Web.py 框架调研

目录

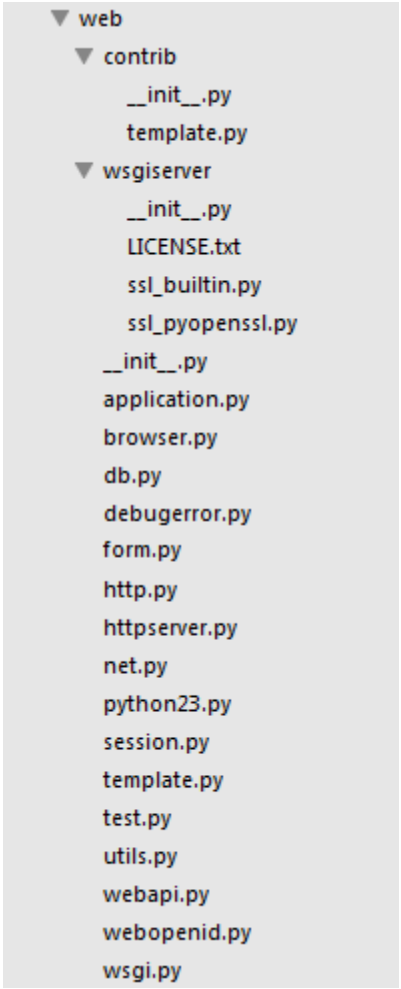
1.概述

Web.py 是一个开源的，用 python 语言写成的 web 框架，既简单又非常强大，你可以应用到任何地方，而没有任何限制。Web.py 框架既有应用框架，也包含了 Web server 服务，可以非常简单、方便、快速的搭建应用。

2.web.py 架构设计

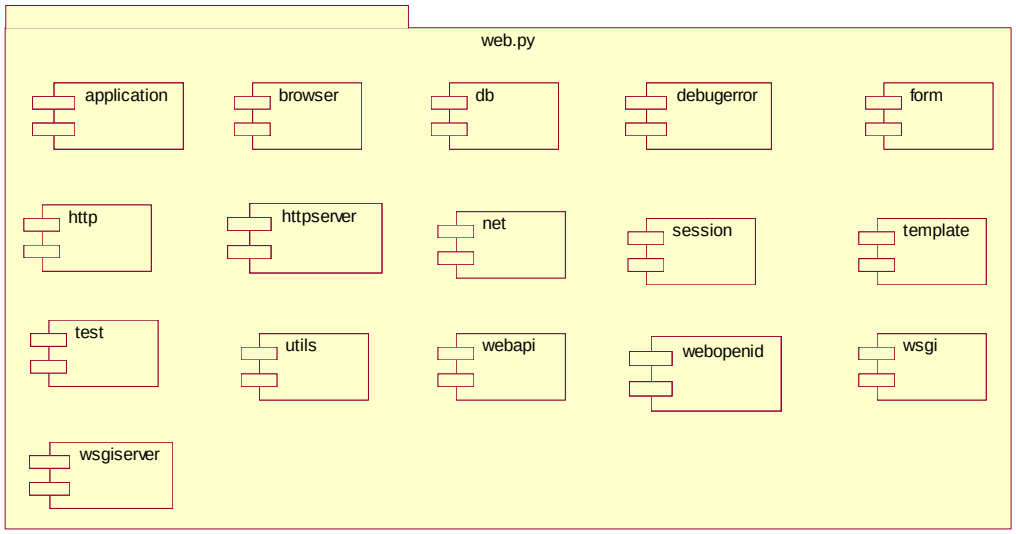
2.1 框架目录

Web.py 框架目录结构如下图所示，每一个目录其实就是一个模块。

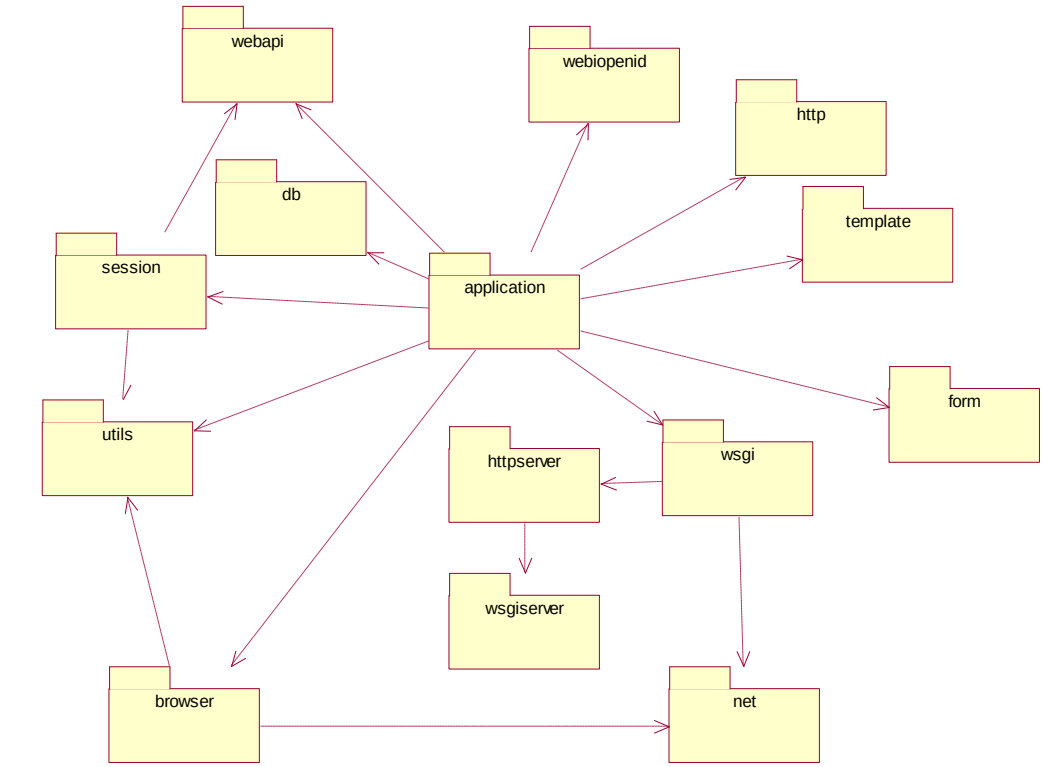


2.2 框架模块

2.2.1 框架组件图



2.2.2 框架模块类图

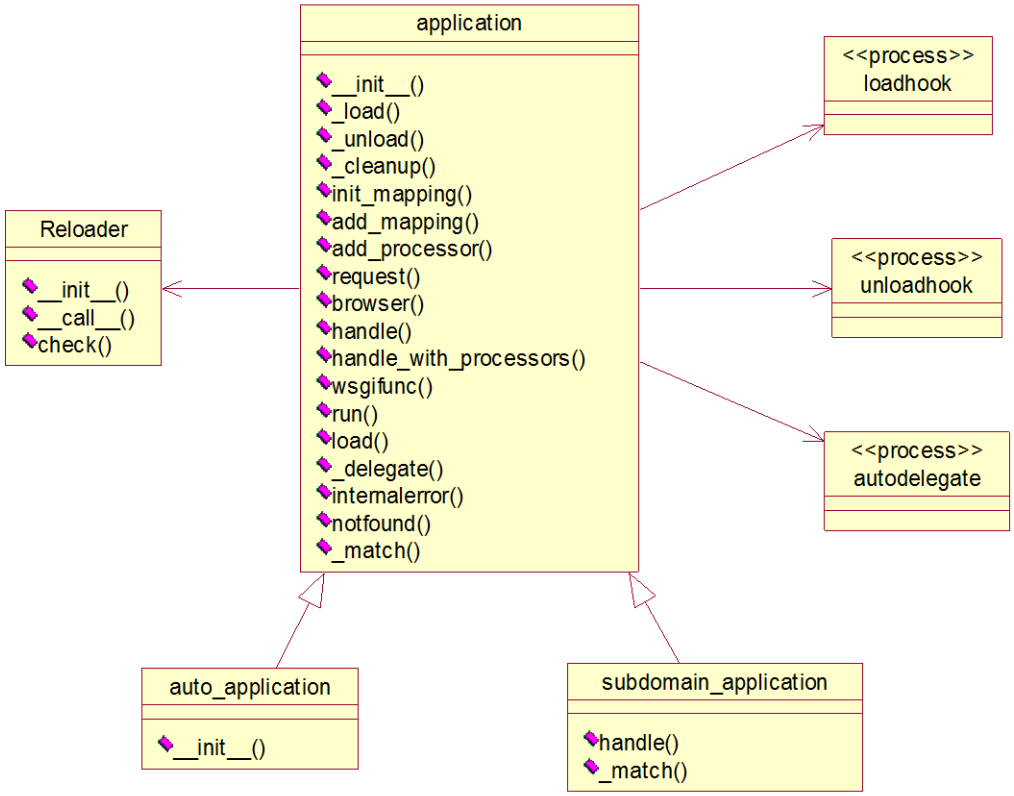


2.2.3 主要功能模块

2.2.3.1 application 模块

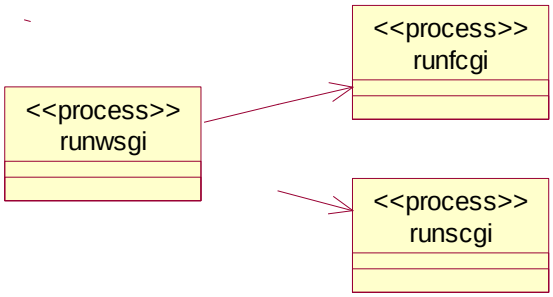
该模块主要是委托处理请求，初始化、注册所需组件等，同时会根据解析的请求路由到相应的 handler 进行处理。

该模块类图如下所示：



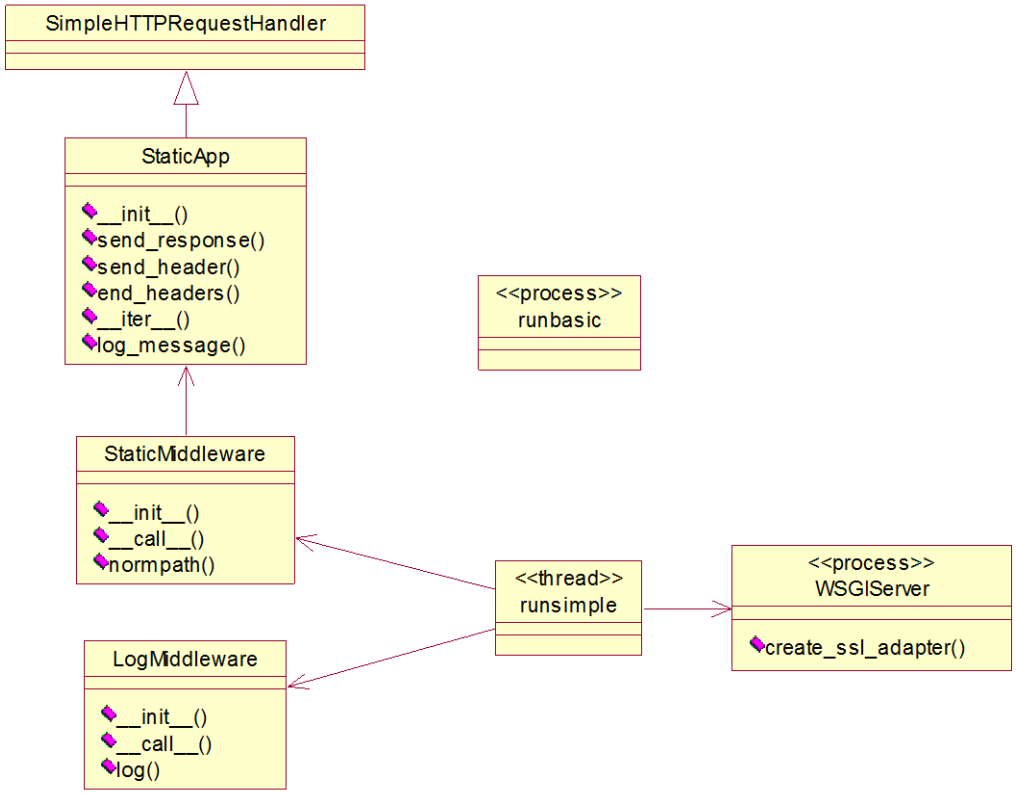
2.2.3. 2 wsgi 模块

该模块是 websever 工具类，根据 webserver 运行方式的不同做相应的处理，同时支持 FastCGI 和 SCGI 等运行方式，模块类图如下所示，这里在 python 里一切皆为对象，所以这里的函数也用类图表示：



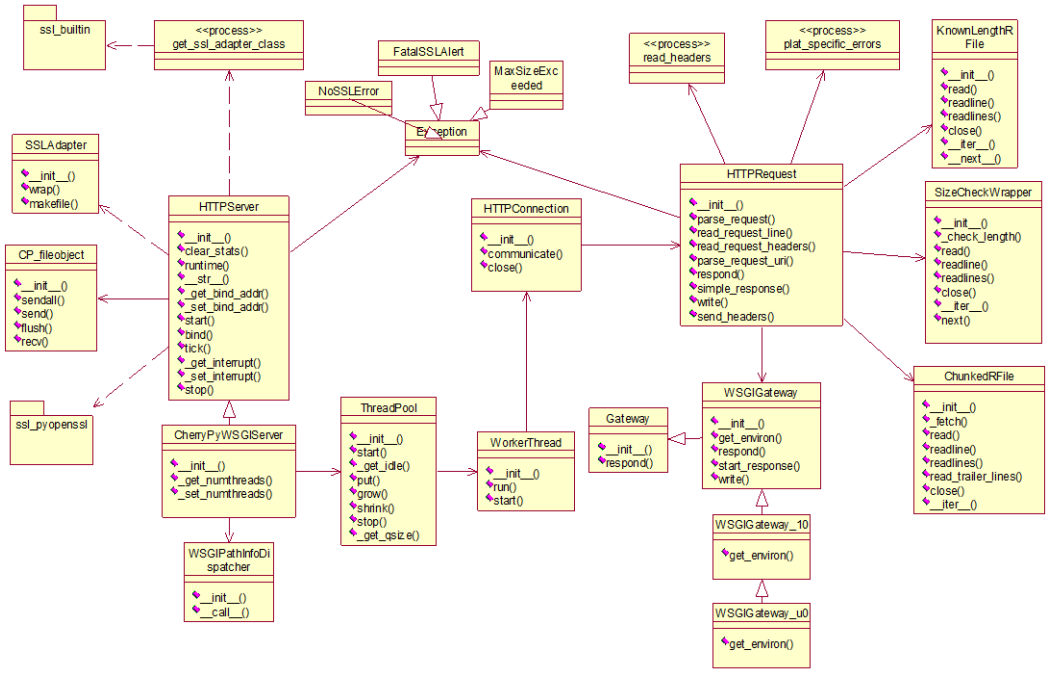
2.2.3.3 httpserver 模块

此模块封装了运行并初始化 server 的逻辑，同时做好日志记录及对静态资源访问的逻辑等，类图如下：



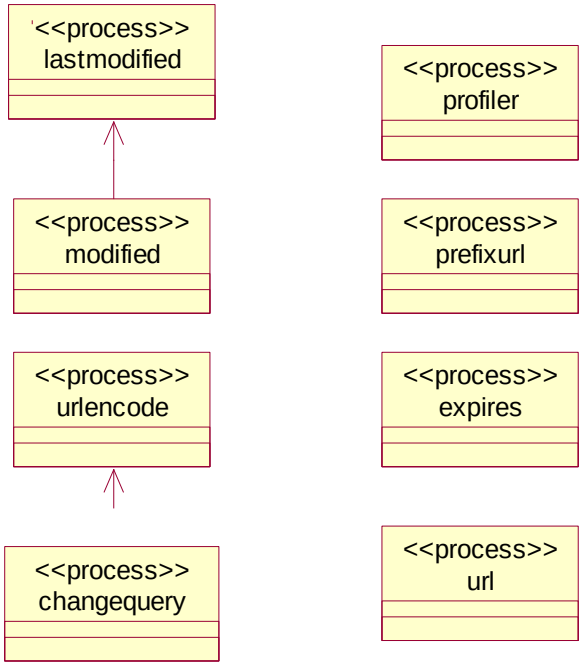
2.2.3.4 wsgiserver 模块

该模块封装了底层的 socket 通信，做为简单的 server 运行，持续监听指定的端口并接收请求，而由 WorkerThread 处理请求连接通信等并做出相应的响应处理，类图如下所示：



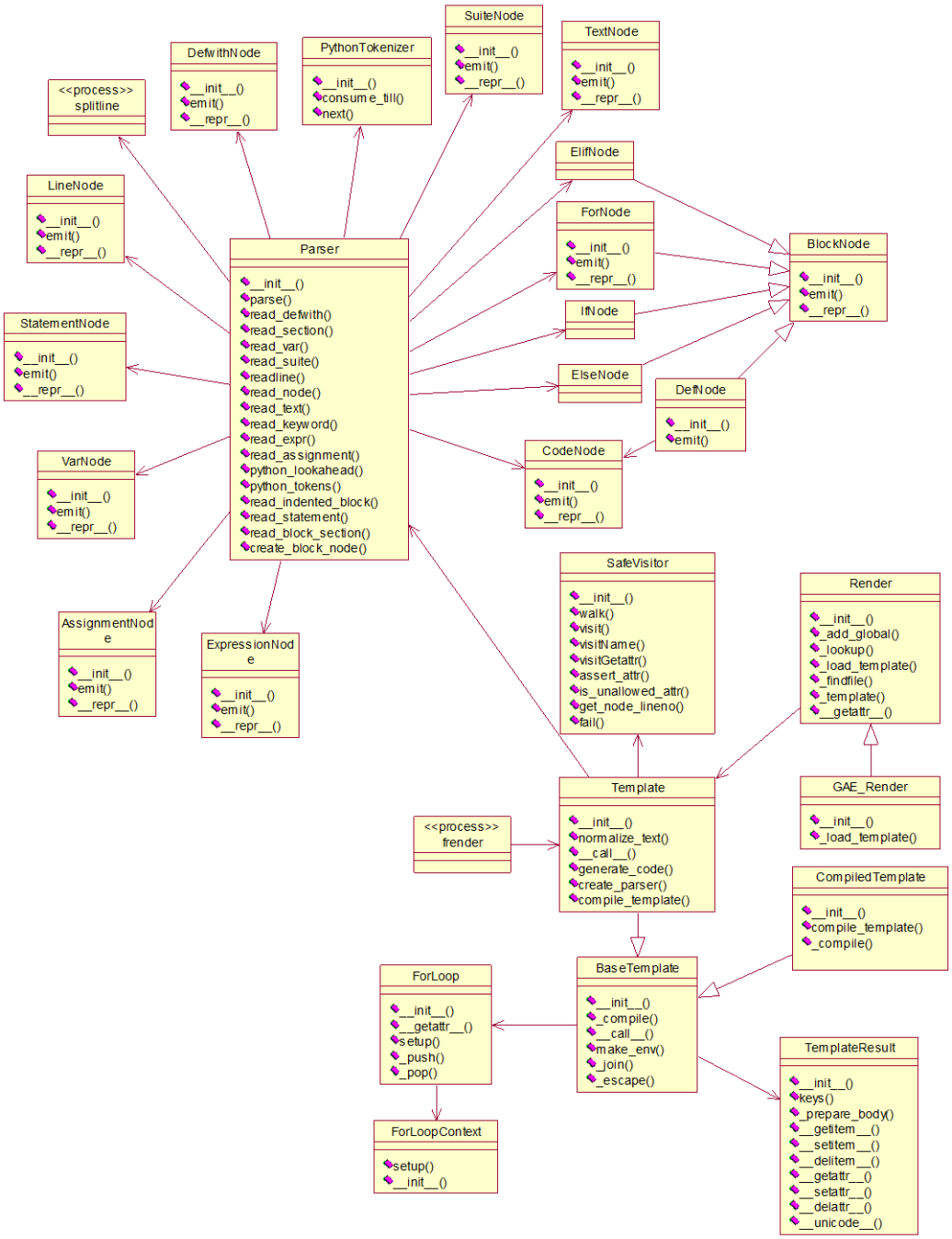
2.2.3.5 http 模块

该模块提供了处理 http 请求的一些工具，针对 expires、lastmodified、modified、url 等进行处理，被整体框架所用，类图如下所示：



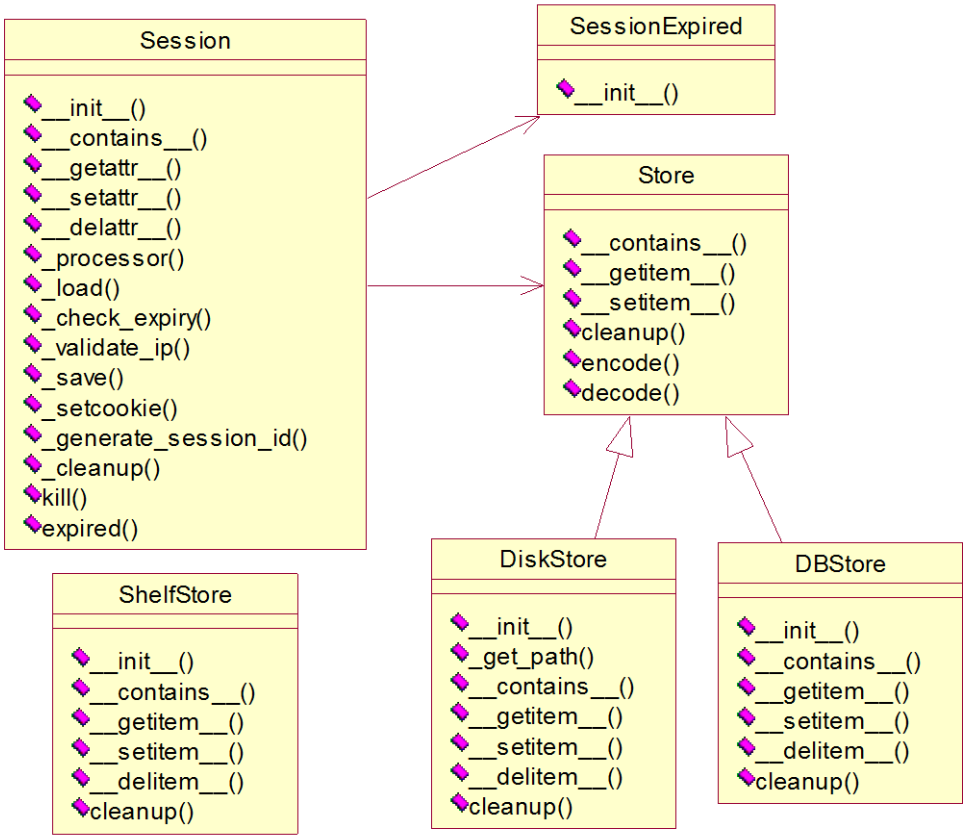
2.2.3.6 template 模块

该模块提供了解析模板的引擎，会根据一定的语法把模板解析成由一个个不同的结点组成的树形结构，并通过 python 运行返回模板字符串显示，类图如下所示：



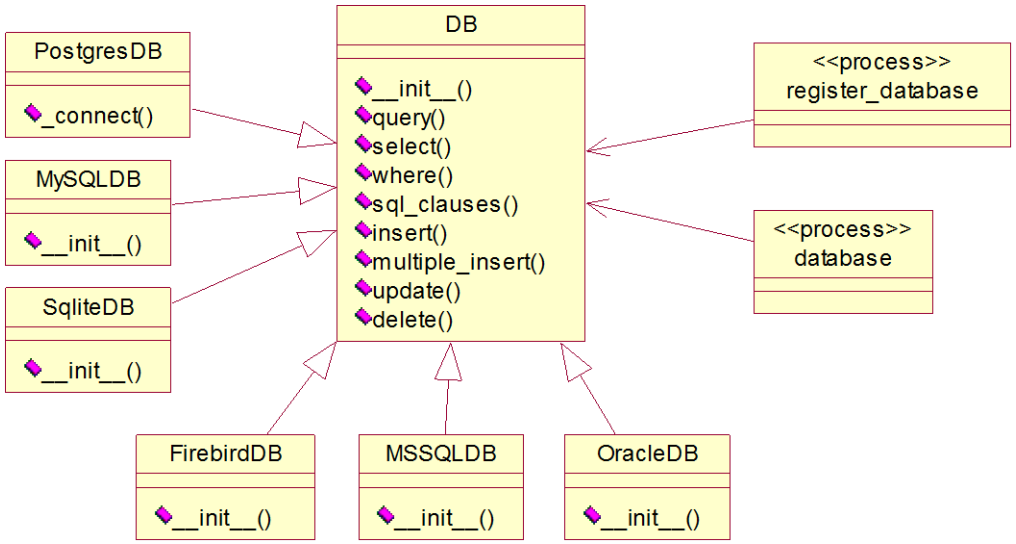
2.2.3.7 session 模块

此模块提供对 session 的管理，包括添加、删除、过期、保存等等处理，类图如下所示：



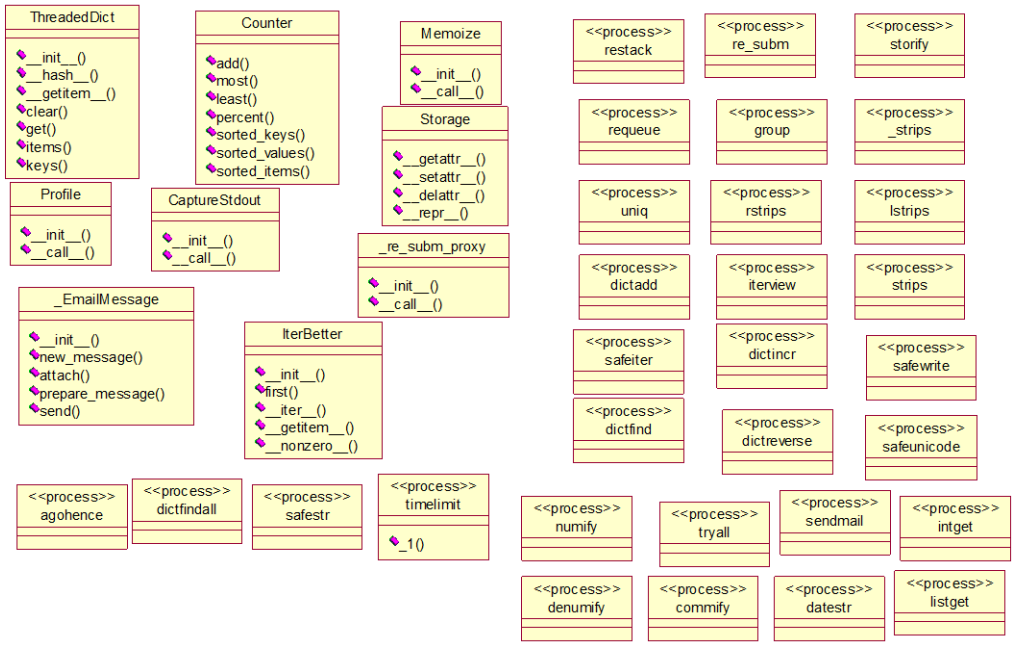
2.2.3.8 db 模块

封装了针对数据库的一些操作，也可以自定义写 Sql 语句查询，整体类图如下所示：



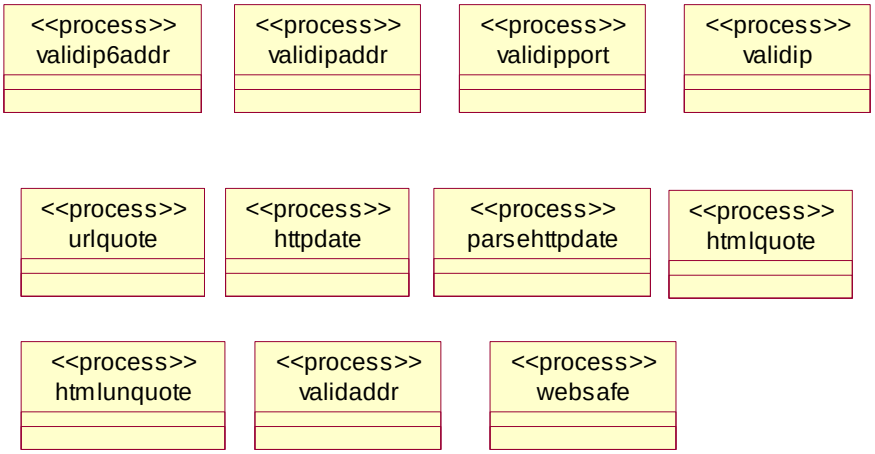
2.2.3.9 utils 模块

该模块提供框架的一般的工具，比如 kv 存储、计数、列表处理、类型转换、迭代器、邮件等操作工具，其类图如下所示：



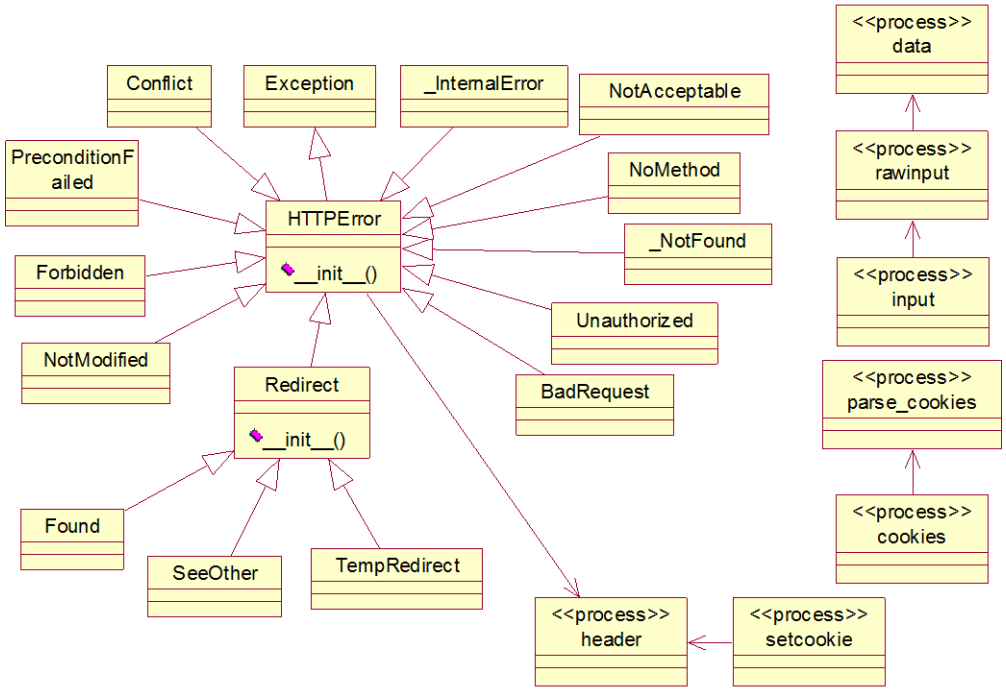
2.2.3.10 net 模块

网络工具模块，提供各种网络操作工具，如：验证 IP 的正确性、验证端口的正确性、编码 html 等及解析格式化日志、url 等，工具函数都用类来表示，其类图如下所示：



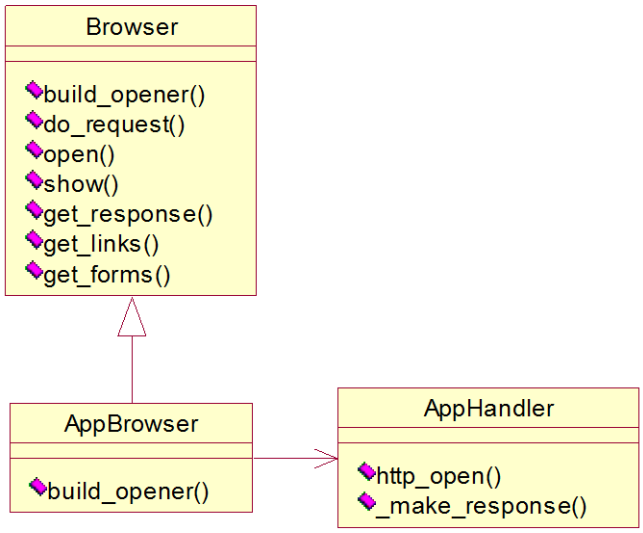
2.2.3.11 webapi 模块

提供 web 服务的工具，包括对请求数据及 cookie 和环境变量的操作，同时封装了各种 http 错误码等，其类图如下所示：



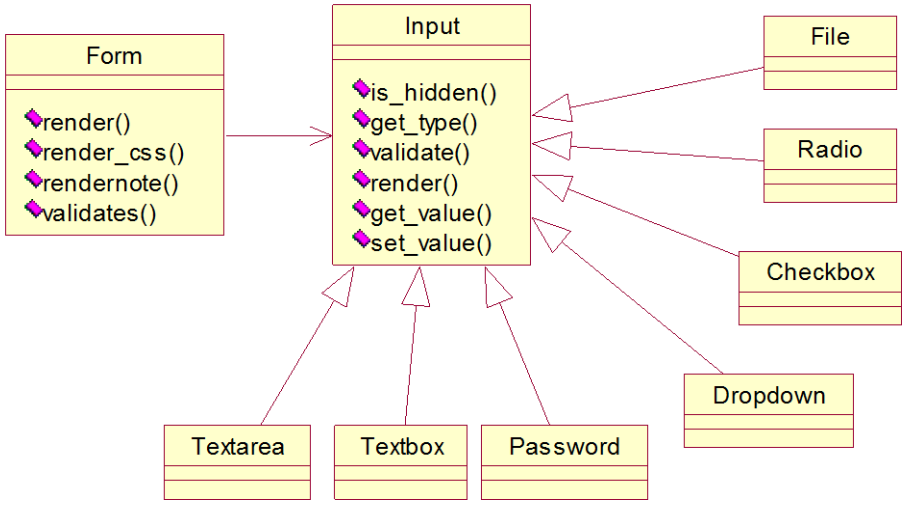
2.2.3.12 browser 模块

该模块主要用于测试 web 应用程序，相当于单元测试模块等，如下类图所示：



2.2.3.13 form 模块

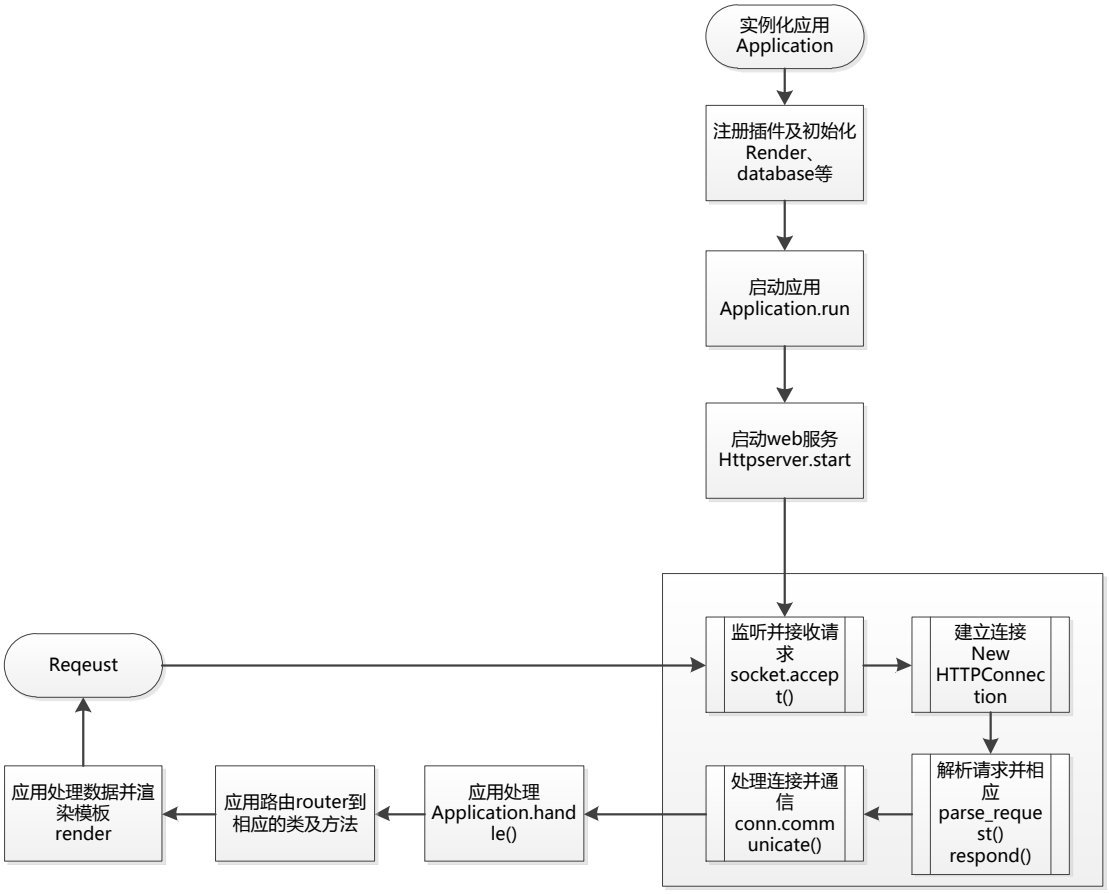
该模块提供了针对表单的操作，根据其封装的工具不用写 html 即可显示操作表单等，其类图如下所示：



3.web.py 处理流程

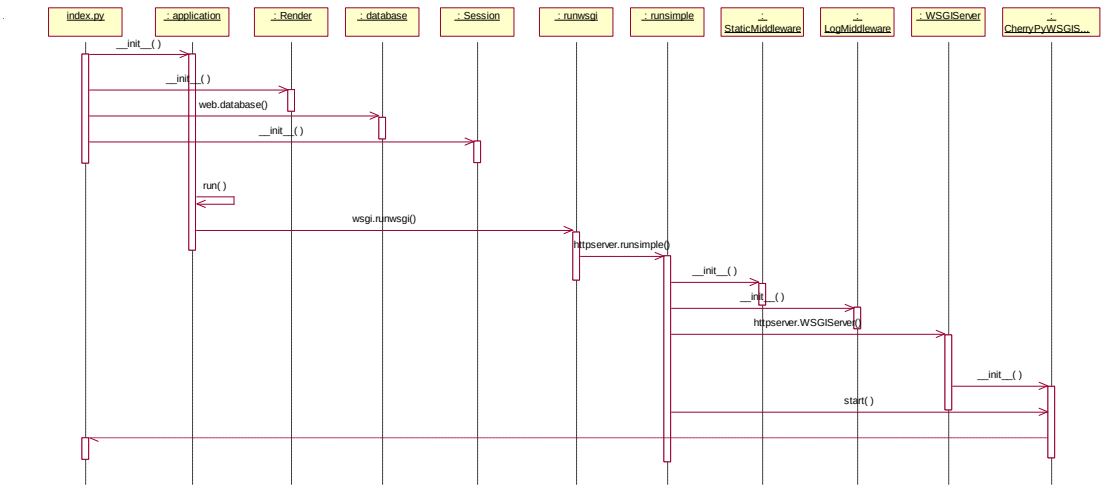
3.1 框架整体处理流程

Web.py 框架自带 web 服务，基本是对 socket 通信的封装，非常强大灵活，可指定绑定端口，同时也支持 fastcgi 方式等运行，循环接收请求并解析路由处理等，流程图如下所示：



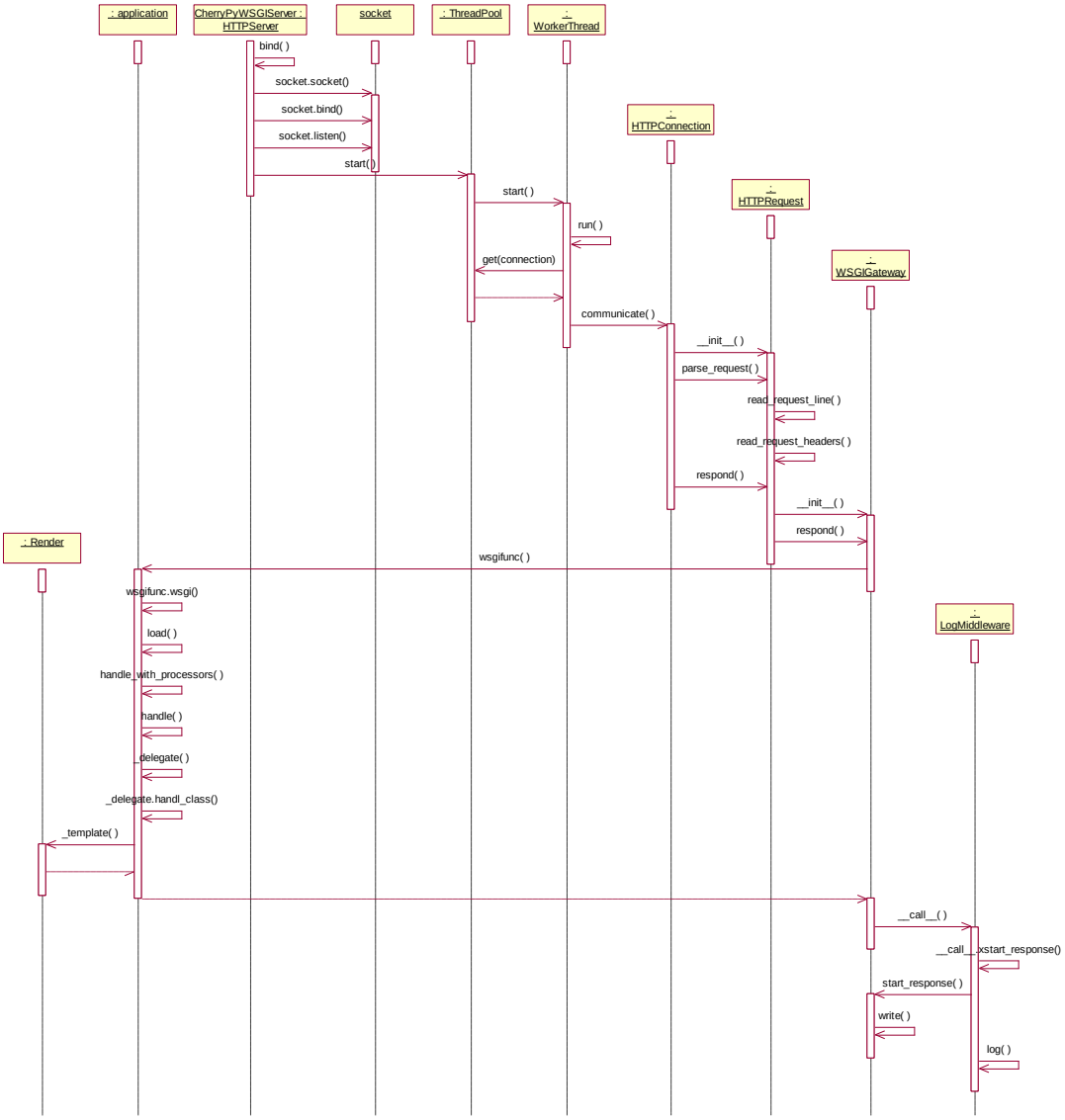
3.2 框架 webserver 启动运行流程

服务根据指定的 IP 及端口进行启动，会有线程循环监听，接收进来的连接到队列里，workthread 线程从队列里取出一个连接来进行具体的通信，并处理相应的请求，最后返回响应结果，如下图所示：



3.3 框架请求处理流程

Webserver 在接收到请求后，会另起一工作线程去处理请求，对请求数据及环境参数做处理后传给应用，应用程序根据请求数据进行路由，并通过相应的处理器去处理数据，最后返回处理结果，同时会进行模板的解析及渲染等，具体流程如下图所示：



3.4 web.py 框架与 nginx 的结合

Web.py 框架当然也可以做为 fastcgi 方式运行，并与 nginx web server 结合使用。
首先配置并启动 nginx，如下示例代码所示：

server {

```
        ##监听端口
        listen                  8600;
        server_name             dbl-wsie-rdtest10.vm.baidu.com;
        more_set_headers        'Server: Apache';
        ##错误处理
        error_page      400     403     404     500     501     502     503     504     505
http://www.baidu.com/search/error.html;
        ##应用根目录
        root /home/work/workspaces/python_dev/bd3600_log;
        ##访问日志文件
        access_log /home/work/workspaces/log/python_web_access.log main;
        location / {
            fastcgi_pass       127.0.0.1:8900;
            include             fastcgi_params;
            include             fastcgi.conf;
        }
        ##静态资源直接访问根目录 static 下相应的文件
        location /static/ {
            rewrite ^/static/(.*)$    /static/$1 break;
        }
}
```

启动 nginx，在 8600 端口访问。

其次以 fastcgi 的方式运行 web.py 框架，启动 fastcgi：

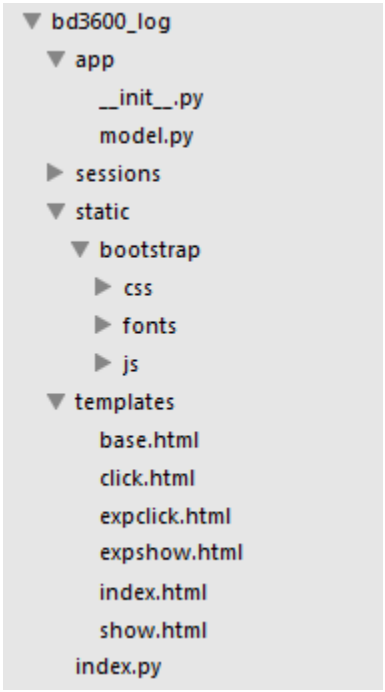
python index.py 8900 fastcgi

之后就可以在 8600 端口访问应用了。

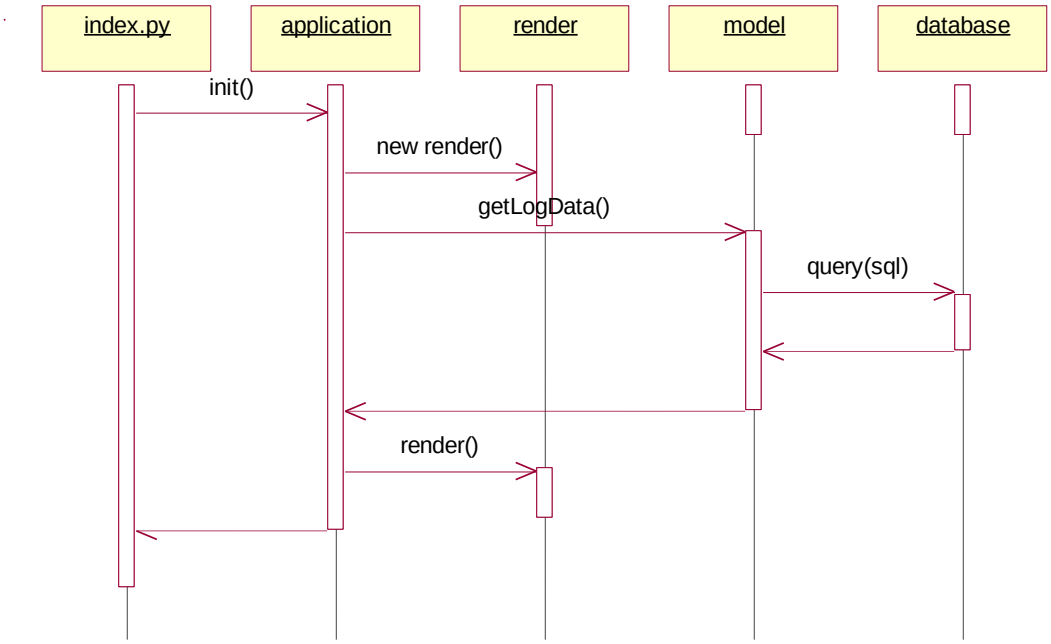
4.实例

用 web.py 创建一应用 bd3600_log，简单的根据 3600 日志数据，生成几个页面报表。

4.1 应用目录结构



4.2 应用运行流程



Index.py 是入口文件，运行该文件，通过里面的 application.run()函数即可运行应用程序，同时会启动 web server 来接收请求，具体代码如下所示：

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import web
import logging
from app import model

###这里配置相应的路由处理器
urls = (
    '/', 'index',
    '/show', 'show',
    '/click', 'click',
)
###实例化一个应用
app = web.application(urls, locals())
###初始化模板全居变量
gbs={
    'session':session,
    'nav':1,
    }
```

```
###实例化模板引擎
render = web.template.render('templates/', base='base', globals=gbs)
###实例化数据库查询对象
db = web.database(host='fe.baidu.com', dbn='mysql', user='root', pw='', db='wise_middle')

###以下是相应的路由及相应的 action
class index:
    def GET(self):
        gbs['nav'] = 1;
        return render.index()

class show:
    def GET(self):
        logData = model.getShowLog()
        gbs['nav'] = 2;
        return render.show(logData)

class click:
    def GET(self):
        logData = model.getClickLog()
        gbs['nav'] = 3;
        return render.click(logData);
###运行该应用程序
if __name__ == "__main__":
    app.run()
```

5.总结

Web.py 框架整体简单灵活，适合熟悉 python 开发程序员快速搭建简单的 web 应用，目录结构清楚，可以自由组织，代码也相当的清淅明了，有完整的日志、数据库、请求及响应处理模块，同时包含了各种完善的工具类供其使用，开发人员也可以方便的开发自己的组件做为其一部分扩展包导入使用。

但不足的是感觉模板引擎方面不是特别好，模板变量的使用，及一些控制逻辑等不是特别方便，但总体感觉还是不错了，强大、灵活、简单是给我的感觉。

6.参考资料

Web.py 官网：http://webpy.org/

Web.py 源码：https://github.com/webpy/
Python 官网：https://www.python.org/