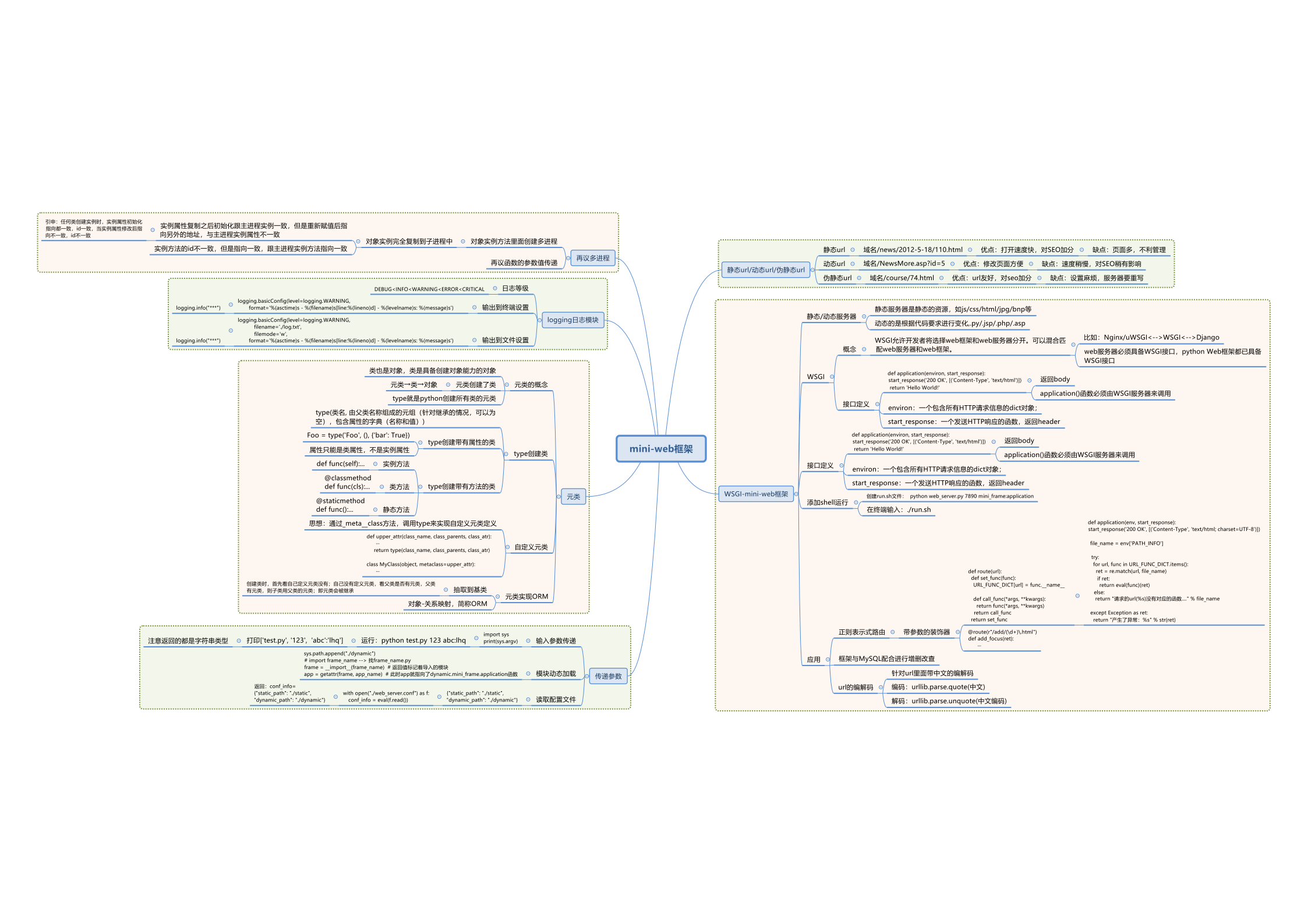
mini-web框架
再议多进程
引申：任何类创建实例时，实例属性初始化指向都一致，id一致，当实例属性修改后指向不一致，id不一致
实例属性复制之后初始化跟主进程实例一致，但是重新赋值后指向另外的地址，与主进程实例属性不一致
实例方法的id不一致，但是指向一致，跟主进程实例方法指向一致
对象实例完全复制到子进程中
对象实例方法里面创建多进程
再议函数的参数值传递
logging日志模块
日志等级
DEBUG<INFO<WARNING<ERROR<CRITICAL
输出到终端设置
logging.basicConfig(level=logging.WARNING, format='%(asctime)s - %(filename)s[line:%(lineno)d] - %(levelname)s: %(message)s')
logging.info("****")
输出到文件设置
logging.basicConfig(level=logging.WARNING, filename='./log.txt', filemode='w', format='%(asctime)s - %(filename)s[line:%(lineno)d] - %(levelname)s: %(message)s')
元类
元类的概念
类也是对象，类是具备创建对象能力的对象
元类→类→对象
元类创建了类
type就是python创建所有类的元类
type创建类
type(类名, 由父类名称组成的元组（针对继承的情况，可以为空），包含属性的字典（名称和值））
type创建带有属性的类
Foo = type('Foo', (), {'bar': True})
属性只能是类属性，不是实例属性
type创建带有方法的类
def func(self):... 实例方法
@classmethod def func(cls):... 类方法
@staticmethod def func():... 静态方法
自定义元类
思想：通过_meta__class方法，调用type来实现自定义元类定义
def upper_attr(class_name, class_parents, class_atr): ... return type(class_name, class_parents, class_atr)
class MyClass(object, metaclass=upper_attr): ...
元类实现ORM
抽取到基类
创建类时，首先看自己定义元类没有；自己没有定义元类，看父类是否有元类，父类有元类，则子类用父类的元类；即元类会被继承
对象-关系映射，简称ORM
传递参数
注意返回的都是字符串类型
打印['test.py', '123', 'abc':'lhq']
运行：python test.py 123 abc:lhq
输入参数传递
import sys print(sys.argv)
模块动态加载
sys.path.append("./dynamic") # import frame_name --> 找frame_name.py frame = __import__(frame_name) # 返回值标记着导入的模块 app = getattr(frame, app_name) # 此时app就指向了dynamic.mini_frame.application函数
读取配置文件
with open("./web_server.conf") as f: conf_info = eval(f.read())
{"static_path": "./static", "dynamic_path": "./dynamic"}
返回：conf_info={"static_path": "./static", "dynamic_path": "./dynamic"}
静态url/动态url/伪静态url
静态url 域名/news/2012-5-18/110.html 优点：打开速度快，对SEO加分 缺点：页面多，不利管理
动态url 域名/NewsMore.asp?id=5 优点：修改页面方便 缺点：速度稍慢，对SEO稍有影响
伪静态url 域名/course/74.html 优点：url友好，对seo加分 缺点：设置麻烦，服务器要重写
WSGI-mini-web框架
静态/动态服务器
静态服务器是静态的资源，如js/css/html/jpg/bnp等
动态的是根据代码要求进行变化,.py/.jsp/.php/.asp
WSGI
概念
WSGI允许开发者将选择web框架和web服务器分开。可以混合匹配web服务器和web框架。
比如：Nginx/uWSGI<-->WSGI<-->Django
web服务器必须具备WSGI接口，python Web框架都已具备WSGI接口
接口定义
def application(environ, start_response): start_response('200 OK', [('Content-Type', 'text/html')]) return 'Hello World!'
返回body
application()函数必须由WSGI服务器来调用
environ：一个包含所有HTTP请求信息的dict对象；
start_response：一个发送HTTP响应的函数，返回header
接口定义
def application(environ, start_response): start_response('200 OK', [('Content-Type', 'text/html')]) return 'Hello World!'
返回body
application()函数必须由WSGI服务器来调用
environ：一个包含所有HTTP请求信息的dict对象；
start_response：一个发送HTTP响应的函数，返回header
添加shell运行
创建run.sh文件： python web_server.py 7890 mini_frame:application
在终端输入：./run.sh
应用
正则表示式路由
带参数的装饰器
@route(r"/add/(\d+)\.html") def add_focus(ret): ...
def route(url): def set_func(func): URL_FUNC_DICT[url] = func.__name__ def call_func(*args, **kwargs): return func(*args, **kwargs) return call_func return set_func
def application(env, start_response): start_response('200 OK', [('Content-Type', 'text/html; charset=UTF-8')]) file_name = env['PATH_INFO'] try: for url, func in URL_FUNC_DICT.items(): ret = re.match(url, file_name) if ret: return eval(func)(ret) else: return "请求的url(%s)没有对应的函数...." % file_name except Exception as ret: return "产生了异常：%s" % str(ret)
框架与MySQL配合进行增删改查
url的编解码
针对url里面带中文的编解码
编码：urllib.parse.quote(中文)
解码：urllib.parse.unquote(中文编码)