

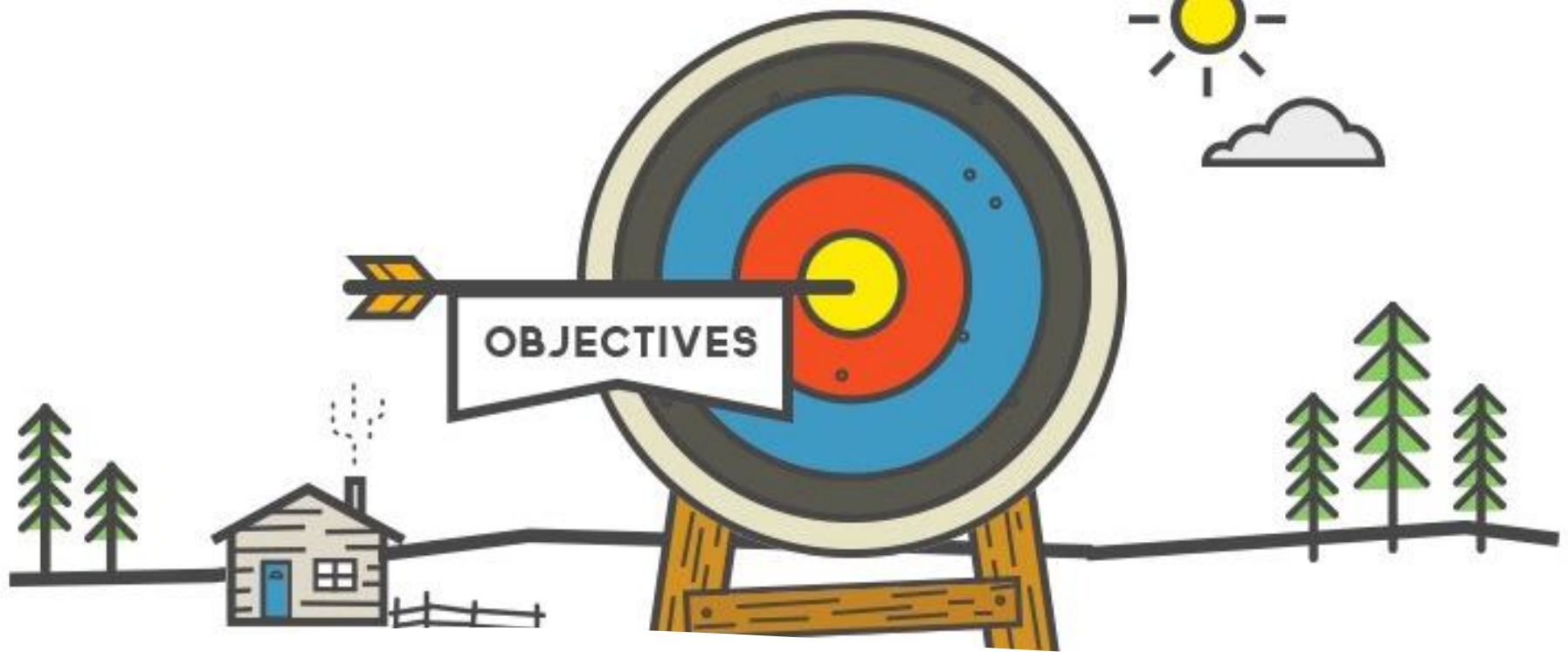


**Lisbon School
of Economics
& Management**
Universidade de Lisboa



Flask

Carlos J. Costa



Learning Goals

- Understand main Concepts of Flask
- Now how to deploy
- Create small Web App

Agenda

- Flask
- Libraries Werkzeug and Jinja
- Flask Application
- Routing
- Static Files
- Rendering Templates
- HTTP Methods
- Save Data
- Creating a small App
- Where to deploy



Flask

Flask

- Flask is a microframework for Python
- is a lightweight WSGI web application framework
- began as a simple wrapper around Werkzeug and Jinja
- Created by Armin Ronacher
- BSD licensed
- <https://palletsprojects.com/p/flask/>



Werkzeug

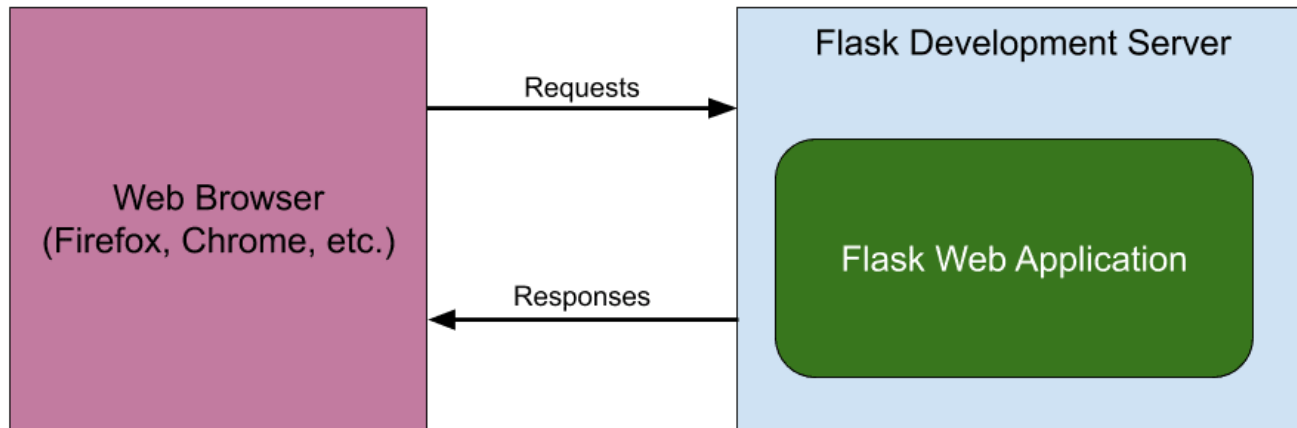
Werkzeug

- utility library for Python
- toolkit for Web Server Gateway Interface (WSGI) applications,
- licensed under a BSD License. Werkzeug
- can realize software objects for request, response, and utility functions.
- Python 2.7 and 3.5 and late
- <https://palletsprojects.com/p/werkzeug/>

Jinja

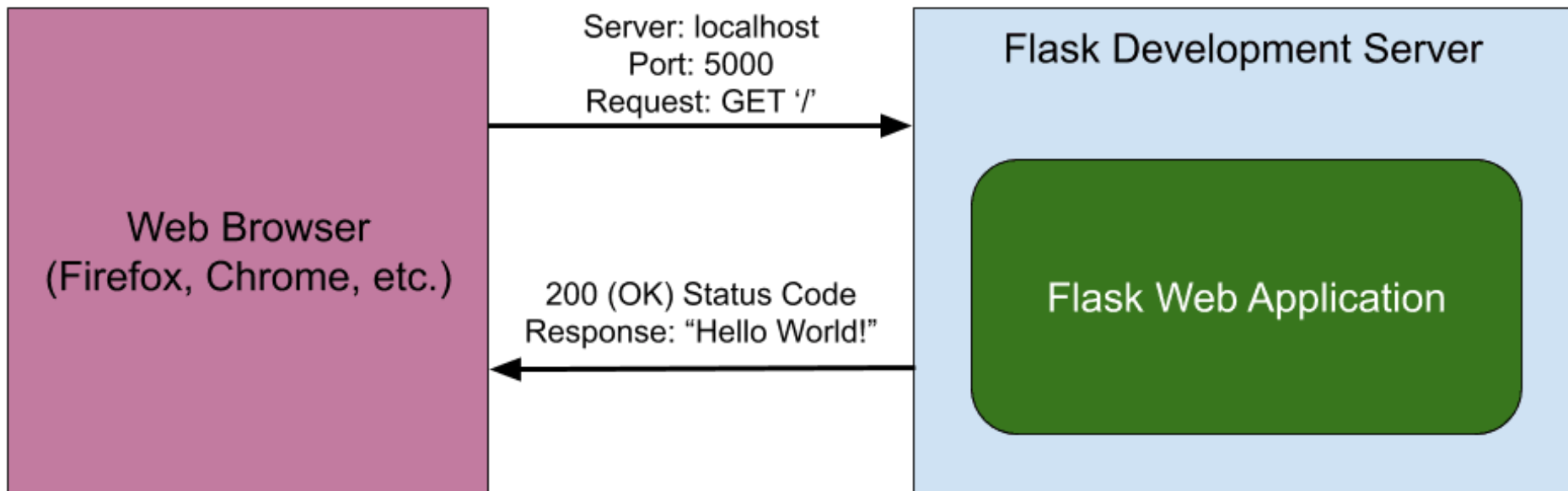
- is a template engine for the Python
- is licensed under a BSD License.
- similar to the Django web framework
- it handles templates in a sandbox.
- <https://palletsprojects.com/p/jinja/>
- <https://github.com/pallets/jinja>





Flask Application (ex01)

```
from flask import Flask
app = Flask(__name__)
@app.route('/')
def hello_world():
    return 'Hello, World!'
if __name__ == "__main__":
    app.run()
```



Flask Application

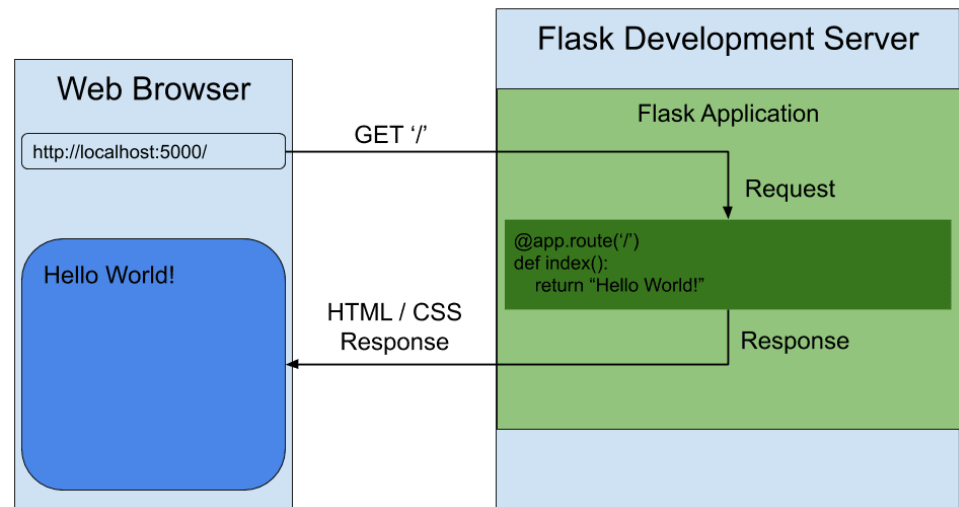
- Save in a folder flask_app.py
- Call:
`python flask_app.py`
- In the browser:
`localhost:5000`

Routing

- Use the `route()` decorator to bind a function to a URL.

```
@app.route('/')  
def index():  
    return 'Index  
Page'
```

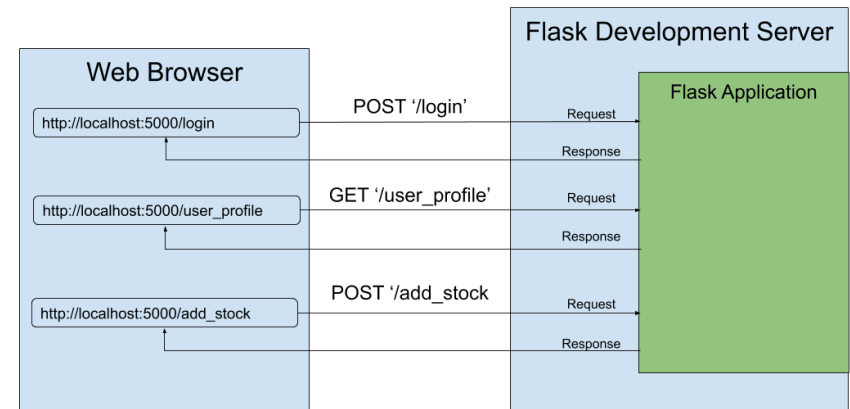
```
@app.route('/hello')  
def hello():  
    return 'Hello,  
World'
```



Routing (ex02)

```
from flask import Flask
app = Flask(__name__)
@app.route('/')
def hello_world():
    return 'Hello, World (in index page)'
@app.route('/hello')
def hello():
    return 'Hello, World (in hello page)'

if __name__ == "__main__":
    app.run()
```

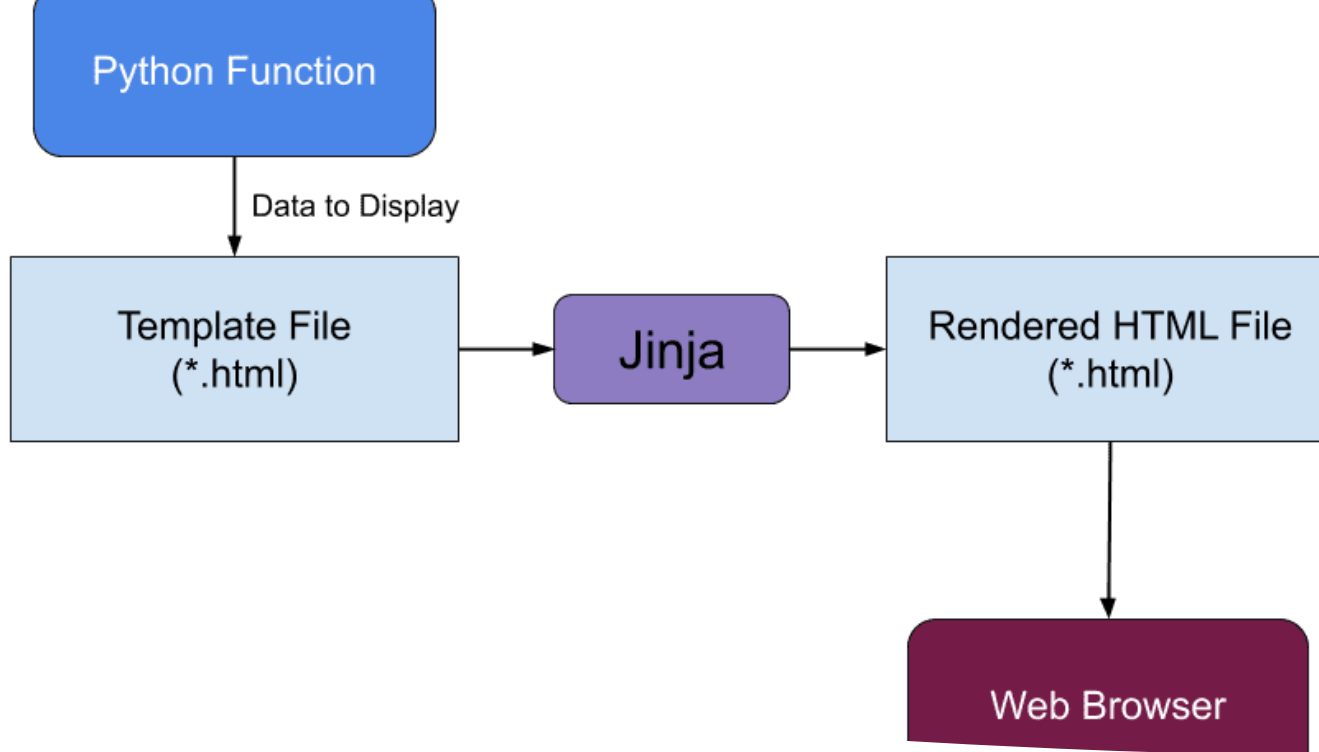


Static Files

- Dynamic web applications also need static files.
- E.g.CSS and JavaScript
- Create a folder called in the package or next to the module

/static

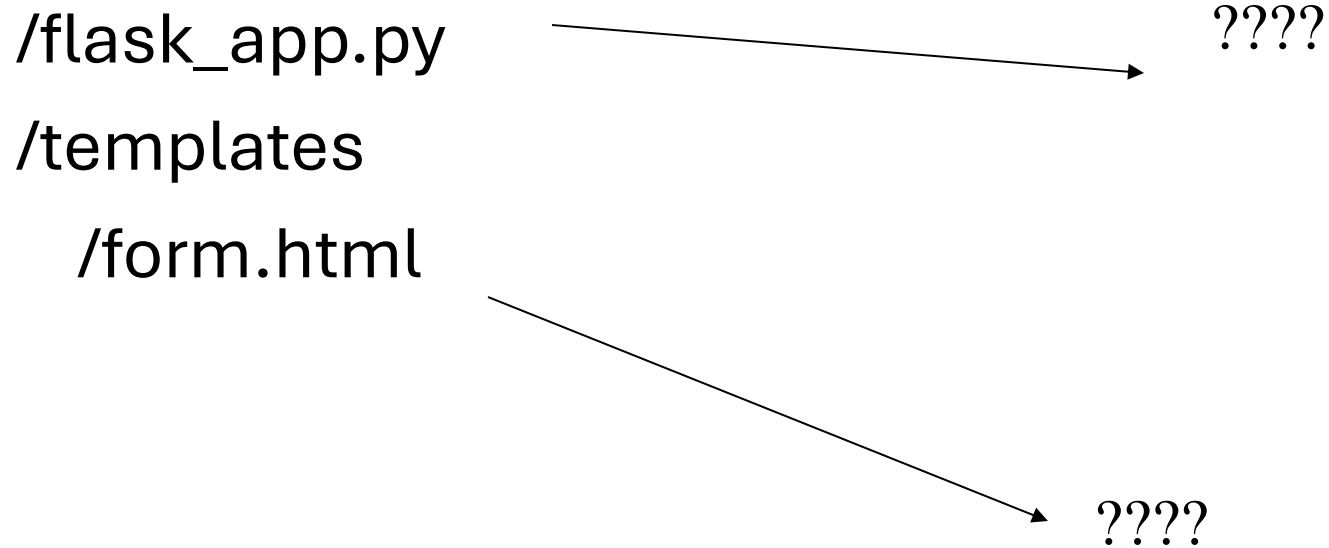
```
url_for('static', filename='style.css')
```



Rendering Templates

- Flask configures the Jinja2 template engine automatically
- To render a template use the `render_template()` method
- Example:
 `/flask_app.py`
 `/templates`
 `/hello.html`

Rendering Templates (ex03)



Rendering Templates

/flask_app.py

/templates

/hello.html

```
from flask import render_template

@app.route('/hello/')
@app.route('/hello/<name>')
def hello(name=None):
    return render_template('hello.html', name=name)
```

```
<!doctype html>
<title>Hello from Flask</title>
{% if name %}
  <h1>Hello {{ name }}!</h1>
{% else %}
  <h1>Hello, World!</h1>
{% endif %}
```

Rendering Templates (ex03)

/flask_app.py
/templates
/form.html

```
from flask import render_template

@app.route('/hello/')
@app.route('/hello/<name>')
def hello(name=None):
    return render_template('hello.html', name=name)
```

Rendering Templates (ex03)

flask_app.py
File

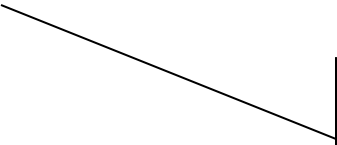
```
from flask import Flask, render_template
app = Flask(__name__)
@app.route('/')
def render_static():
    return render_template('form.html')
@app.route('/hello')
def hello():
    return 'Hello, World (in hello page)'
if __name__ == "__main__":
    app.run()
```

Rendering Templates (ex03)

/flask_app.py

/templates

/form.html



```
<!doctype html>
<title>Hello from Flask</title>
{% if name %}
  <h1>Hello {{ name }}!</h1>
{% else %}
  <h1>Hello, World!</h1>
{% endif %}
```

Rendering Templates (ex03)

```
<html>
```

```
<body>
```

```
<form action = "/save/" method = "POST">
```

```
<p>author <input type = "text" name = "Author" /></p>
```

```
<p>phrase <input type = "text" name = "Phrase" /></p>
```

```
<p><input type = "submit" value = "submit" /></p>
```

```
</form>
```

```
</body>
```

```
</html>
```

HTTP Methods

- Web applications use different HTTP methods when accessing URLs.
- By default, a route only answers to GET requests.
- use the methods argument of the route() decorator to handle different HTTP methods.



HTTP Methods

```
from flask import request
```

```
@app.route('/login', methods=['GET', 'POST'])
```

```
def login():
```

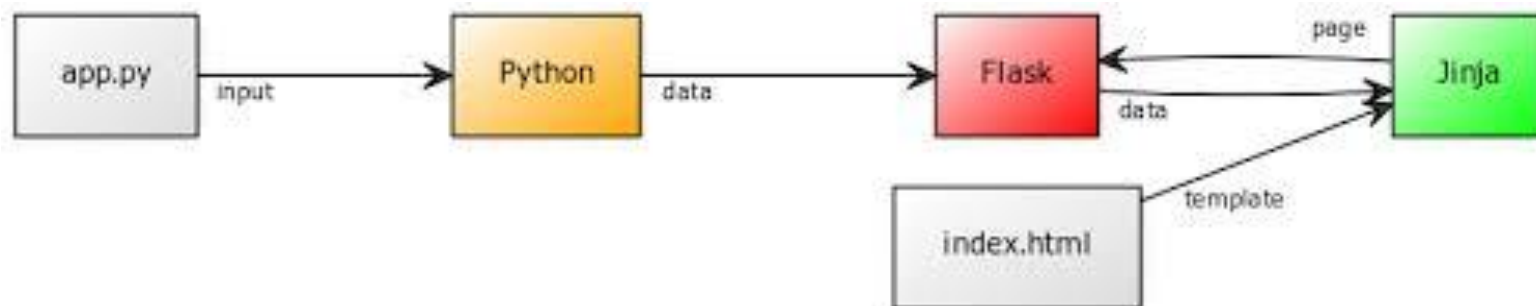
```
    if request.method == 'POST':
```

```
        return do_the_login()
```

```
    else:
```

```
        return show_the_login_form()
```





Rendering Template s (ex04)

```
from flask import Flask, render_template, request

app = Flask(__name__)

@app.route('/')
def render_static():
    return render_template('form.html')

@app.route('/save/', methods=['GET', 'POST'])
def hello():
    return 'Hello!!!'

if __name__ == "__main__":
    app.run()
```

Save Data (ex05)

```
from flask import Flask, render_template, request
app = Flask(__name__)
@app.route('/')
def render_static():
    return render_template('form.html')
@app.route('/save/', methods=['GET', 'POST'])
def index():
    data = request.form['Author']+" - "+request.form['Phrase']
    fo= open("test1.txt", "a+")
    fo.write(data+"\n")
    fo.close()
    return "thank you"
if __name__ == '__main__':
    app.run(debug = True)
```

Tiny App (ex06)

/flask_app.py

/templates

 /form.html

 /index.html



Tiny App

/flask_app.py

/templates

/form.html

/index.html



Tiny App

```
from flask import Flask, request, render_template
app = Flask(__name__)
@app.route('/save/', methods=['GET', 'POST'])
def write():
    data = request.form['Author']+" - "+request.form['Phrase']
    fo= open("test1.txt", "a+")
    fo.write(data+"\n")
    fo.close()
    return render_template('index.html')
@app.route('/read/')
def read():
    fo= open("test1.txt", "r")
    data1=fo.read()
    fo.close()
    return data1
@app.route('/')
def render_static():
    return render_template('form.html')
if __name__ == '__main__':
    app.run(debug = True)
```



Tiny App

/flask_app.py

/templates

/form.html

/index.html



Tiny App

```
<html>
  <body>
    <form action = "/save/" method =
"POST">
      <p>author <input type = "text" name =
"Author" /></p>
      <p>phrase <input type = "text" name =
"Phrase" /></p>
      <p><input type = "submit" value =
"submit" /></p>
    </form>
  </body>
</html>
```



Tiny App

/flask_app.py

/templates

/form.html

/index.html



Tiny App (ex06)

```
<html>
  <body>
    menu
    <p><a href="/read/">
      list data
    </a></p>
    <a href="..">form</a>
  </body>
</html>
```





| WikiWikiWeb.de |

- Where to deploy?

Alternatives

References

- <https://github.com/masterfloss/python-web>