Framework

by

Dr. Günter Kolousek

Framework

- Framework vs. Library?
  - Library bietet Klassen, Funktion,… an, die von Applikationscode aufgerufen werden

Framework

- Framework vs. Library?
  - Library bietet Klassen, Funktion,… an, die von Applikationscode aufgerufen werden
  - Ein Framework ruft Applikationscode auf!
- Ein Framework
  - ist wie ein Text mit Leerstellen, die vom Programmierer befüllt werden
  - ist wie eine Grammatik, die mit eigenen Worten befüllt wird
  - hat auch gewisse Überschneidung mit Bibliotheken

Library

```
import sys
from framework import PersonForm
from form_utils import transform

form = PersonForm()
form.load()
form.fill()
form.validate()
form.data = transform(form.data)
form.save(sys.stdout)
```

Framework – 2

- Es gibt verschiedene Möglichkeiten ein Framework zu strukturieren
  - Callback Funktion
  - Vererbung mittels Subclassing
  - Interfaces
  - imperatives API zur Registrierung
  - *Convention over configuration*
  - …

Callback Funktion

```
from framework import Form

def person_save(data):
    # ...
    pass

form = Form(save=person_save)
Form.add_control(NameControl())
# ...
form.do()
```

- Funktionen müssen first-class Objekte sein
- Closure wären wünschenswert, um Konfiguration der Funktionen zu ermöglichen

Subclassing

```
from framework import Form

class MyPersonForm(Form):
    def __init__(self):
        add_control(NameControl())
        # ...
    def save(self, data):
        # ...
        pass

form = MyPersonForm()
form.do()
```

- einfach, aber Methoden müssen idR überschrieben werden
  - Komplexität wächst mit Anzahl der zu überschreibenden Methoden
- immer eigene Klasse nötig…

Interfaces

```
from framework import Form, IFormBackend

class PersonFormBackend(IFormBackend):
    def load(self):
        pass

    def save(self, data):
        pass

form = Form(PersonFormBackend())
```

- Ähnlich wie Funktionen, aber sinnvoll wenn *umfangreicherer* Vertrag notwendig

imperatives API zur Registrierung

```
from framework import form_save_registry

def save(data):
    # ...
    pass

# configure to use 'save' for the form named 'perso
form_save_registry.register('person_form', save)
# ...
person_form = Form()
```

- einfach, aber es nicht offensichtlich wie konfiguriert wurde
- Reihenfolge der Aufrufe kann von Relevanz sein
- Effekt von mehrfachen Registrieren des Gleichen

Convention over configuration

```
# prefix ist form_save_...
def form_save_person_form(data):
    # ...
    pass
```

- einfach zu verwenden
  - aber kompliziert, wenn viele verschiedene Regeln zu merken sind
- Implementierung schwierig, nur in dynamischen Programmiersprachen