Objekter i Flask

Aflevering i programmering B, november 2020

Rammer

- Opgaven laves individuelt.
- Der er seks timers fordybelsestid og fire timers undervisningstid til projektet.
- Dokumentation af opgaven inkl. et link til programmet afleveres som PDF.

Indhold

Der er følgende krav til programmet:

- Klasser og underklasser, attributter og metoder
- Brugergrænseflade i web med Flask
- Bygget op om Model-View-Controller
- Tilgås via link til pythonanywhere.com

Fremgangsmåde

Du skal arbejde dig frem efter følgende fremgangsmåde:

1. Beskrivelse af ide (kortfattet) og eventuelt skitse
2. Udarbejdelse af prioriteret liste af krav
3. Realisering af krav (kode)
4. Idriftsættelse på pythonanywhere.com

Dokumentation

Programmets dokumenation afleveres som en rapport på højst seks sider ekskl. forside, indholdsfortegnelse og bilag.

Der er følgende krav til indhold af rapporten:

- Forside
- Indholdsfortegnelse
- Link til programmet
- Prioriteret liste af krav
- Klassediagram

- Skærmbilleder af websider
- Forklaring af udvalgt kode for henholdsvis model, view og controller.
- Bilag med alt kode

Materiale

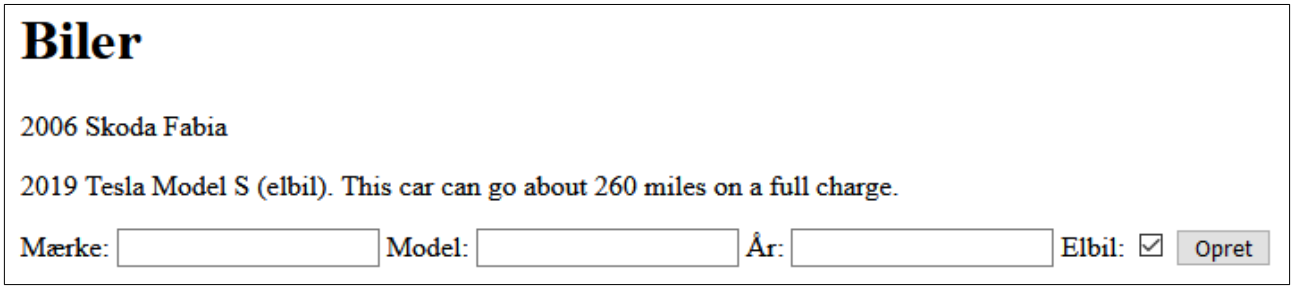
Ud over materiale tidligere oplyst i lektioner på Ludus kan følgende være hjælp i projektarbejdet:

- Introduktion til Jinja
- Nedenstående eksempel som inspiration

Eksempel

Følgende eksempel skal illustrere brug af nedarvning og designmønstret MVC i en webapplikation om biler og elbiler.

Websiden http://jesperbuch.pythonanywhere.com/ er et *view* på listen af biler og tilbyder brugeren at oprette en ny bil via en formular:



I MVC-designmønstret er *view* lavet i html og jinja, som kan ses i Bilag A.

Når brugeren klikker på knappen *Opret*, sendes de indtastede data med metoden POST til ruten */create*. Denne behandles med en *controller* action i metoden *create*:

```
@app.route('/create', methods = ['POST', 'GET'])
def create():
    if request.method == 'POST':
        make = request.form['make']
        model = request.form['model']
        year = request.form['year']
        if 'ecar' in request.form:
            cars.append(ElectricCar(make, model, int(year)))
        else:
            cars.append(Car(make, model, int(year)))
    return render_template('index.html', cars=cars)
```

I denne metode opdateres *modellen* med en ny bil. Denne vil være en elbil, hvis der er sat kryds i *Elbil*. Se hele koden for *controlleren* i Bilag B.

Bemærk at beskrivelsen af elbilen inkluderer informationer om batteriets rækkevidde. Det sker i modellen i underklassen ElectricCar, hvor metoden overlæses:

```
def get_descriptive_name(self):
    long_name = f"{self.year} {self.make} {self.model} (elbil). "
    long_name += self.battery.get_range()
    return long_name
```

Se hele koden for modulet *car* i Bilag C.

Bilag A

index.html

```
<html>
<head>
<link rel="stylesheet" href='/static/style.css' />
</head>
<body>
<h1>Biler</h1>
{% for car in cars: %}
<p>{{ car.get_descriptive_name() }}</p>
{% endfor %}
<form method="post" action="/create">
    Mærke: <input type="text" name="make"/>
    Model: <input type="text" name="model"/>
    År: <input type="text" name="year"/>
    Elbil: <input type="checkbox" checked name="ecar"/>
    <input type="submit" value="Opret"/>
</form>
</body>
</html>
```

Bilag B

app.py

```
from flask import Flask, render_template, request
from car import Car, ElectricCar

app = Flask(__name__)
cars = []

@app.route('/')
def index():
    cars.append(Car('Skoda', 'Fabia', 2006))
    cars.append(ElectricCar('Tesla', 'Model S', 2019))
    return render_template('index.html', cars=cars)

@app.route('/create', methods = ['POST', 'GET'])
def create():
    if request.method == 'POST':
        make = request.form['make']
        model = request.form['model']
        year = request.form['year']
        if 'ecar' in request.form:
            cars.append(ElectricCar(make, model, int(year)))
        else:
            cars.append(Car(make, model, int(year)))
    return render_template('index.html', cars=cars)

if __name__ == '__main__':
    app.run(debug=True, host='0.0.0.0')
```

Bilag C

car.py

```
''' This module includes classes Car and ElectricCar. '''

class Car:

    def __init__(self, make, model, year):
        self.make = make
        self.model = model
        self.year = year
        self.odometer_reading = 0

    def get_descriptive_name(self):
        long_name = f"{self.year} {self.make} {self.model}"
        return long_name.title()

    def read_odometer(self):
        print(f"This car has {self.odometer_reading} miles on it.")

    def update_odometer(self, mileage):
        if mileage >= self.odometer_reading:
            self.odometer_reading = mileage
        else:
            print("You can't roll back an odometer!")

    def increment_odometer(self, miles):
        self.odometer_reading += miles

class Battery:
    """A simple attempt to model a battery for an electric car."""
    def __init__(self, battery_size=75):
        """Initialize the battery's attributes."""
        self.battery_size = battery_size
    def describe_battery(self):
        """Print a statement describing the battery size."""
        return f"This car has a {self.battery_size}-kWh battery."

    def get_range(self):
        """Print a statement about the range this battery provides."""
        range = 0
        if self.battery_size == 75:
            range = 260
        elif self.battery_size == 100:
            range = 315
        return f"This car can go about {range} miles on a full charge."
```

```python
class ElectricCar(Car):
    """Represent aspects of a car, specific to electric vehicles."""

    def __init__(self, make, model, year):
        """Initialize attributes of the parent class."""
        super().__init__(make, model, year)
        self.battery = Battery()

    def get_descriptive_name(self):
        long_name = f"{self.year} {self.make} {self.model} (elbil). "
        long_name += self.battery.get_range()
        return long_name
```