Tutorial 6
Introduction to Flask

Kai CHEN

Overview

Flask is a micro web development framework for Python.

keep the core simple but extensible

- Lightweight
- Extensible

Overview

A minimum example

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello_world():
    return 'Hello World'

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

Contents

- Setting up
- Routing
- URL Building
- Accessing Request Data
- Responses

Setting up

- conda install flask
- pip install flask

Routing

Bind a function to a URL

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

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

http://localhost -> index()

http://localhost/hello -> hello()

Routing

Make parts of the URL dynamic

route(‘<variable_name>’)
route(‘<converter:variable_name>’)

```
@app.route('/user/<username>')
def show_user_profile(username):
    return 'User %s' % username

@app.route('/post/<int:post_id>')
def show_post(post_id):
    return 'Post %d' % post_id
```

http://localhost/user/jack -> show_user_profile(jack)

http://localhost/post/1234 -> show_post(1234)

Variable Rules

Converter types

string	(default) accepts any text without a slash
int	accepts positive integers
float	accepts positive floating point values
path	like string but also accepts slashes
uuid	accepts UUID strings

Routing

Specific HTTP Methods

```
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        do_the_login()
    else:
        show_the_login_form()
```

Accessing Request Data

Request object

- request.method
- request.url
- request.path
- request.headers
- request.cookies

Accessing Request Data

URL parameters

from flask import request

```
@app.route('/search', methods=['GET'])
def search():
    keyword = request.args.get('keyword', '')
```

http://localhost/search?keyword=cat
http://localhost/search

Accessing Request Data

Json data

```
@app.route('/search', methods=['POST'])
def search():
    data = request.get_json()
```

Accessing Request Data

Form data

```
@app.route('/login', methods=['GET', 'POST'])
def login():
    if method == 'POST':
        username= request.form.get('username', '')
        password= request.form.get('password', '')

@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        username= request.form['username']
        password= request.form['password']
```

catch KeyError

Accessing Request Data

Files

```python
@app.route('/upload', methods=['GET', 'POST'])
def upload_file():
    if request.method == 'POST':
        f = request.files['avatar_img']
        f.save('some_path')
```

```html
<form action="#" enctype="multipart/form-data" method="post">
    <input type="file" name="avatar_img">
    <input type="submit" value="Upload">
</form>
```

Responses

HTML page

```
@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'GET':
        return ""
            <form action="" method="post">
                <p><input type=text name=username>
                <p><input type=submit value=Login>
            </form>
        ""
```

Responses

Json string

```
@app.route('/submit', methods=['GET', 'POST'])
def submit():
    if request.method == 'POST':
        result = do_some_things() # result is a dict object
        return jsonify(result)
```

Responses

Redirection

```
@app.route('/profile', methods=['GET', 'POST'])
def submit():
    if request.method == 'GET':
        if validate_user():
            return render_template('profile.html')
        else:
            return redirect(url_for('login'))
```

Responses

Errors

```
@app.route('/profile/<username>')
def profile(username):
    if not user_exists(username):
        abort(404)
    else:
        do_something()


@app.errorhandler(404)
def page_not_found(error):
    return render_template('404.html'), 404
```

Examples

How to organize a Flask project

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

```
config.py
requirements.txt
run.py
instance/
    config.py
yourapp/
    __init__.py
    views.py
    models.py
    forms.py
    static/
    templates/
```

```
config.py
requirements.txt
run.py
instance/
    config.py
blueprint1/
blueprint2/
blueprint3/
```

complexity

Thank you!