Challenge: Flask

Prof. Dr. Markus Heckner

Web

- HTML, CSS und JavaScript zunehmend verwendet, um Anwendungen für Desktops, Laptops und mobile Geräte zu entwickeln

Webserver bis jetzt

Replit als statischer Webserver…

… kann keine Formulardaten verarbeiten und
somit nicht dynamisch auf Eingaben reagieren…

Heute

- SQL
- Python
- Flask

… Framework zur Entwicklung von Webanwendungen

```
http://www.example.com/
```

```
http://www.example.com/file.html
```

```
http://www.example.com/folder
```

```
http://www.example.com/folder/file.html
```

```
http://www.example.com/path
```

```
http://www.example.com/route
```

```
http://www.example.com/route?key=value
```

Flask kann HTTP-Anfragen wie diese verarbeiten…

```
GET /search?q=cats HTTP/1.1
Host: www.google.com
...
```

Flask

```
from flask import Flask
```

Struktur einer minimalen Flask-App

```
app.py
requirements.txt
static/
templates/
```

Flask – Hello world!

```
from flask import Flask, render_template

app = Flask(__name__)


@app.route("/")
def index():
    return render_template("index.html")
```


```
request.args
request.form
```


Selbst Eingaben erstellen: Formulare

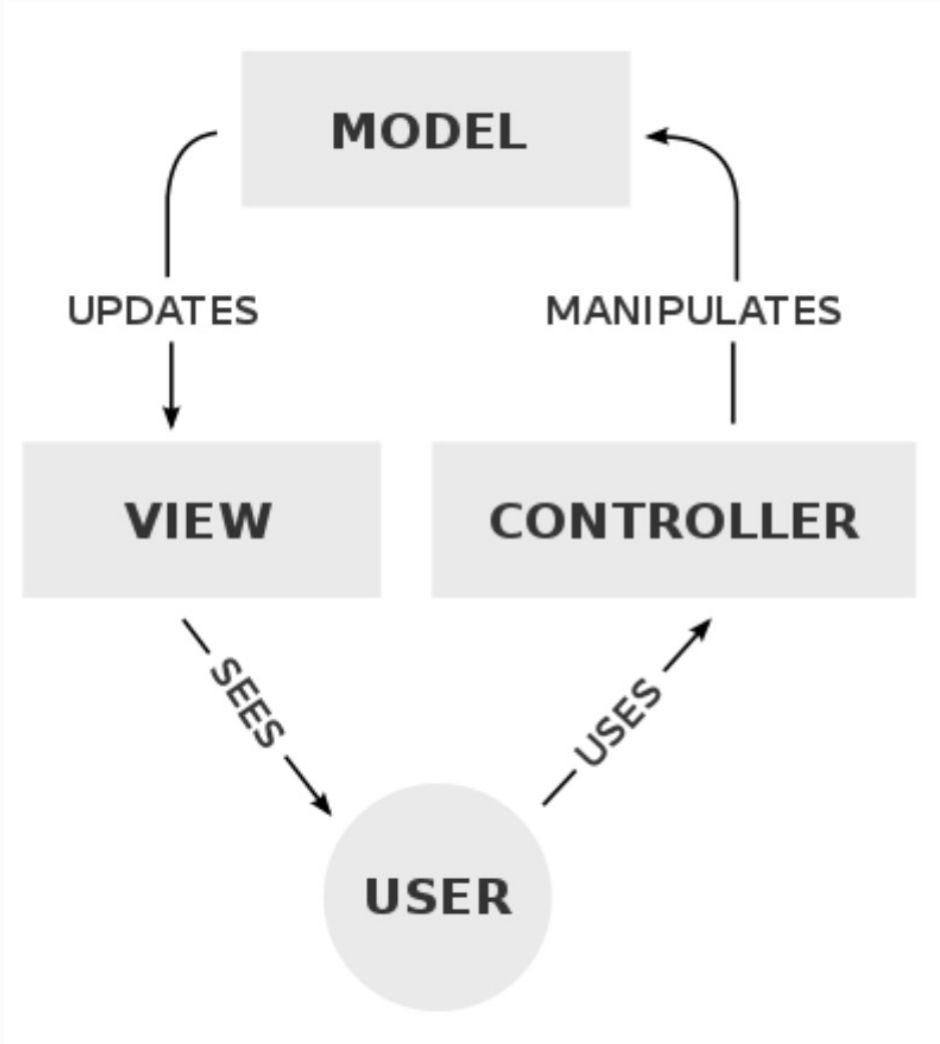
```
<form action="/greet" method="get">
  <input autocomplete="off" autofocus name="name" placeholder="Name" type="text">
  <input type="submit" value="Send">
</form>
```

Layouts

- Doppelter Code in den Templates
- Mit Flask Templates lassen sich Codebestandteile auslagern und in mehreren Dateien einbinden

POST

- GET übermittelt Daten als Teil der URL (unsicher)
- Formulare lassen sich auf POST umstellen – Daten werden als Inhalt des Requests geschick und können somit verschlüsselt werden (sicher)



Bildquelle: https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller

OTH-Sport

- Studierende können sich für Sportaktivitäten registrieren

oth_sport_1 bis oth_sport_5

Daten speichern

- Web-Applikationen speichern Daten typischerweise in Datenbanken

DEMO: oth_sport_6, oth_sport_7

Sessions

- HTTP ist *stateless*
- Webserver müssen sich Informationen über die Clients merken (z.B. Warenkorb, eingeloggt bleiben)
- Sessions ermöglichen es einem Webserver *stateful* zu sein

Sessions

```
GET / HTTP/1.1
Host: gmail.com

...
```

Sessions

```
HTTP/1.1 200 OK
Content-Type: text/html

...
```

Sessions

```
HTTP/1.1 200 OK
Content-Type: text/html
Set-Cookie: session=value
...
```

Sessions

```
GET / HTTP/1.1
Host: gmail.com

...
```

Sessions

```
GET / HTTP/1.1
Host: gmail.com
Cookie: session=value
...
```

login 

Shopping

- Einkaufswagen für einen Online-Shop für Bücher

Ausblick: Client- und serverseitiger Code im Zusammenspiel

- Bis jetzt: Jede Anfrage an den Webserver immer durch Neuladen der Seite
- Interaktive Anwendungen tauschen Informationen “im Hintergrund” aus, ohne, dass ein Neuladen der Seite von den Nutzern bemerkt wird