/save" method="POST">
10        <input type="hidden" value="${key_s}" id="id_spa" name="id_spa" />
11        <div>
12           <label for="name1_spa">1. Name</label>
13           <input type="text"
14                  value="${data_o[0]}"
15                  id="name1_spa"
16                  name="name1_spa" required />
17        </div>
18        <div>
19           <label for="vorname1_spa">1. Vorname</label>
20           <input type="text"
```

```
                    value="${data_o[1]}"
                    id="vorname1_spa"
                    name="vorname1_spa" required />
            </div>
            <div>
               <label for="matrnr1_spa">1. Matrikelnummer</label>
               <input type="number"
                      value="${data_o[2]}"
                      id="matrnr1_spa"
                      name="matrnr1_spa" required />
            </div>
            <div>
               <input type="submit" value="Speichern"/>
            </div>
         </form>
      </body>
      </html>
```

Datei list2.tpl

(Darstellung der Liste als Aufzählung)

```
      ## coding: utf-8
      <!DOCTYPE html>
      <html>
         <head>
            <title>Web-Teams</title>
            <meta charset="UTF-8" />
            <script type="text/javascript" src="/webteams.js"></script>
         </head>
         <body>
            <%
               nr_i = 0
            %>
            <ul>
               % for key_s in data_o:
               <%
                  nr_i += 1
               %>
               <li>Team ${nr_i}:
                  <a href="/edit/${key_s}/${listform0}/">bearbeiten</a>
                  <!-- hier müssen Sie den "Schalter" für das Löschen ergänzen -->
                  <ul>
                     <li>${data_o[key_s][0]}, ${data_o[key_s][1]}, ${data_o[key_s][2]}</li>
                     <!-- hier müssen Sie die Angaben für das 2. Team-Mitglied ergänzen -->
                  </ul>
               </li>
               % endfor
            </ul>
            <div>
               <a href="/add/${listform0}/">erfassen</a>
            </div>
            <div>
               <a href="/?listform=${listform}">Als ${listformText} darstellen</a>
            </div>
         </body>
      </html>
```

2 / 7: Datei webteams.js

```
function confirmDelete_p (event_opl) {
   if ((event_opl.target.tagName.toLowerCase() == 'a'       ) &&
       (event_opl.target.className             == "clDelete")    ) {
      // Klick auf Link zum Löschen

      // Ihre Ergänzung
   }
}
window.onload = function () {
   let body_o = document.getElementsByTagName('body')[0];
   body_o.addEventListener('click', confirmDelete_p, false);
}
```