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 3.4 oder höher verwenden!
  - `http://www.python.org/downloads` ("looking for a specific release?"")
- Entwicklungsumgebung "WingIde 101"
  - lizenzkostenfrei erhältlich bei `http://www.wingide.com/downloads/wingide-101`, Version 5 verwenden!
- Quelltexteditor und Debugger für die Programmiersprache Python Quelltexteditor, lizenzkostenfreie / ohne Kosten verwendbare Beispiele:
  - für MS-Windows-Nutzer: notepad++, Sublime 2 oder 3, SynWrite, Atom, VS Code
  - für Linux / Mac OS -Nutzer: SciTe oder Sublime 2, Atom, VS Code
    - Hinweis: Sublime 2 oder 3 kann kostenfrei benutzt werden, es erscheint dann von Zeit zu Zeit ein Hinweis
- Webbrowser FireFox
  - installieren Sie die Erweiterung "Web Developer Toolbar"
- 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. Zur Vereinfachung können die Daten von maximal 15 Teams gespeichert werden.

Die Webanwendung besteht aus den beiden folgenden Seiten:

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

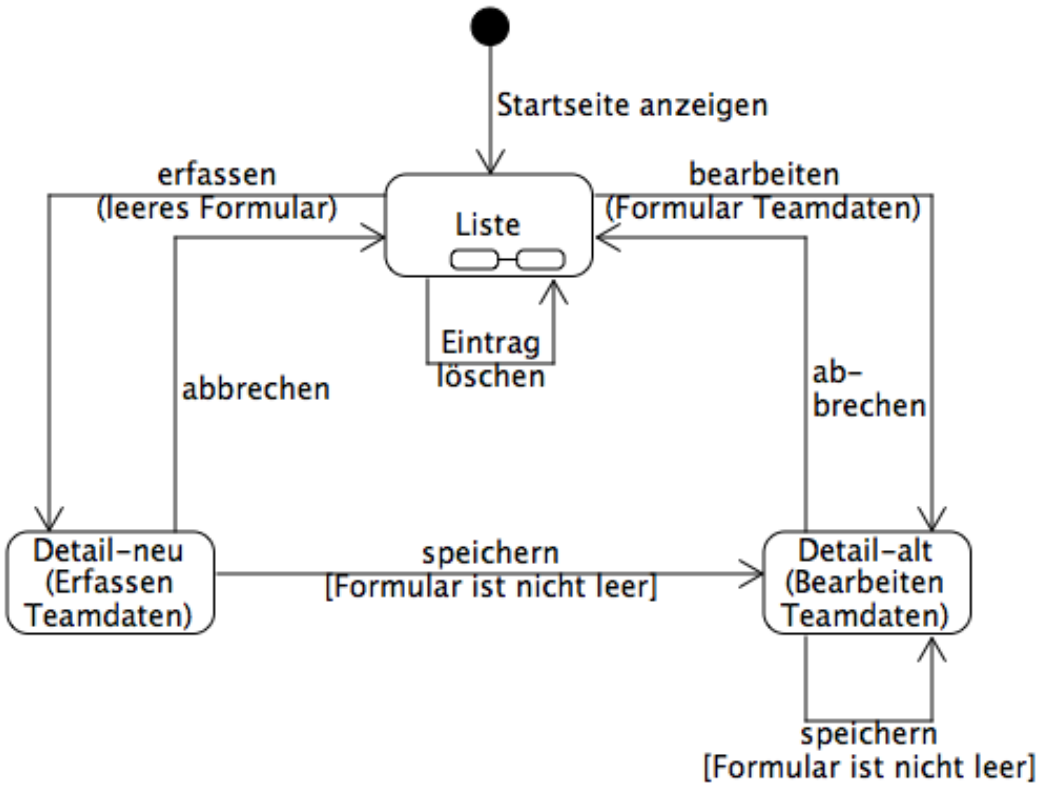

Abbildung 1: Zustandsmodell Webanwendung

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.

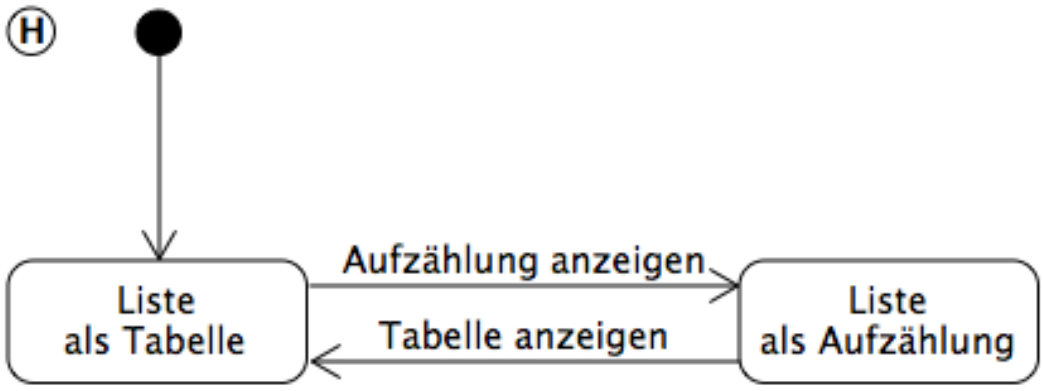

Abbildung 2: Verfeinerung Zustand Liste

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
                /template
```

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
- `view.py` (Inhalt: siehe Anlage 2 / 4)
  - 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 / 6).

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

- die Vorlagen für das Markup der Liste und des Formulars (Inhalt: siehe Anlage 2 / 5).

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 "WingIde", 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" 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 folgende Punkte:

- berücksichtigen Sie die Dateneingabe für die Daten des 2. Team-Mitglieds im Formular (siehe Vorlagen)
- berücksichtigen Sie das zusätzliche Attribut "Semesteranzahl" in allen Listen und Formularen sowie in der Datenbasis
- implementieren Sie die Aktion "Abbrechen" im Formular (siehe Vorlagen)

- gestalten Sie die beiden Seiten mit Hilfe von CSS: tragen Sie die CSS-Stilregeln dazu in die Datei `webteams.css` ein
- implementieren Sie das Löschen der Daten eines Teams auf der Serverseite (siehe Datei `database.py`)
- vervollständigen Sie die Rückfrage an den Benutzer beim Löschen von Einträgen in der Liste (Datei `webteams.js`)
- 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).

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
  - Varianten des Webclient (die beiden Listenformen, Formular)
  - Aufbau des Webservers
- 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" können 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

```python
# 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

```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 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'
      }
   }
   # 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
```

2 / 2: Datei application.py

```
# coding: utf-8

import cherrypy

from .database import Database_cl
from .view import View_cl

#-------------------------------------------------------
```

```
9  class Application_cl(object):
10 #-----------------------------------------------------------
11
12    #-----------------------------------------------------------
13    def __init__(self):
14    #-----------------------------------------------------------
15       # spezielle Initialisierung können hier eingetragen werden
16       self.db_o = Database_cl()
17       self.view_o = View_cl()
18
19    @cherrypy.expose
20    #-----------------------------------------------------------
21    def index(self):
22    #-----------------------------------------------------------
23       return self.createList_p()
24
25    @cherrypy.expose
26    #-----------------------------------------------------------
27    def add(self):
28    #-----------------------------------------------------------
29       return self.createForm_p()
30
31    @cherrypy.expose
32    #-----------------------------------------------------------
33    def edit(self, id):
34    #-----------------------------------------------------------
35       return self.createForm_p(id)
36
37    @cherrypy.expose
38    #-----------------------------------------------------------
39    def save(self, **data_opl):
40    #-----------------------------------------------------------
41       # Sichern der Daten: aufgrund der Formularbearbeitung muss
42       # eine vollständige HTML-Seite zurückgeliefert werden!
43
44       # data_opl: Dictionary mit den gelieferten key-value-Paaren
45
46       # hier müsste man prüfen, ob die Daten korrekt vorliegen!
47
48       # HIER müssen Sie die Semesterzahl(en) ergänzen
49
50       id_s = data_opl["id_s"]
51       data_a = [ data_opl["name1_s"]
52       ,   data_opl["vorname1_s"]
53       ,   data_opl["matrnr1_s"]
54       ,   data_opl["name2_s"]
55       ,   data_opl["vorname2_s"]
56       ,   data_opl["matrnr2_s"]
57       ]
58       if id_s != "None":
59          # Update-Operation
60          self.db_o.update_px(id_s, data_a)
61       else:
62          # Create-Operation
63          id_s = self.db_o.create_px(data_a)
64
65       return self.createForm_p(id_s)
```

```
66
67    @cherrypy.expose
68    #-----------------------------------------------------
69    def delete(self, id):
70    #-----------------------------------------------------
71       # Eintrag löschen, dann Liste neu anzeigen
72
73       self.db_o.delete_px(id)
74       return self.createList_p()
75
76    @cherrypy.expose
77    #-----------------------------------------------------
78    def default(self, *arguments, **kwargs):
79    #-----------------------------------------------------
80       msg_s = "unbekannte Anforderung: " + \
81               str(arguments) + \
82               ' ' + \
83               str(kwargs)
84       raise cherrypy.HTTPError(404, msg_s)
85    default.exposed= True
86
87    #-----------------------------------------------------
88    def createList_p(self):
89    #-----------------------------------------------------
90       data_o = self.db_o.read_px()
91       # mit diesen Daten Markup erzeugen
92       return self.view_o.createList_px(data_o)
93
94    #-----------------------------------------------------
95    def createForm_p(self, id_spl = None):
96    #-----------------------------------------------------
97       if id_spl != None:
98          data_o = self.db_o.read_px(id_spl)
99       else:
100         data_o = self.db_o.getDefault_px()
101      # mit diesen Daten Markup erzeugen
102      return self.view_o.createForm_px(id_spl, data_o)
103
104 # EOF
```

2 / 3: Datei database.py

```
1  # coding: utf-8
2
3  import os
4  import os.path
5  import codecs
6
7  import json
8
9  #-----------------------------------------------------
10 class Database_cl(object):
11 #-----------------------------------------------------
12
13    # da es hier nur darum geht, die Daten dauerhaft zu speichern,
```

```
14    # wird ein sehr einfacher Ansatz verwendet:
15    # - es können Daten zu genau 15 Teams gespeichert werden
16    # - je Team werden 2 Teilnehmer mit Namen, Vornamen und Matrikelnummer
17    #   berücksichtigt
18    # - die Daten werden als eine JSON-Datei abgelegt
19
20    #-----------------------------------------------------
21    def __init__(self):
22    #-----------------------------------------------------
23       self.data_o = None
24       self.readData_p()
25
26    #-----------------------------------------------------
27    def create_px(self, data_opl):
28    #-----------------------------------------------------
29       # Überprüfung der Daten müsste ergänzt werden!
30
31       # 'freien' Platz suchen,
32       # falls vorhanden: belegen und Nummer des Platzes als Id zurückgeben
33
34       id_s = None
35       for loop_i in range(0,15):
36          if self.data_o[str(loop_i)][0] == '':
37             id_s = str(loop_i)
38             self.data_o[id_s] = data_opl
39             self.saveData_p()
40             break
41
42       return id_s
43
44    #-----------------------------------------------------
45    def read_px(self, id_spl = None):
46    #-----------------------------------------------------
47       # hier zur Vereinfachung:
48       # Aufruf ohne id: alle Einträge liefern
49       data_o = None
50       if id_spl == None:
51          data_o = self.data_o
52       else:
53          if id_spl in self.data_o:
54             data_o = self.data_o[id_spl]
55
56       return data_o
57
58    #-----------------------------------------------------
59    def update_px(self, id_spl, data_opl):
60    #-----------------------------------------------------
61       # Überprüfung der Daten müsste ergänzt werden!
62       status_b = False
63       if id_spl in self.data_o:
64          self.data_o[id_spl] = data_opl
65          self.saveData_p()
66          status_b = True
67
68       return status_b
69
70    #-----------------------------------------------------
```

```
71    def delete_px(self, id_spl):
72    #-------------------------------------------------------
73       status_b = False
74       if id_spl in self.data_o:
75          pass
76          # hier müssen Sie den Code ergänzen
77          # Löschen als Zurücksetzen auf die Default-Werte implementieren
78
79          # Ihre Ergänzung
80
81       return status_b
82
83    #-------------------------------------------------------
84    def getDefault_px(self):
85    #-------------------------------------------------------
86       return ['', '', '', '', '', ''] # HIER müssen Sie eine Ergänzung vornehmen
87
88    #-------------------------------------------------------
89    def readData_p(self):
90    #-------------------------------------------------------
91       try:
92          fp_o = codecs.open(os.path.join('data', 'webteams.json'), 'r', 'utf-8')
93       except:
94          # Datei neu anlegen
95          self.data_o = {}
96          for loop_i in range(0,15):
97             self.data_o[str(loop_i)] = ['', '', '', '', '', ''] # HIER müssen Sie eine Ergänzung vornehmen
98             self.saveData_p()
99       else:
100         with fp_o:
101            self.data_o = json.load(fp_o)
102
103      return
104
105   #-------------------------------------------------------
106   def saveData_p(self):
107   #-------------------------------------------------------
108      with codecs.open(os.path.join('data', 'webteams.json'), 'w', 'utf-8') as fp_o:
109         json.dump(self.data_o, fp_o)
110
111# EOF
```

2 / 4: Datei view.py

```
1  # coding: utf-8
2
3  # sehr einfache Erzeugung des Markups für vollständige Seiten
4  # jeweils 3 Abschnitte:
5  # - begin
6  # - content
7  # - end
8
9  # bei der Liste wird der content-Abschnitt wiederholt
10 # beim Formular nicht
11
```

```
12 import codecs
13 import os.path
14 import string
15
16 #-------------------------------------------------------
17 class View_cl(object):
18 #-------------------------------------------------------
19
20    #-------------------------------------------------------
21    def __init__(self):
22    #-------------------------------------------------------
23       pass
24
25    #-------------------------------------------------------
26    def createList_px(self, data_opl):
27    #-------------------------------------------------------
28       # hier müsste noch eine Fehlerbehandlung ergänzt werden !
29       markup_s = ''
30       markup_s += self.readFile_p('list0.tpl')
31
32       markupV_s = self.readFile_p('list1.tpl')
33       lineT_o = string.Template(markupV_s)
34       # mehrfach nutzen, um die einzelnen Zeilen der Tabelle zu erzeugen
35       for loop_i in range(0,15):
36          data_a = data_opl[str(loop_i)]
37          markup_s += lineT_o.safe_substitute (name1_s=data_a[0] # HIER müssen Sie eine Ergänzung vornehmen
38          ,   vorname1_s=data_a[1]
39          ,   matrnr1_s=data_a[2]
40          ,   name2_s=data_a[3]
41          ,   vorname2_s=data_a[4]
42          ,   matrnr2_s=data_a[5]
43          ,   id_s=str(loop_i)
44          )
45
46       markup_s += self.readFile_p('list2.tpl')
47
48       return markup_s
49
50    #-------------------------------------------------------
51    def createForm_px(self, id_spl, data_opl):
52    #-------------------------------------------------------
53
54       # hier müsste noch eine Fehlerbehandlung ergänzt werden !
55       markup_s = ''
56       markup_s += self.readFile_p('form0.tpl')
57
58       markupV_s = self.readFile_p('form1.tpl')
59       lineT_o = string.Template(markupV_s)
60       markup_s += lineT_o.safe_substitute (name1_s=data_opl[0] # HIER müssen Sie eine Ergänzung vornehmen
61       ,   vorname1_s=data_opl[1]
62       ,   matrnr1_s=data_opl[2]
63       ,   name2_s=data_opl[3]
64       ,   vorname2_s=data_opl[4]
65       ,   matrnr2_s=data_opl[5]
66       ,   id_s=id_spl
67       )
68
```

```
69          markup_s += self.readFile_p('form2.tpl')
70
71          return markup_s
72
73       #--------------------------------------------------------
74       def readFile_p(self, fileName_spl):
75       #--------------------------------------------------------
76          content_s = ''
77          with codecs.open(os.path.join('template', fileName_spl), 'r', 'utf-8') as fp_o:
78             content_s = fp_o.read()
79
80          return content_s
81    # EOF
```

2 / 5: Vorlagen

list0.tpl

```
1  <!DOCTYPE html>
2  <html>
3     <head>
4        <title>Web-Teams</title>
5        <meta charset="UTF-8" />
6        <style>
7           @import "/webteams.css";
8        </style>
9        <script type="text/javascript" src="/webteams.js"></script>
10    </head>
11    <body>
12       <table> <!-- HIER müssen Sie eine Ergänzung vornehmen -->
13          <tr>
14             <th>Name (1)</th><th>Vorname (1)</th><th>Matr.-Nr. (1)</th>
15             <th>Name (2)</th><th>Vorname (2)</th><th>Matr.-Nr. (2)</th>
16             <th>Aktion</th>
17          </tr>
```

list1.tpl

```
1           <tr> <!-- HIER müssen Sie eine Ergänzung vornehmen -->
2              <td>$name1_s</td><td>$vorname1_s</td><td>$matrnr1_s</td>
3              <td>$name2_s</td><td>$vorname2_s</td><td>$matrnr2_s</td>
4              <td><a href="/edit/$id_s">bearbeiten</a> <a href="/delete/$id_s">löschen</a></td>
5           </tr>
```

list2.tpl

```
1        </table>
2        <div>
3           <a href="/add">erfassen</a>
4        </div>
```

```
   </body>
</html>
```

listaufz0.tpl

```
<!DOCTYPE html>
<html>
   <head>
      <title>Web-Teams</title>
      <meta charset="UTF-8" />
      <style>
         @import "/webteams.css";
      </style>
      <script type="text/javascript" src="/webteams.js"></script>
   </head>
   <body>
      <ul>
```

listaufz1.tpl

```
         <!-- Sie müssen Ergänzungen vornehmen -->
         <li>Gruppe $nr: <a href="/edit/$id_s">bearbeiten</a> <a href="/delete/$id_s">löschen</a>
             <ul>
                 <li>$name1_s, $vorname1_s, $matrnr1_s</li>
                 <li>$name2_s, $vorname2_s, $matrnr2_s</li>
             </ul>
         </li>
```

listaufz2.tpl

```
      </ul>
      <div>
         <a href="/add">erfassen</a>
      </div>
   </body>
</html>
```

form0.tpl

```
<!DOCTYPE html>
<html>
   <head>
      <title>Web-Teams</title>
      <meta charset="UTF-8" />
      <style>
         @import "/webteams.css";
      </style>
      <script type="text/javascript" src="/webteams.js"></script>
```

```
10    </head>
11    <body>
12       <form id="idWTForm" action="/save" method="POST">
```

form1.tpl

```
1           <input type="hidden" value="$id_s" id="id_s" name="id_s" />
2           <div>
3              <label for="name1_s">1. Name</label>
4              <input type="text" value="$name1_s" id="name1_s" name="name1_s" required />
5           </div>
6           <div>
7              <label for="vorname1_s">1. Vorname</label>
8              <input type="text" value="$vorname1_s" id="vorname1_s" name="vorname1_s" required />
9           </div>
10          <div>
11             <label for="matrnr1_s">1. Matrikelnummer</label>
12             <input type="text" value="$matrnr1_s" id="matrnr1_s" name="matrnr1_s" required />
13          </div>
14          <!-- Ihre Ergänzung -->
```

form2.tpl

```
1           <div>
2              <input type="submit" value="Speichern" /><!-- Ihre Ergänzung -->
3           </div>
4        </form>
5     </body>
6  </html>
```

2 / 6: Datei webteams.js

```
1  function confirmDelete_p (event_opl) {
2     if ((event_opl.target.tagName.toLowerCase() == 'a') && (event_opl.target.className == "clDelete")) {
3        // Klick auf Link zum Löschen
4
5        // Ihre Ergänzung
6     }
7  }
8  window.onload = function () {
9     let body_o = document.getElementsByTagName('body')[0];
10    body_o.addEventListener('click', confirmDelete_p, false);
11 }
```