Beijing Forest Studio
北京理工大学信息系统及安全对抗实验中心

Web快速开发方法

门元昊　硕士研究生

2020年01月12日

内容提要

- Web开发模式
- 常用Web框架

预期收获

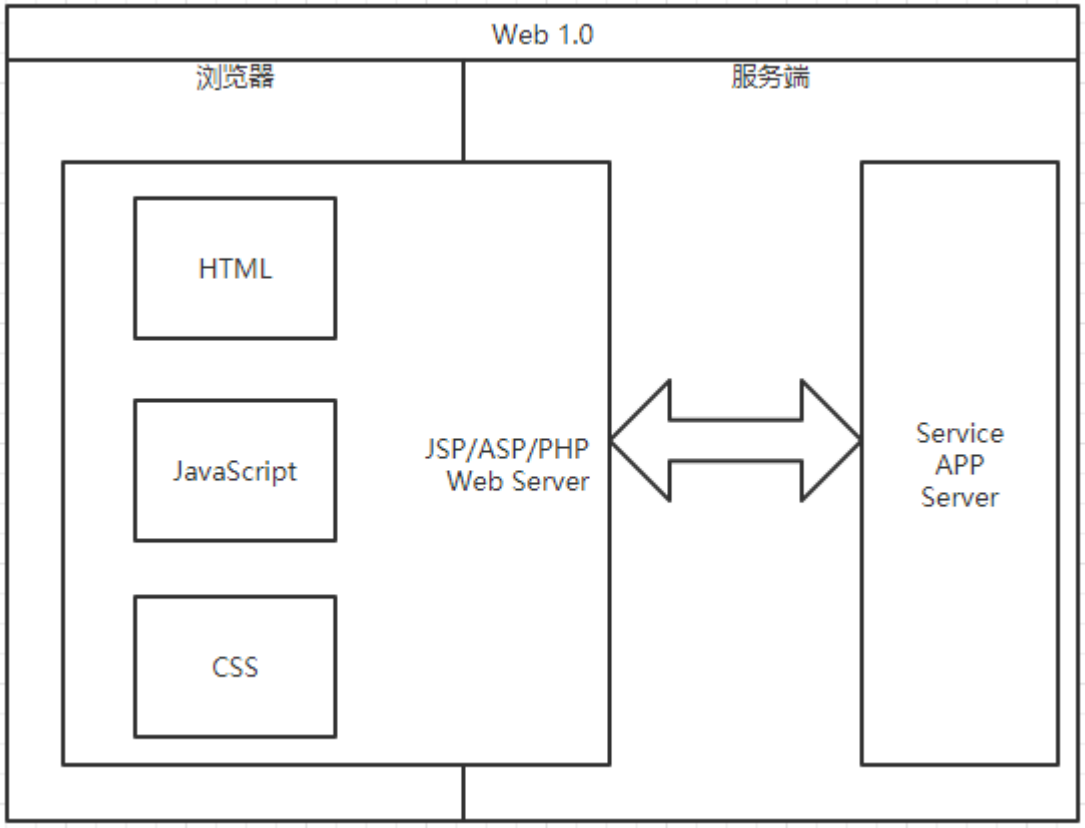
- 了解Web开发模式
- 了解常见的开发框架
- 常用功能（数据库、认证鉴权、任务调度等）的常用库

Web开发模式

- 前后端混合
- 后端主导的MVC
- 基于Ajax的SPA（单页面应用）
- 前端主导的MV*
- 基于Nodejs的全栈

前后端混合

- 特点
  - 完全不区分前后端，页面直接由服务端生成



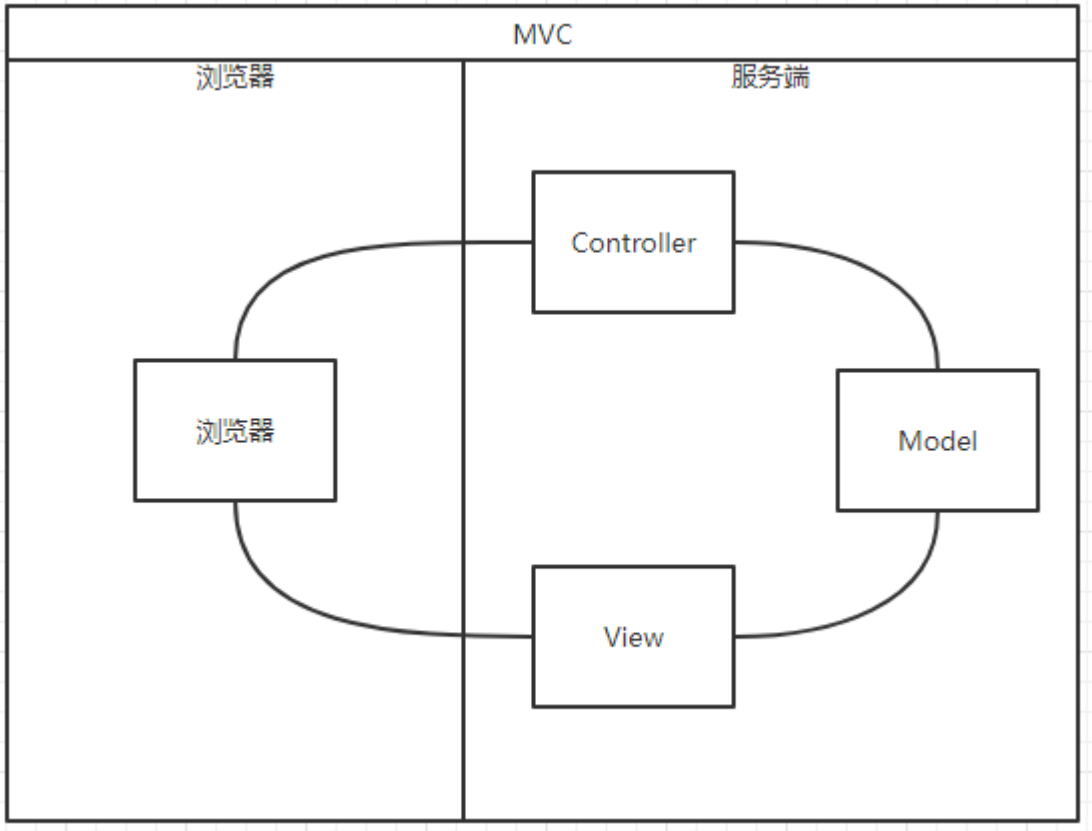
前后端混合

- 问题
  - 随着服务增加，后端的调用关系变的更加复杂
    - 前端开发不友好（搭建环境困难、远程开发验证繁琐）
    - 后端维护工作量大大提高（除了样式问题，其他全是后端的锅）
  - 代码可维护性越来越差
    - 业务代码与展示代码高度耦合，无法维护

```
1 <html>
2 <head><title>Hello World</title></head>
3 <body>
4 Hello World!
5 <br/>
6 <%
7     out.println("Your IP address is " + request.getRemoteAddr());
8 %>
9 </body>
10 </html>
```

后端主导的MVC

- 特点
  - 数据、逻辑、表现分离，各司其职

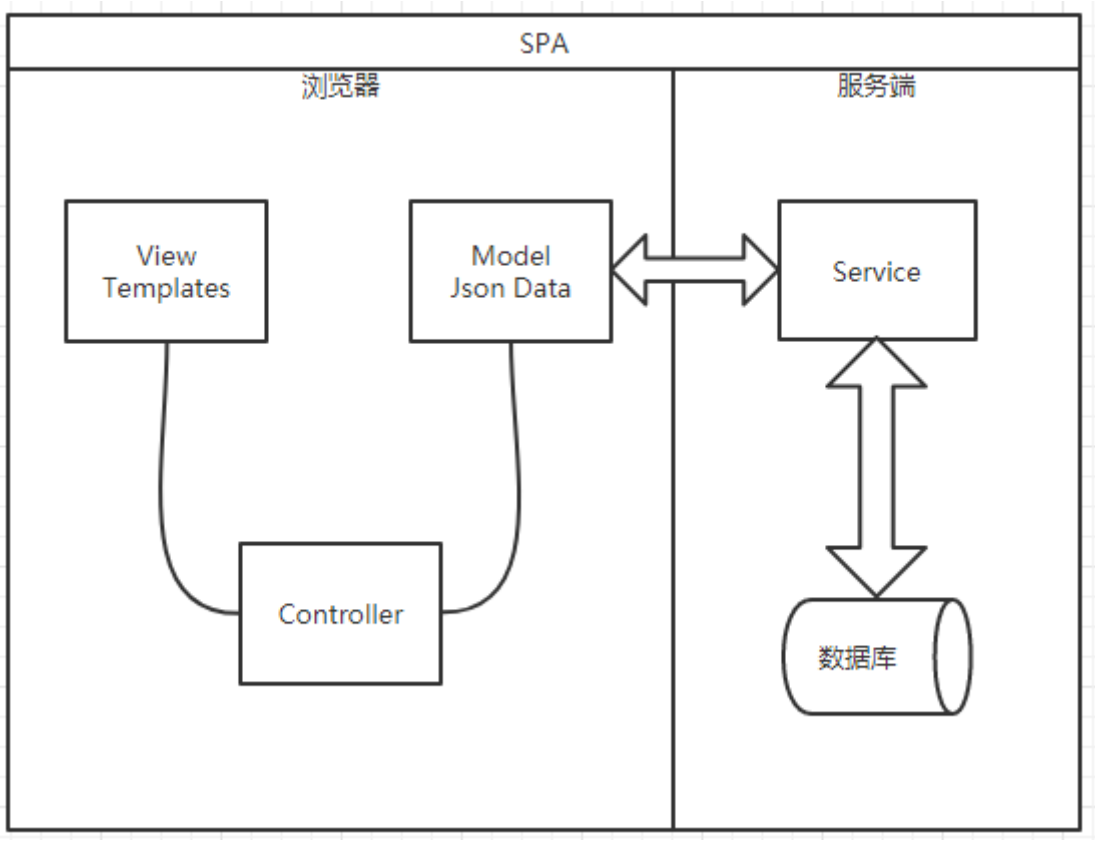


后端主导的MVC

- 问题
  - 仍然没有解决前端对整体开发环节的依赖
    - 前端提供模版后端套用：来回需要改动调整，沟通成本高
    - 前端负责View（视图层）的开发：前端重度绑定后端环节
  - 前后端分工不明
    - Model 后端负责
    - View 前端负责
    - Controller
      - 页面路由部分
      - 业务逻辑部分
      - 数据管理部分

基于Ajax的SPA

- 特点
  - 仅有一张Web页面，使用Ajax与后端同步数据
  - 主要的业务逻辑均放到前端由JS实现

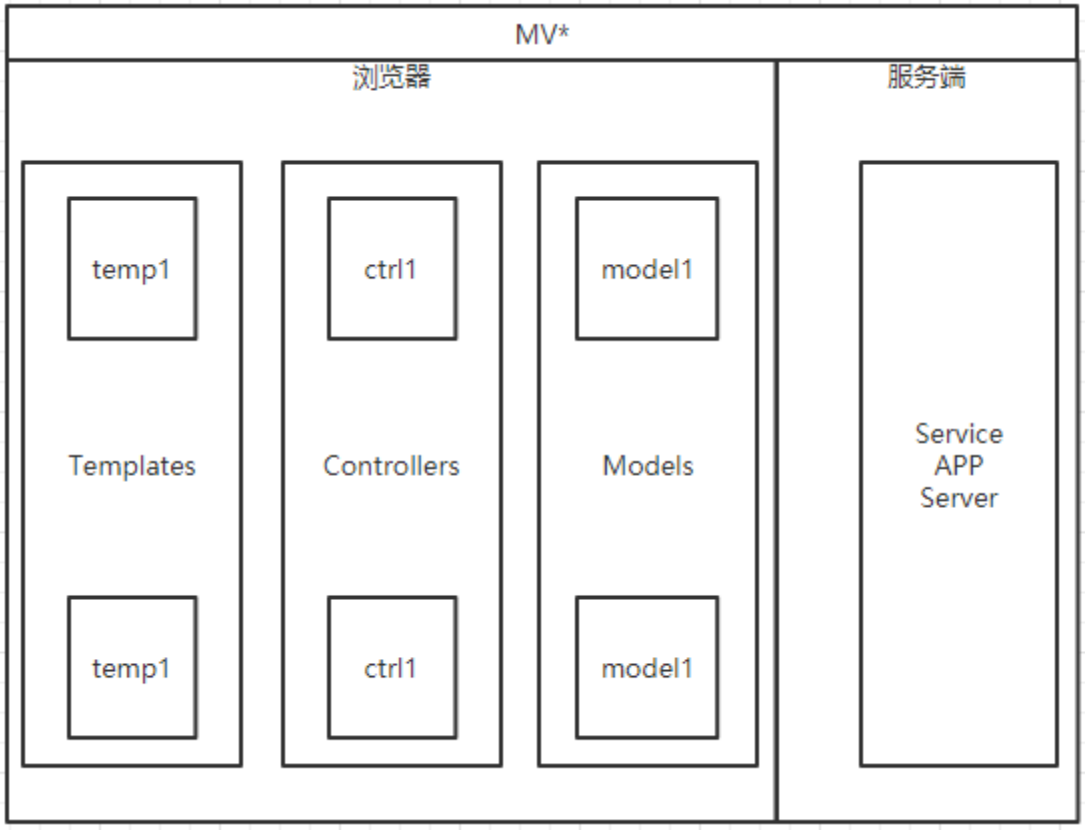


Ajax的SPA

- 问题
  - 前后端接口约定
  - 前端开发复杂度
    - 大量js逻辑代码的引入，对整个前端的代码组织、文件管理的方案都提出了挑战

前端主导的MV*

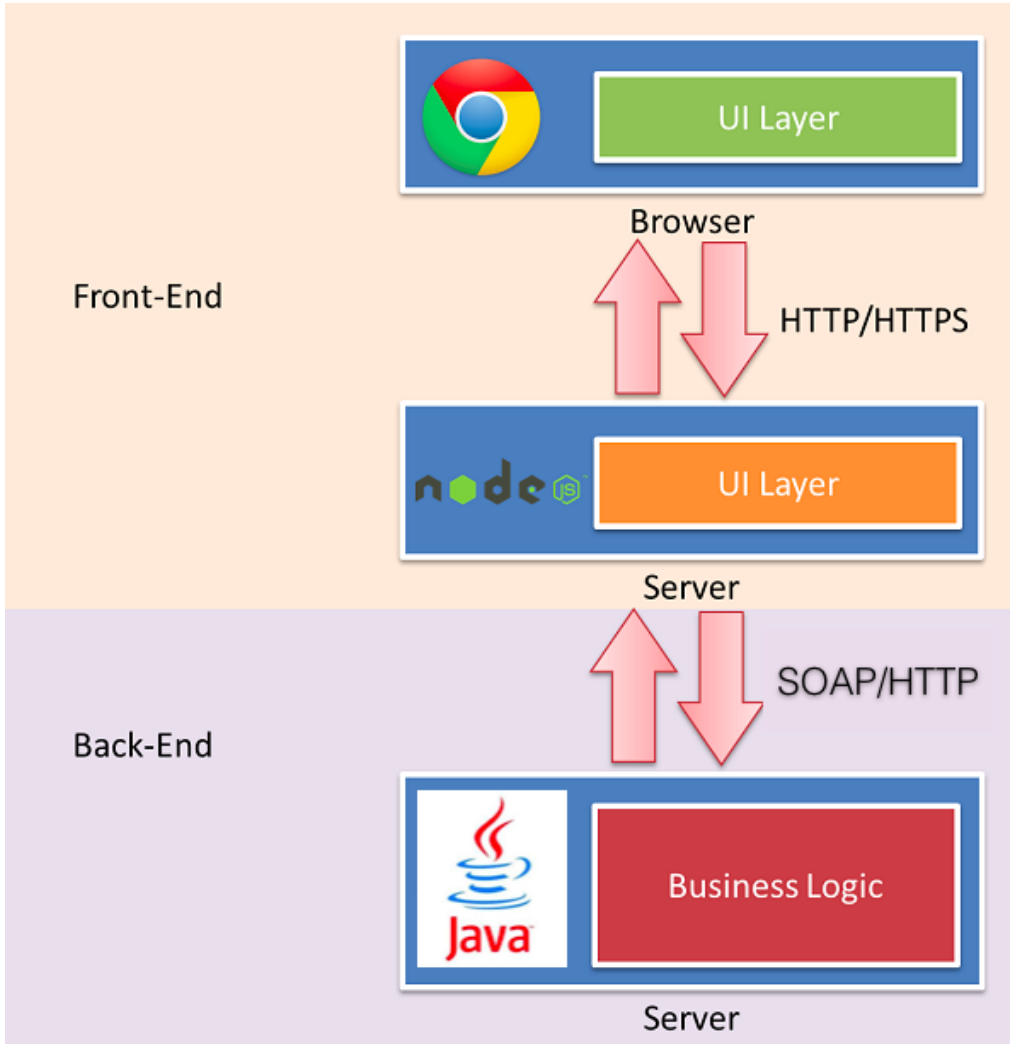
- 特点
  - 基于SPA的发展
  - 在前端代码中先分层，再做层内逻辑切分

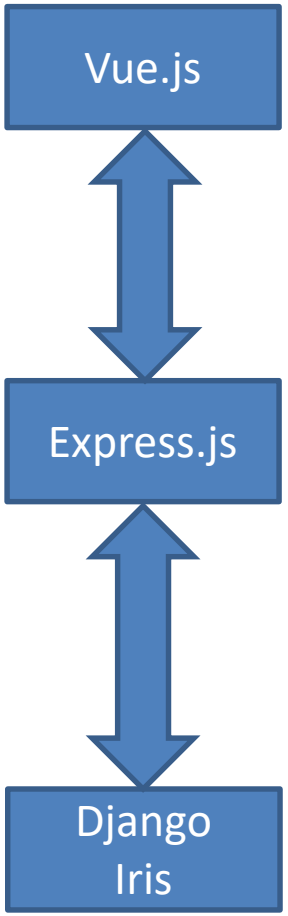


前端主导的MV*

- 问题
  - 代码不能复用
    - 前后端均需要对数据进行校验
  - 单页应用无法满足全部需求，因此需要保留部分多页面应用，从而使前端无法完全独立掌握URL设计

基于Nodejs的全栈





Web开发技术发展

- 总结
  - 不同的开发模式并无好坏高低之分，只有合不合适
  - 实验室常见场景

前端需求	后端需求	推荐方案	备注
几个简单页面 （静态、js逻辑简单）	简单后端 （无登陆、数据库等）	前后端混合	
		后端主导的MVC	Django Flask …
路由复杂页面 （页面多、跳转多）	有数据库管理、登陆等需求	后端主导的MVC	
逻辑复杂页面 （js逻辑复杂）	简单后端 （无登陆、数据库等）	基于Ajax的SPA 前端主导的MV*	
复杂页面 （路由复杂、逻辑复杂）	有数据库管理、登陆等需求	基于Nodejs的全栈	
		后端主导的MVC	Django

Web框架

- 概念
  - Web应用框架（Web application framework）是一种开发框架，用来支持动态网站、网络应用程序等的开发
- 功能
- 常用框架介绍
  - Django
  - Flask
  - Python

Web框架

- 功能
  - **路由管理**
  - **Cookie、Session、Token（状态管理）**
  - 模版渲染
  - 静态资源
  - 日志记录
  - 配置管理
  - ORM（数据库管理）
  - 任务调度
  - …

```
class IdentityLog(models.Model):
    """"""
    task_name = models.CharField(max_length=256)
    task_amount = models.IntegerField()
    crt_amount = models.IntegerField()
    cld_amount = models.IntegerField()
    mil_amount = models.IntegerField()
    tv_amount = models.IntegerField()
    isp_amount = models.IntegerField()
    gov_amount = models.IntegerField()
    edu_amount = models.IntegerField()
    file_path = models.CharField(max_length=512)
    task_date = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return self.task_name
```

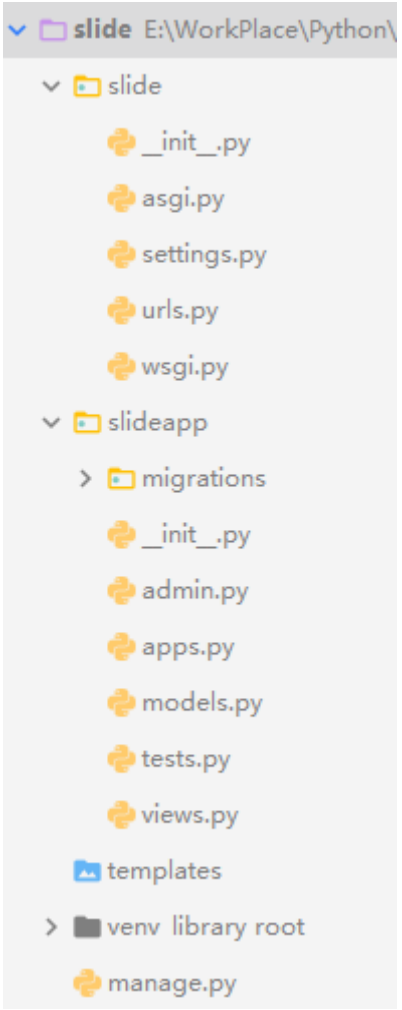
```
    "method": method_state,
    "tech": techs_state,
    "info": method_info
}
logging.debug('ctx is {0}'.format(context))
```

Django

- 简介
  - Django是一个由Python写成的Web应用框架。
  - Django的主要目的是简便、快速的开发**数据库驱动**的网站。

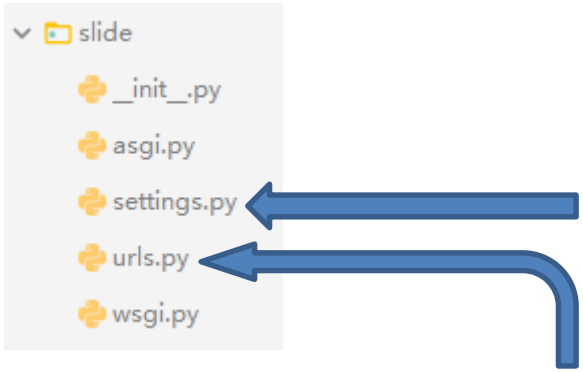
Django

- 安装方法
  - (python3 -m) pip install Django
- 使用方法
  - 创建项目
    - 使用pycharm创建django项目（推荐）
    - 使用cli创建django项目
      - django-admin startproject 【项目名称】
  - 创建app（django的业务功能单元）
    - python manage.py startapp 【app名称】



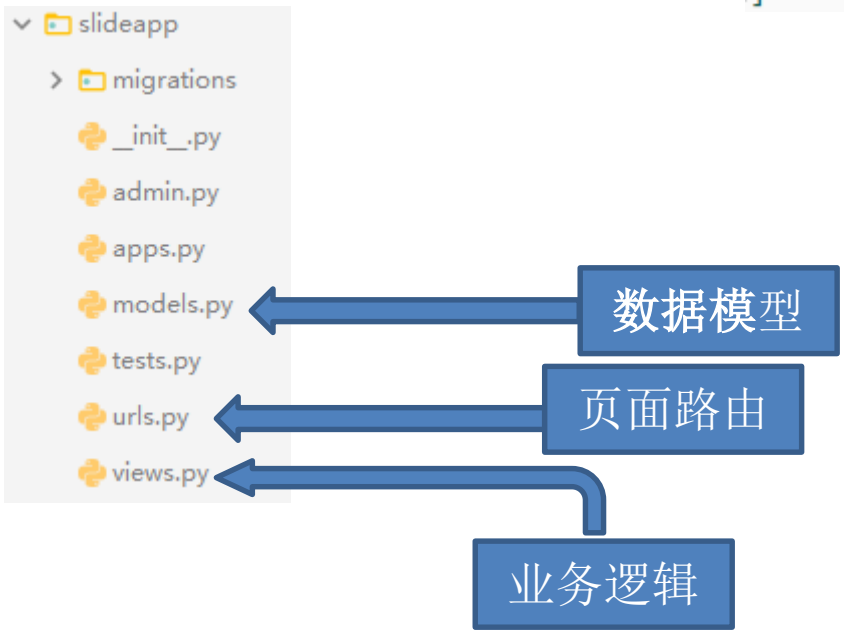
Django

- 程序编写
  - 前置准备
    - 注册app
    - 添加app路由
  - 编写app




```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'slideapp.apps.SlideappConfig',
]
```

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('slide/', include('slideapp.urls')),
]
```

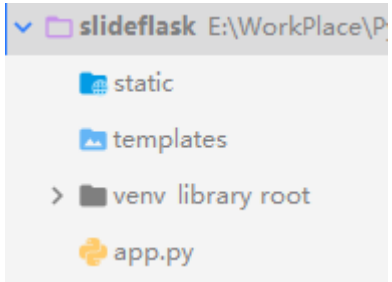


Django

- App编写示例
  - Writing your first Django app 【官方教程】
    - https://docs.djangoproject.com/en/3.0/intro/tutorial01/

Flask

- 简介
  - Flask是一个由Python写成的**轻量级**Web应用框架。
  - 默认不具有数据库管理和表单验证等功能
- 安装方法
  - (python3 -m) pip install Flask
- 使用方法
  - 创建项目
    - 使用pycharm创建Flask项目（推荐）




```python
from flask import Flask

app = Flask(__name__)


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


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

Cherrypy

- 简介
  - Cherrypy是一个由Python写成的**面向对象**的**轻量级**Web应用框架。
  - 默认不具有数据库管理等功能
- 安装方法
  - (python3 -m) pip install cherrypy
- 使用方法
  - 直接import cherrypy即可

Cherrypy

- 代码示例

```python
import cherrypy


class SimpleServer(object):
    @cherrypy.expose
    def index(self):
        return open('index.html')

    @cherrypy.expose
    def generate(self, length=8):
        token = ''.join(random.sample(string.hexdigits, int(length)))
        cherrypy.session['token'] = token
        return token

    @cherrypy.expose
    def form(self):
        return """<html>
          <head></head>
          <body>
            <form method="get" action="generate">
              <input type="text" value="8" name="length" />
              <button type="submit">Give it now!</button>
            </form>
          </body>
        </html>"""
```

```python
if __name__ == "__main__":
    config = {
        '/': {
            'tools.sessions.on': True,
            'tools.staticdir.root': os.path.abspath(os.getcwd())
        },
        '/static': {
            'tools.staticdir.on': True,
            'tools.staticdir.dir': './static'
        },
    }
    srv = SimpleServer()
    cherrypy.quickstart(srv, '/', config)
```

后端框架

- 总结
  - Django作为自带模版渲染、数据库管理、登陆验证等一系列功能的Web开发框架，非常适合于快速开发有功能具有一定复杂性的应用系统。
  - Flask、Cherrypy相比于Django，框架较为轻量化，框架本身不具有Django那样全面的功能，需要依赖其他功能插件或自己编写代码来实现。因此，在需求功能比较简单或者希望自由度比较高的情况下，推荐用这类框架进行快速开发。

后端框架

- 附录：功能插件推荐

功能需求	名称	介绍	备注
模版渲染	jinja2		Django自动模版渲染
数据库管理	SQLAlchemy		Django自带orm
登陆认证	Python_jwt	基于JWT的登陆认证	Django自带登陆认证
任务调度	Celery	异步任务队列，支持实时操作和任务调度	
日志记录	Logging	Python自带的日志库	Django自带日志记录
配置管理	Configparser	Python自带的针对INI文件的解析库	Django自带配置管理
	Json	Python自带的针对JSON的解析库	
	pyyaml	针对yaml文件的解析库	

- awesome python 项目
  - 寻找需要的python第三方库

道德经

知人者智，自知者明。胜人者有力，自胜者强。知足者富。强行者有志。不失其所者久。死而不亡者，寿。

谢谢！