Cheat Sheet for comprensive Flask

Comprehensive Flask Cheat Sheet

1. Introduction to Flask

- **Flask** is a lightweight WSGI web application framework in Python.

- It is designed to make getting started quick and easy, with the ability to scale up to complex applications.
- Flask is often referred to as a microframework because it does not require particular tools or libraries.

---

2. Basic Setup

Installation

```
pip install Flask
```

Minimal Flask Application

```
from flask import Flask

app = Flask(__name__)

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

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

Running the Application

```
python app.py
```

---

3. Routing

Basic Routing

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

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

Variable Rules

```
@app.route('/user/<username>')
def show_user_profile(username):
    return f'User {username}'

@app.route('/post/<int:post_id>')
def show_post(post_id):
    return f'Post {post_id}'
```

HTTP Methods

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

---

4. Templates

Rendering Templates

```
from flask import render_template

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

Template Inheritance

```
<!-- base.html -->
<!doctype html>
```

```
<html>
<head>
    <title>{% block title %}{% endblock %}</title>
</head>
<body>
    <div id="content">{% block content %}{% endblock %}</div>
</body>
</html>

<!-- child.html -->
{% extends "base.html" %}
{% block title %}Index{% endblock %}
{% block content %}
    <h1>Hello, World!</h1>
{% endblock %}
```

---

5. Request Handling

Accessing Request Data

```
from flask import request

@app.route('/login', methods=['POST'])
def login():
    username = request.form['username']
    password = request.form['password']
    return f'Username: {username}, Password: {password}'
```

Query Parameters

```
@app.route('/query')
def query():
    search_query = request.args.get('q')
    return f'Search Query: {search_query}'
```

---

6. Static Files

Serving Static Files

```
from flask import send_from_directory

@app.route('/static/<path:filename>')
```

```
def static_files(filename):
    return send_from_directory(app.static_folder, filename)
```

Default Static Folder

- Flask automatically serves files from the `static` folder located in the root directory.

---

7. Sessions and Cookies

Using Sessions

```
from flask import session

@app.route('/')
def index():
    if 'username' in session:
        return f'Logged in as {session["username"]}'
    return 'You are not logged in'

@app.route('/login', methods=['POST'])
def login():
    session['username'] = request.form['username']
    return 'Logged in'

@app.route('/logout')
def logout():
    session.pop('username', None)
    return 'Logged out'
```

Setting Cookies

```
from flask import make_response

@app.route('/set_cookie')
def set_cookie():
    resp = make_response('Setting a cookie')
    resp.set_cookie('username', 'the_username')
    return resp
```

---

8. Forms and Validation

Form Handling

```
from flask import request

@app.route('/submit', methods=['POST'])
def submit():
    name = request.form['name']
    email = request.form['email']
    return f'Name: {name}, Email: {email}'
```

Form Validation with Flask-WTF

```
from flask_wtf import FlaskForm
from wtforms import StringField, PasswordField, SubmitField
from wtforms.validators import DataRequired

class LoginForm(FlaskForm):
    username = StringField('Username', validators=[DataRequired()])
    password = PasswordField('Password', validators=[DataRequired()])
    submit = SubmitField('Login')

@app.route('/login', methods=['GET', 'POST'])
def login():
    form = LoginForm()
    if form.validate_on_submit():
        return f'Username: {form.username.data}, Password:
{form.password.data}'
    return render_template('login.html', form=form)
```

---

9. Database Integration

SQLAlchemy with Flask

```
from flask_sqlalchemy import SQLAlchemy

app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///test.db'
db = SQLAlchemy(app)

class User(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(80), unique=True, nullable=False)
    email = db.Column(db.String(120), unique=True, nullable=False)

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

```
def add_user():
    username = request.form['username']
    email = request.form['email']
    new_user = User(username=username, email=email)
    db.session.add(new_user)
    db.session.commit()
    return 'User added'
```

Database Migrations with Flask-Migrate

```
pip install Flask-Migrate
```

```
from flask_migrate import Migrate

migrate = Migrate(app, db)

# Initialize migrations
flask db init

# Create a migration
flask db migrate -m "Initial migration."

# Apply the migration
flask db upgrade
```

---

10. Blueprints

Creating a Blueprint

```
from flask import Blueprint

bp = Blueprint('admin', __name__, url_prefix='/admin')

@bp.route('/')
def admin_index():
    return 'Admin Page'

app.register_blueprint(bp)
```

Organizing Blueprints

- Blueprints help in organizing a Flask application into smaller, modular components.

---

11. Error Handling

Custom Error Pages

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

@app.errorhandler(500)
def internal_error(error):
    return render_template('500.html'), 500
```

Debug Mode

```
app.run(debug=True)
```

---

12. Testing

Writing Tests

```
import unittest
from app import app

class FlaskTestCase(unittest.TestCase):
    def setUp(self):
        self.app = app.test_client()
        self.app.testing = True

    def test_home_page(self):
        response = self.app.get('/')
        self.assertEqual(response.status_code, 200)
        self.assertEqual(response.data, b'Index Page')

if __name__ == '__main__':
    unittest.main()
```

Running Tests

```
python -m unittest discover
```

---

13. Deployment

Deploying with Gunicorn

```
pip install gunicorn
gunicorn -w 4 app:app
```

Deploying on Heroku

```
pip install gunicorn
pip freeze > requirements.txt
echo "web: gunicorn app:app" > Procfile
git init
git add .
git commit -m "Initial commit"
heroku create
git push heroku master
```

---

14. Advanced Topics

Custom CLI Commands

```
import click
from flask import Flask

app = Flask(__name__)

@app.cli.command("hello")
@click.argument("name")
def hello_command(name):
    click.echo(f"Hello, {name}!")

# Usage: flask hello <name>
```

Background Tasks with Celery

```
pip install celery
```

```
from celery import Celery

app = Flask(__name__)
app.config['CELERY_BROKER_URL'] = 'redis://localhost:6379/0'
app.config['CELERY_RESULT_BACKEND'] = 'redis://localhost:6379/0'
```

```
celery = Celery(app.name, broker=app.config['CELERY_BROKER_URL'])
celery.conf.update(app.config)

@celery.task
def add_together(a, b):
    return a + b

@app.route('/add')
def add():
    result = add_together.delay(10, 15)
    return f'Task ID: {result.id}'
```

---

15. Tips and Tricks

Debugging with Flask-DebugToolbar

```
pip install flask-debugtoolbar
```

```
from flask_debugtoolbar import DebugToolbarExtension

app.config['DEBUG_TB_INTERCEPT_REDIRECTS'] = False
toolbar = DebugToolbarExtension(app)
```

Environment Configuration

```
import os

app.config['SECRET_KEY'] = os.getenv('SECRET_KEY',
'default_secret_key')
```

Profiling with Flask-Profiler

```
pip install flask-profiler
```

```
from flask_profiler import Profiler

app.config["flask_profiler"] = {
    "enabled": app.config["DEBUG"],
    "storage": {
        "engine": "sqlite"
    },
```

```
    "basicAuth":{
        "enabled": True,
        "username": "admin",
        "password": "admin"
    },
    "ignore": [
        "^/static/.*"
    ]
}

profiler = Profiler()
profiler.init_app(app)
```

This cheat sheet provides a comprehensive overview of Flask, covering essential features, setup, routing, templates, request handling, database integration, and more. Use this as a quick reference guide to build and deploy Flask applications efficiently.

By Ahmed Baheeg Khorshid

ver 1.0