Flask Cheat Sheet

Routing · Jinja2 · Blueprints · SQLAlchemy

SETUP & APP FACTORY

```
pip install flask flask-sqlalchemy
def create_app(config=None):
app = Flask(__name__)
app.config.from_object(config)
db.init_app(app)
app.register_blueprint(api_bp,
url_prefix='/api')
return app
```

ROUTING

```
@app.route('/users',
methods=['GET','POST'])
def users():
if request.method == 'POST':
data = request.get_json()
return jsonify(data), 201
return jsonify(User.query.all())
@app.route('/users/<int:user_id>')
```

REQUEST & RESPONSE

```
request.method  request.args
request.form
request.json  request.get_json()
request.files  request.headers
jsonify({'key': 'value'})
make_response(body, 200, {'X-Custom':
'val'})
redirect(url_for('index'))
abort(404) / abort(401)
```

TEMPLATES (JINJA2)

```
return render_template('index.html',
users=users)
{{ user.name | upper }}
{% for u in users %}{{ u.name }}{%
endfor %}
{% extends 'base.html' %}
{% block body %}...{% endblock %}
{{ url_for('users', _external=True) }}
```

BLUEPRINTS

```
api_bp = Blueprint('api', __name__)
@api_bp.route('/users')
def get_users(): ...
# In factory:
app.register_blueprint(api_bp,
url_prefix='/api')
```

Separate modules by feature; maintain isolation

FLASK-SQLALCHEMY

```
db = SQLAlchemy()
class User(db.Model):
id = db.Column(db.Integer,
primary_key=True)
email = db.Column(db.String(120),
unique=True)
User.query.filter_by(active=True).all()
db.session.add(user) /
db.session.commit()
flask db upgrade  (Flask-Migrate)
```

ERROR HANDLING

```
@app.errorhandler(404)
def not_found(e):
return jsonify(error='Not found'), 404
@app.errorhandler(Exception)
def handle_exception(e):
return jsonify(error=str(e)), 500
try_except + abort() for controlled
errors
```

SESSIONS & COOKIES

```
from flask import session
session['user_id'] = user.id
session.get('user_id')
session.clear()
app.secret_key = 'change-me'
response.set_cookie('token', value,
httponly=True)
```

Session signed server-side with secret_key

BEFORE/AFTER REQUESTS

```
@app.before_request
def auth_check():
if not session.get('user_id') and
request.endpoint != 'login':
return jsonify(error='Unauthorized'),
401
@app.after_request
def add_headers(response):
response.headers['X-Frame-Options'] =
'DENY'
return response
```

CONFIGURATION

```
app.config.from_pyfile('config.py')
app.config.from_envvar('APP_SETTINGS')
app.config['DEBUG'] =
os.environ.get('DEBUG', False)
class ProductionConfig:
SQLALCHEMY_DATABASE_URI =
os.environ['DATABASE_URL']
SECRET_KEY = os.environ['SECRET_KEY']
```

POPULAR EXTENSIONS

```
Flask-SQLAlchemy  ORM integration
Flask-Migrate     Alembic migrations
Flask-Login       user session
management
Flask-JWT-Extended  JWT tokens
Flask-WTF         CSRF + form validation
Flask-CORS        CORS headers
Marshmallow
serialization/validation
```

COMMON GOTCHAS

Application context: push ctx for DB outside request
Thread safety: don't store state on app object
SECRET_KEY must be set or sessions won't work
FLASK_ENV=development enables debugger
Use factory pattern for testing
avoid module-level app