1.创建虚拟环境

2.思考（关键）

3.建立文件结构

4.创建app应用

app/\_\_init\_\_

from flask import Flask

app = Flask(\_\_name\_\_)

from app import routes

5.创建视图

app/routes

from app import app

@app.route(‘/’)

def index():

return ‘Hello World’

6.使用模板

在步骤5中可以看到返回的是一个简单的Hello World,但是这样无法实现前后端分离,而且遇到大型的HTML文件显得很无力，所以需要引进模板app/templates/index.html,将return改为return render\_template(‘index.html’)

7.表单

为了能够处理 web 表单，我们将使用 [Flask-WTF](http://packages.python.org/Flask-WTF) ，该扩展封装了 [WTForms](http://wtforms.simplecodes.com/docs/dev) 并且恰当地集成进 Flask 中。

首先在配置文件/config.py中设置如下

SECRET\_KEY = os.environ.get('SECRET\_KEY') or 'you-will-never-know'

在app/\_\_init\_\_.py下应用

“””在jinjia中有多少种api”””

8.数据库

/config.py

**import** **os**

basedir = os.path.abspath(os.path.dirname(\_\_file\_\_))

SQLALCHEMY\_DATABASE\_URI = os.environ.get('DATABASE\_URL') or \  
 'sqlite:///' + os.path.join(basedir, 'app.db')  
SQLALCHEMY\_TRACK\_MODIFICATIONS = False

SQLALCHEMY\_DATABASE\_URI 是 Flask-SQLAlchemy 扩展需要的。这是我们数据库文件的路径。

SQLALCHEMY\_MIGRATE\_REPO 是文件夹，我们将会把 SQLAlchemy-migrate 数据文件存储在这里。

初始化应用的时候也需要初始化数据库

/app/\_\_init\_\_.py

**from** **flask** **import** Flask

**from** **flask.ext.sqlalchemy** **import** SQLAlchemy

**from flask\_migrate import** Migrate

app = Flask(\_\_name\_\_)

app.config.from\_object(Config)

db = SQLAlchemy(app)

migrate = Migrate(app, db)

**from** **app** **import** views, models

创建模型

/app/models

**from** **app** **import** db

**class** **User**(db.Model):

id = db.Column(db.Integer, primary\_key = **True**)

nickname = db.Column(db.String(64), index = **True**, unique = **True**)

email = db.Column(db.String(120), index = **True**, unique = **True**)

**def** \_\_repr\_\_(self):

**return** '<User *%r*>' % (self.nickname)

创建数据库

自己编写python脚本创建数据库

/db\_create.py

**from** **migrate.versioning** **import** api

**from** **config** **import** SQLALCHEMY\_DATABASE\_URI

**from** **config** **import** SQLALCHEMY\_MIGRATE\_REPO

**from** **app** **import** db

**import** **os.path**

db.create\_all()

**if** **not** os.path.exists(SQLALCHEMY\_MIGRATE\_REPO):

api.create(SQLALCHEMY\_MIGRATE\_REPO, 'database repository')

api.version\_control(SQLALCHEMY\_DATABASE\_URI, SQLALCHEMY\_MIGRATE\_REPO)

**else**:

api.version\_control(SQLALCHEMY\_DATABASE\_URI, SQLALCHEMY\_MIGRATE\_REPO, api.version(SQLALCHEMY\_MIGRATE\_REPO))

在建立好模型之后进行数据库迁移,启动虚拟环境

set FLASK\_APP = microblog.py

flask db migrate -m ‘注释’

flask db upgrade

9.用户登陆

密码哈希(werkzeug)：

hash = generate\_password\_hash('foobar')//生成哈希值

hash：'pbkdf2:sha256:50000$vT9fkZM8$04dfa35c6476acf7e788a1b5b3c35e217c78dc04539d295f011f01f18cd2175f'

check\_password\_hash(hash, 'foobar')//判断密码与哈希值是否匹配,如果为真返回True

Flask-Login:

app/views

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

def login():

if current\_user.is\_authenticated:

return redirect(url\_for('index'))

form = LoginForm()

if form.validate\_on\_submit():

user = User.query.filter\_by(username=form.username.data).first()

if user is None or not user.check\_password(form.password.data):

flash('Invalid username or password')

return redirect(url\_for('login'))

login\_user(user, remember=form.remember\_me.data)

return redirect(url\_for('index'))

return render\_template('login.html', title='Sign In', form=form)