

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.

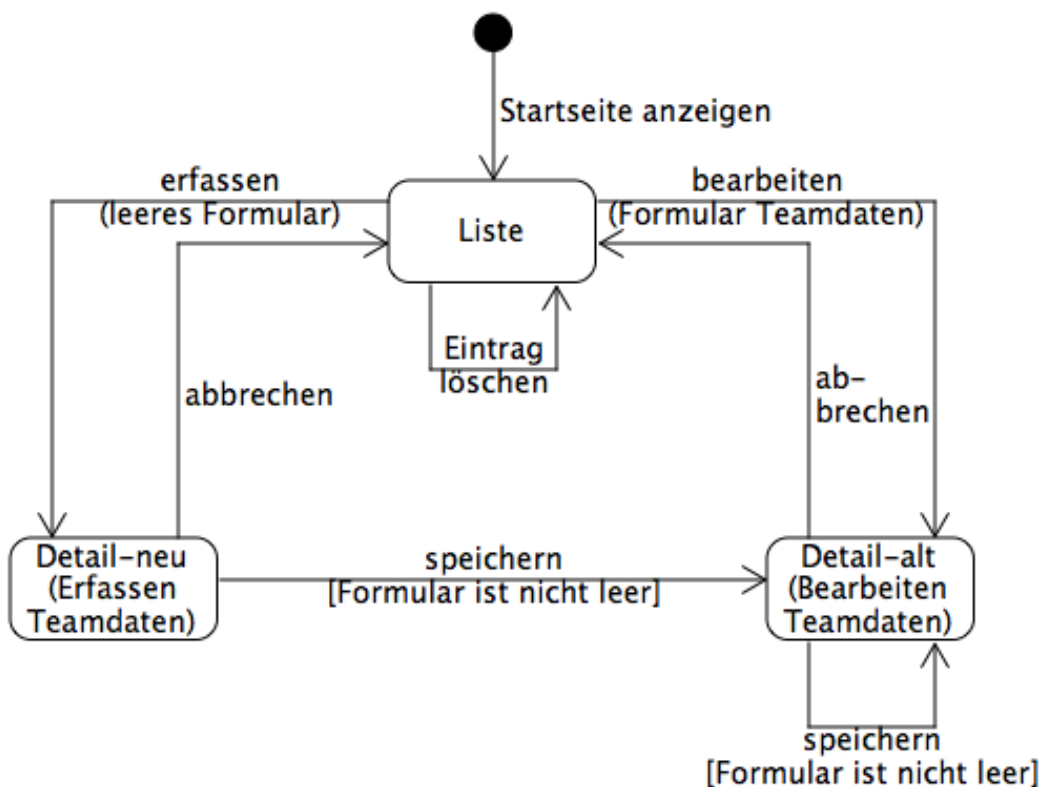


Abbildung 1: Zustandsmodell Webanwendung

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

Schritt 2.1: Verzeichnisse und Dateien erstellen

Ergänzen Sie die in Teil 1 eingerichtete Verzeichnisstruktur:

```
web
  /p1
    /webteams
      /app
      /content
      /data
      /doc
```

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 Vorlagen für das Markup der Liste und des Formulars (Inhalt: siehe Anlage 2 / 5)
- die zunächst leere Datei `webteams.css`
- die Datei `webteams.js` (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 "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)
- 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`).

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.

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

```
1 #coding: utf-8
2 import os
3 import cherrypy
4 from app import application
5
6 #-----
7 def main():
8     #-----
9     # Get current directory
10    try:
11        current_dir = os.path.dirname(os.path.abspath(__file__))
12    except:
13        current_dir = os.path.dirname(os.path.abspath(sys.executable))
14    # disable autoreload and timeout_monitor
15    cherrypy.engine.autoreload.unsubscribe()
16    cherrypy.engine.timeout_monitor.unsubscribe()
17    # Static content config
18    static_config = {
19        '/': {
20            'tools.staticdir.root': current_dir,
21            'tools.staticdir.on': True,
22            'tools.staticdir.dir': './content',
23            'tools.staticdir.index': 'index.html'
24        }
25    }
26    # Mount static content handler
27    root_o = cherrypy.tree.mount(application.Application_cl(), '/', static_config)
28    # suppress traceback-info
29    cherrypy.config.update({'request.show_tracebacks': False})
30    # Start server
31    cherrypy.engine.start()
32    cherrypy.engine.block()
33
34 #-----
35 if __name__ == '__main__':
36     #-----
37     main()
38 # EOF
```

1 / 2: Datei __init__.py

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

1 / 3: Datei application.py

```
1 # coding: utf-8
2
3 import cherrypy
4
5 #-----
6 class Application_cl(object):
7 #-----
8     #-----
9     def __init__(self):
10 #-----
11         # constructor
12         pass
13
14 @cherrypy.expose
15 #-----
16 def greeting(self):
17 #-----
18     return "CherryPy-Server, Version %s" % cherrypy.__version__
19
20 @cherrypy.expose
21 #-----
22 def default(self, *arglist, **kwargs):
23 #-----
24     msg_s = "unbekannte Anforderung: " + \
25         str(arglist) + \
26         ' '+\
27         str(kwargs)
28     raise cherrypy.HTTPError(404, msg_s)
29 # EOF
```

1 / 4: Datei index.html

```
1 <!DOCTYPE html>
2 <html>
3     <head>
4         <title>Titel</title>
5         <meta charset="UTF-8" />
6     </head>
7     <body>
8         <p>Stellen Sie eine Anfrage an den Testserver: <a href="greeting">Anfrage</a></p>
9     </body>
10 </html>
```

Anlagen Teil 2

2 / 1: Datei server.py

```
1 #coding: utf-8
2 import os
3 import cherrypy
4 from app import application
5
6 #-----
7 def main():
8     #-----
9     # Get current directory
10    try:
11        current_dir = os.path.dirname(os.path.abspath(__file__))
12    except:
13        current_dir = os.path.dirname(os.path.abspath(sys.executable))
14    # disable autoreload and timeout_monitor
15    cherrypy.engine.autoreload.unsubscribe()
16    cherrypy.engine.timeout_monitor.unsubscribe()
17    # Static content config
18    static_config = {
19        '/': {
20            'tools.staticdir.root': current_dir,
21            'tools.staticdir.on': True,
22            'tools.staticdir.dir': './content'
23        }
24    }
25    # Mount static content handler
26    root_o = cherrypy.tree.mount(application.Application_cl(), '/', static_config)
27    # suppress traceback-info
28    cherrypy.config.update({'request.show_tracebacks': False})
29    # Start server
30    cherrypy.engine.start()
31    cherrypy.engine.block()
32
33 #-----
34 if __name__ == '__main__':
35     #-----
36     main()
37 # EOF
```

2 / 2: Datei application.py

```
1 # coding: utf-8
2
3 import cherrypy
4
5 from .database import Database_cl
6 from .view import View_cl
7
8 #-----
```

```
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 @cherry.py.expose
20 #-----
21 def index(self):
22 #-----
23     return self.createList_p()
24
25 @cherry.py.expose
26 #-----
27 def add(self):
28 #-----
29     return self.createForm_p()
30
31 @cherry.py.expose
32 #-----
33 def edit(self, id):
34 #-----
35     return self.createForm_p(id)
36
37 @cherry.py.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     id_s = data_opl["id_s"]
49     data_a = [ data_opl["name1_s"]
50 , data_opl["vorname1_s"]
51 , data_opl["matrnr1_s"]
52 , data_opl["name2_s"]
53 , data_opl["vorname2_s"]
54 , data_opl["matrnr2_s"]
55 ]
56     if id_s != "None":
57         # Update-Operation
58         self.db_o.update_px(id_s, data_a)
59     else:
60         # Create-Operation
61         id_s = self.db_o.create_px(data_a)
62
63     return self.createForm_p(id_s)
64
65 @cherry.py.expose
```



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

```
16 # - es können Daten zu genau 15 Teams gespeichert werden
17 # - je Team werden 2 Teilnehmer mit Namen, Vornamen und Matrikelnummer
18 #   berücksichtigt
19 # - die Daten werden als eine JSON-Datei abgelegt
20
21 #-----
22 def __init__(self):
23 #-----
24     self.data_o = None
25     self.readData_p()
26
27 #-----
28 def create_px(self, data_opl):
29 #-----
30     # Überprüfung der Daten müsste ergänzt werden!
31
32     # 'freien' Platz suchen,
33     # falls vorhanden: belegen und Nummer des Platzes als Id zurückgeben
34
35     id_s = None
36     for loop_i in range(0,15):
37         if self.data_o[str(loop_i)][0] == '':
38             id_s = str(loop_i)
39             self.data_o[id_s] = data_opl
40             self.saveData_p()
41             break
42
43     return id_s
44
45 #-----
46 def read_px(self, id_spl = None):
47 #-----
48     # hier zur Vereinfachung:
49     # Aufruf ohne id: alle Einträge liefern
50     data_o = None
51     if id_spl == None:
52         data_o = self.data_o
53     else:
54         if id_spl in self.data_o:
55             data_o = self.data_o[id_spl]
56
57     return data_o
58
59 #-----
60 def update_px(self, id_spl, data_opl):
61 #-----
62     # Überprüfung der Daten müsste ergänzt werden!
63     status_b = False
64     if id_spl in self.data_o:
65         self.data_o[id_spl] = data_opl
66         self.saveData_p()
67         status_b = True
68
69     return status_b
70
71 #-----
72 def delete_px(self, id_spl):
```

```
73 #-----
74     status_b = False
75     if id_spl in self.data_o:
76         pass
77         # hier müssen Sie den Code ergänzen
78         # Löschen als Zurücksetzen auf die Default-Werte implementieren
79
80         # Ihre Ergänzung
81
82     return status_b
83
84 #-----
85 def getDefault_px(self):
86 #-----
87     return ['', '', '', '', '', '']
88
89 #-----
90 def readData_p(self):
91 #-----
92     try:
93         fp_o = codecs.open(os.path.join('data', 'webteams.json'), 'r', 'utf-8')
94     except:
95         # Datei neu anlegen
96         self.data_o = {}
97         for loop_i in range(0,15):
98             self.data_o[str(loop_i)] = ['', '', '', '', '', '']
99             self.saveData_p()
100     else:
101         with fp_o:
102             self.data_o = json.load(fp_o)
103
104     return
105
106 #-----
107 def saveData_p(self):
108 #-----
109     with codecs.open(os.path.join('data', 'webteams.json'), 'w', 'utf-8') as fp_o:
110         json.dump(self.data_o, fp_o)
111
112 # 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]
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]
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>
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>
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/${id_s}">bearbeiten</a>&nbsp;<a href="/delete/${id_s}">löschen</a></td>
5         </tr>
```

list2.tpl

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

```
5     </body>
6 </html>
```

form0.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    <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_op1) {
2   if ((event_op1.target.tagName.toLowerCase() == 'a') && (event_op1.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 }
```