20.1 mini web 框架-4-路由

20.1.1 使用字典替换 if 判断

在上一章中，访问 index.py，我们判断 filename 是否等于 index.py，判断 center.py，我们需要通过下面的代码各种 if，elif 来进行判断，如果页面多，将会造成非常多的 elif 判断，有没有办法解决？

```
if file_name == "/index.py":
    return index()
elif file_name == "/center.py":
    return center()
```

我们采用配置文件，其实 Java 的三大框架很多都是通过配置，从而让初学者能够快速得到自己想要的效果，少写代码，同时也降低了出错的概率。

我们定义

```
URL_FUNC_DICT = {
    "/index.py": index,
    "/center.py": center
}
```

在 application 函数中，编写

```
func=URL_FUNC_DICT[file_name]
return func()
```

运行，同样会实现对应的效果

20.1.2 使用装饰器实现路由功能

上面通过字典的方式非常方便，如果我们能够让字典自动生成是不是就更爽了？接下来我们来看 mini_frame.py 的代码，通过装饰器来实现 URL_FUNC_DICT 字典的生成

mini_frame.py

```
import re
# 用来存放 url 路由映射
URL_FUNC_DICT = dict()


def route(url):
   def set_func(func):
```

```
        # URL_FUNC_DICT["/index.py"] = index
        URL_FUNC_DICT[url] = func

    # def call_func(*args, **kwargs):
    #     return func(*args, **kwargs)
    # return call_func

    return set_func


@route("/index.py")
def index():
    with open("./templates/index.html", encoding="utf-8") as f:
        content = f.read()

    my_stock_info = "哈哈哈哈 这是你的本月名称....."

    content = re.sub(r"\{%content%\}", my_stock_info, content)

    return content


@route("/center.py")
def center():
    with open("./templates/center.html", encoding="utf-8") as f:
        content = f.read()

    my_stock_info = "这里是从 mysql 查询出来的数据。。。"

    content = re.sub(r"\{%content%\}", my_stock_info, content)

    return content


def application(env, start_response):
    start_response('200 OK', [('Content-Type', 'text/html;charset=utf-
8')])

    file_name = env['PATH_INFO']
    # file_name = "/index.py"
```

```
    # if file_name == "/index.py":
    #     return index()
    # elif file_name == "/center.py":
    #     return center()
    # else:
    #     return 'Hello World! 我爱你中国....'
    try:
        func = URL_FUNC_DICT[file_name]
        return func()
        # return URL_FUNC_DICT[file_name]()
    except Exception as ret:
        return "产生了异常：%s" % str(ret)
```

20.2 伪静态、静态和动态的区别

目前开发的网站其实真正意义上都是动态网站，只是 URL 上有些区别，一般 URL 分为静态 URL、动态 URL、伪静态 URL，他们的区别是什么?

20.2.1 静态 URL

静态 URL 类似 域名/news/2012-5-18/110.html 我们一般称为真静态 URL，每个网页有真实的物理路径，也就是真实存在服务器里的。

- 优点是：

  网站打开速度快，因为它不用进行运算；另外网址结构比较友好，利于记忆。

- 缺点是：

  最大的缺点是如果是中大型网站，则产生的页面特别多，不好管理。至于有的开发者说占用硬盘空间大，我觉得这个可有忽略不计，占用不了多少空间的，况且目前硬盘空间都比较大。还有的开发者说会伤硬盘，这点也可以忽略不计。

- 一句话总结:

  静态网站对 SEO 的影响：静态 URL 对 SEO 肯定有加分的影响，因为打开速度快，这个是本质。

20.2.2 动态 URL

动态 URL 类似 域名/NewsMore.asp?id=5 或者 域名/DaiKuan.php?id=17，带有？号的 URL，我们一般称为动态网址，每个 URL 只是一个逻辑地址，并不是真实物理存在服务器硬盘里的。

- 优点是：

  适合中大型网站，修改页面很方便，因为是逻辑地址，所以占用硬盘空间要比纯静态网站小。

- 缺点是：

  因为要进行运算，所以打开速度稍慢，不过这个可有忽略不计，目前有服务器缓存技术可以解决速度问题。最大的缺点是 URL 结构稍稍复杂，不利于记忆。

- 一句话总结:

  动态 URL 对 SEO 的影响：目前百度 SE 已经能够很好的理解动态 URL，所以对 SEO 没有什么减分的影响（特别复杂的 URL 结构除外）。所以你无论选择动态还是静态其实都无所谓，看你选择的程序和需求了。

20.2.3 伪静态 URL

伪静态 URL 类似 域名/course/74.html 这个 URL 和真静态 URL 类似。他是通过伪静态规则把动态 URL 伪装成静态网址。也是逻辑地址，不存在物理地址。

- 优点是：

  URL 比较友好，利于记忆。非常适合大中型网站，是个折中方案。

- 缺点是：

  设置麻烦，服务器要支持重写规则，小企业网站或者玩不好的就不要折腾了。另外进行了伪静态网站访问速度并没有变快，因为实质上它会额外的进行运算解释，反正增加了服务器负担，速度反而变慢，不过现在的服务器都很强大，这种影响也可以忽略不计。还有可能会造成动态 URL 和静态 URL 都被搜索引擎收录，不过可以用 robots 禁止掉动态地址。

- 一句话总结:

  对 SEO 的影响：和动态 URL 一样，对 SEO 没有什么减分影响。

20.3 mini-web 框架-实现伪静态 url

readme.txt

运行方式如下:

```
python3 web_server.py 7890 mini_frame:application
```

web_server.py(部分更新)

```
 import socket
import re
import multiprocessing
import time
# import dynamic.mini_frame
import sys


class WSGIServer(object):
    def __init__(self, port, app, static_path):
        # 1. 创建套接字
        self.tcp_server_socket = socket.socket(socket.AF_INET,
socket.SOCK_STREAM)
        self.tcp_server_socket.setsockopt(socket.SOL_SOCKET,
socket.SO_REUSEADDR, 1)

        # 2. 绑定
        self.tcp_server_socket.bind(("", port))

        # 3. 变为监听套接字
        self.tcp_server_socket.listen(128)

        self.application = app
        self.static_path = static_path

    def service_client(self, new_socket):
        """为这个客户端返回数据"""
```

```
        # 1. 接收浏览器发送过来的请求 , 即 http 请求
        # GET / HTTP/1.1
        # .....
        request = new_socket.recv(1024).decode("utf-8")
        # print(">>>"*50)
        # print(request)
        if request:
            request_lines = request.splitlines()
            print("")
            print(">"*20)
            print(request_lines)

            # GET /index.html HTTP/1.1
            # get post put del
            file_name = ""
            ret = re.match(r"[^/]+(/[^ ]*)", request_lines[0])
            if ret:
                file_name = ret.group(1)
                # print("*"*50, file_name)
                if file_name == "/":
                    file_name = "/index.html"

            # 2. 返回 http 格式的数据，给浏览器
            # 2.1 如果请求的资源不是以.html 结尾，那么就认为是静态资源（css/js/png，jpg 等）--更新了这里
            if not file_name.endswith(".html"):
                try:
                    f = open(self.static_path + file_name, "rb")
                except:
                    response = "HTTP/1.1 404 NOT FOUND\r\n"
                    response += "\r\n"
                    response += "------file not found-----"
                    new_socket.send(response.encode("utf-8"))
                else:
                    html_content = f.read()
                    f.close()
                    # 2.1 准备发送给浏览器的数据---header
                    response = "HTTP/1.1 200 OK\r\n"
                    response += "\r\n"
                    # 2.2 准备发送给浏览器的数据---boy
```

```
                    # response += "hahahhah"

                    # 将 response header 发送给浏览器
                    new_socket.send(response.encode("utf-8"))
                    # 将 response ic.mini_frame.applicationbody 发送给浏览器
                    new_socket.send(html_content)
            else:
                # 2.2 如果是以.py 结尾，那么就认为是动态资源的请求

                env = dict()  # 这个字典中存放的是 web 服务器要传递给 web 框架的数据信息
                env['PATH_INFO'] = file_name
                # {"PATH_INFO": "/index.py"}
                # body = dynamic.mini_frame.application(env, self.set_response_header)
                body = self.application(env, self.set_response_header)

                header = "HTTP/1.1 %s\r\n" % self.status

                for temp in self.headers:
                    header += "%s:%s\r\n" % (temp[0], temp[1])

                header += "\r\n"

                response = header+body
                # 发送 response 给浏览器
                new_socket.send(response.encode("utf-8"))


        # 关闭套接
        new_socket.close()

    def set_response_header(self, status, headers):
        self.status = status
        self.headers = [("server", "mini_web v8.8")]
        self.headers += headers


    def run_forever(self):
```

```
        """用来完成整体的控制"""

        while True:
            # 4. 等待新客户端的链接
            new_socket, client_addr = self.tcp_server_socket.accept()

            # 5. 为这个客户端服务
            p = multiprocessing.Process(target=self.service_client,
args=(new_socket,))
            p.start()

            new_socket.close()


        # 关闭监听套接字
        self.tcp_server_socket.close()


def main():
    """控制整体，创建一个 web 服务器对象，然后调用这个对象的 run_forever
方法运行"""
    if len(sys.argv) == 3:
        try:
            port = int(sys.argv[1])  # 7890
            frame_app_name = sys.argv[2]  # mini_frame:application
        except Exception as ret:
            print("端口输入错误。。。。。")
            return
    else:
        print("请按照以下方式运行:")
        print("python3 xxxx.py 7890 mini_frame:application")
        return

    # mini_frame:application
    ret = re.match(r"([^:]+):(.*)", frame_app_name)
    if ret:
        frame_name = ret.group(1)  # mini_frame
        app_name = ret.group(2)  # application
    else:
        print("请按照以下方式运行:")
```

```
        print("python3 xxxx.py 7890 mini_frame:application")
        return

    with open("./web_server.conf") as f:
        conf_info = eval(f.read())

    # 此时 conf_info 是一个字典里面的数据为:
    # {
    #     "static_path":"./static",
    #     "dynamic_path":"./dynamic"
    # }


    sys.path.append(conf_info['dynamic_path'])

    # import frame_name --->找 frame_name.py
    frame = __import__(frame_name)  # 返回值标记这 导入的这个模板
    app = getattr(frame, app_name)  # 此时 app 就指向了
dynamic/mini_frame 模块中的 application 这个函数

    # print(app)

    wsgi_server = WSGIServer(port, app, conf_info['static_path'])
    wsgi_server.run_forever()


if __name__ == "__main__":
    main()
```

mini_frame.py

```
import re

URL_FUNC_DICT = dict()


def route(url):
   def set_func(func):
      # URL_FUNC_DICT["/index.py"] = index
      URL_FUNC_DICT[url] = func
```

```
    # def call_func(*args, **kwargs):
    #     return func(*args, **kwargs)
    # return call_func

    return set_func


@route("/index.html") #--更新了这里
def index():
    with open("./templates/index.html", encoding="utf-8") as f:
        content = f.read()

    my_stock_info = "哈哈哈哈 这是你的本月名称....."

    content = re.sub(r"\{%content%\}", my_stock_info, content)

    return content


@route("/center.html") #--更新了这里
def center():
    with open("./templates/center.html", encoding="utf-8") as f:
        content = f.read()

    my_stock_info = "这里是从 mysql 查询出来的数据。。。"

    content = re.sub(r"\{%content%\}", my_stock_info, content)

    return content


def application(env, start_response):
    start_response('200 OK', [('Content-Type', 'text/html;charset=utf-
8')])

    file_name = env['PATH_INFO']
    # file_name = "/index.py"

    # if file_name == "/index.py":
    #     return index()
```

```
    # elif file_name == "/center.py":
    #     return center()
    # else:
    #     return 'Hello World! 我爱你中国....'
    try:
        func = URL_FUNC_DICT[file_name]
        return func()
        # return URL_FUNC_DICT[file_name]()
    except Exception as ret:
        return "产生了异常：%s" % str(ret)
```

20.4 准备数据

1. 创建数据库

```
create database stock_db charset=utf8;
```

2. 选择数据库

```
use stock_db;
```

3. 导入数据

stock_db.sql 在课件中

```
source stock_db.sql
```

4. 表结构如下

```
mysql> desc focus;
+-----------+------------------+------+-----+---------+----------------
+
| Field     | Type             | Null | Key | Default | Extra
|
+-----------+------------------+------+-----+---------+----------------
+
| id        | int(10) unsigned | NO   | PRI | NULL    | auto_increment
|
| note_info | varchar(200)     | YES  |     |         |
|
| info_id   | int(10) unsigned | YES  | MUL | NULL    |
|
+-----------+------------------+------+-----+---------+----------------
+

mysql> desc info;
+----------+------------------+------+-----+---------+----------------+
| Field    | Type             | Null | Key | Default | Extra          |
```

```
+----------+------------------+------+-----+---------+----------------+
| id       | int(10) unsigned | NO   | PRI | NULL    | auto_increment |
| code     | varchar(6)       | NO   |     | NULL    |                |
| short    | varchar(10)      | NO   |     | NULL    |                |
| chg      | varchar(10)      | NO   |     | NULL    |                |
| turnover | varchar(255)     | NO   |     | NULL    |                |
| price    | decimal(10,2)    | NO   |     | NULL    |                |
| highs    | decimal(10,2)    | NO   |     | NULL    |                |
| time     | date             | YES  |     | NULL    |                |
+----------+------------------+------+-----+---------+----------------+
```

20.5 mini-web 框架-从 mysql 中查询数据

如果想让页面上显示数据库里的数据怎么办，也就是显示如下图的效果

选股系统 | 股票信息 | 个人中心

序号	股票代码	股票简称	涨跌幅	换手率	最新价(元)	前期高点	前期高点日期	添加自选
1	000007	全新好	10.01%	4.40%	16.05	14.60	2017-07-18	添加
2	000036	华联控股	10.04%	10.80%	11.29	10.26	2017-07-20	添加
3	000039	中集集团	1.35%	1.78%	18.07	18.06	2017-06-28	添加
4	000050	深天马A	4.38%	4.65%	22.86	22.02	2017-07-19	添加
5	000056	皇庭国际	0.39%	0.65%	12.96	12.91	2017-07-20	添加
6	000059	华锦股份	3.37%	7.16%	12.26	12.24	2017-04-11	添加
7	000060	中金岭南	1.34%	3.39%	12.08	11.92	2017-07-20	添加
8	000426	兴业矿业	0.41%	2.17%	9.71	9.67	2017-07-20	添加
9	000488	晨鸣纸业	6.30%	5.50%	16.37	15.59	2017-07-10	添加
10	000528	柳工	1.84%	3.03%	9.42	9.33	2017-07-19	添加
11	000540	中天金融	0.37%	5.46%	8.11	8.08	2017-07-20	添加
12	000581	威孚高科	3.49%	3.72%	27.00	26.86	2017-06-26	添加
13	000627	天茂集团	5.81%	12.51%	10.93	10.33	2017-07-20	添加
14	000683	远兴能源	6.42%	21.27%	3.48	3.29	2017-07-19	添加

首先我们需要把数据从数据库查出来，代码如下：

mini_frame.py(更新)

```
import re
from pymysql import connect

"""
```

```
URL_FUNC_DICT = {
    "/index.html": index,
    "/center.html": center
}
"""

URL_FUNC_DICT = dict()


def route(url):
    def set_func(func):
        # URL_FUNC_DICT["/index.py"] = index
        URL_FUNC_DICT[url] = func
        def call_func(*args, **kwargs):
            return func(*args, **kwargs)
        return call_func
    return set_func


@route("/index.html")
def index():
    with open("./templates/index.html",encoding="utf-8") as f:
        content = f.read()

    # my_stock_info = "哈哈哈哈 这是你的本月名称....."
    # content = re.sub(r"\{%content%\}", my_stock_info, content)
    # 创建 Connection 连接
    conn =
connect(host='192.168.0.111',port=3306,user='root',password='123',datab
ase='stock_db',charset='utf8')
    # 获得 Cursor 对象
    cs = conn.cursor()
    cs.execute("select * from info;")
    stock_infos = cs.fetchall()
    cs.close()
    conn.close()

    content = re.sub(r"\{%content%\}", str(stock_infos), content)

    return content
```

```
@route("/center.html")
def center():
    with open("./templates/center.html",encoding="utf-8") as f:
        content = f.read()

    my_stock_info = "这里是从 mysql 查询出来的数据。。。"

    content = re.sub(r"\{%content%\}", my_stock_info, content)

    return content


def application(env, start_response):
    start_response('200 OK', [('Content-Type', 'text/html;charset=utf-
8')])

    file_name = env['PATH_INFO']
    # file_name = "/index.py"

    """
    if file_name == "/index.py":
        return index()
    elif file_name == "/center.py":
        return center()
    else:
        return 'Hello World! 我爱你中国....'
    """

    try:
        # func = URL_FUNC_DICT[file_name]
        # return func()
        return URL_FUNC_DICT[file_name]()
    except Exception as ret:
        return "产生了异常：%s" % str(ret)
```

执行效果如下：

20)), (76, '600781', '辅仁药业', '8.61%', '4.16%', Decimal('23.46'), Decimal('21.89'), datetime.date(2017, 5, 2)), (77, '600801', '华新水泥', '4.00%', '10.15%', Decimal('12.99'), Decimal('12.49'), datetime.date(2017, 7, 20)), (78, '600846', '同济科技', '2.06%', '17.41%', Decimal('9.39'), Decimal('9.26'), datetime.date(2017, 4, 13)), (79, '600884', '杉杉股份', '1.08%', '3.53%', Decimal('20.67'), Decimal('20.45'), datetime.date(2017, 7, 20)), (80, '600966', '博汇纸业', '2.89%', '5.54%', Decimal('6.41'), Decimal('6.28'), datetime.date(2017, 7, 19)), (81, '600971', '恒源煤电', '2.36%', '8.81%', Decimal('12.16'), Decimal('11.88'), datetime.date(2017, 7, 20)), (82, '601012', '隆基股份', '0.76%', '1.30%', Decimal('19.93'), Decimal('19.78'), datetime.date(2017, 7, 20)), (83, '601100', '恒立液压', '4.78%', '0.92%', Decimal('19.31'), Decimal('18.97'), datetime.date(2017, 7, 13)), (84, '601101', '昊华能源', '4.03%', '6.06%', Decimal('11.10'), Decimal('10.80'), datetime.date(2017, 7, 19)), (85, '601216', '君正集团', '2.16%', '2.26%', Decimal('5.20'), Decimal('5.10'), datetime.date(2017, 4, 17)), (86, '601666', '平煤股份', '2.81%', '6.14%', Decimal('6.96'), Decimal('6.77'), datetime.date(2017, 7, 20)), (87, '601668', '中国建筑', '2.39%', '1.42%', Decimal('10.70'), Decimal('10.45'), datetime.date(2017, 7, 20)), (88, '601678', '滨化股份', '0.13%', '2.47%', Decimal('7.92'), Decimal('7.91'), datetime.date(2017, 7, 20)), (89, '601918', '新集能源', '1.23%', '3.11%', Decimal('4.93'), Decimal('4.92'), datetime.date(2017, 7, 19)), (90, '603167', '渤海轮渡', '2.77%', '3.34%', Decimal('11.87'), Decimal('11.61'), datetime.date(2017, 4, 13)), (91, '603369', '今世缘', '3.34%', '2.13%', Decimal('14.24'), Decimal('13.78'), datetime.date(2017, 7, 20)), (92, '603589', '口子窖', '3.99%', '1.84%', Decimal('39.37'), Decimal('39.04'), datetime.date(2017, 6, 26)), (93, '603799', '华友钴业', '2.38%', '7.19%', Decimal('67.46'), Decimal('65.89'), datetime.date(2017, 7, 20)), (94, '603993', '洛阳钼业', '2.94%', '2.50%', Decimal('7.36'), Decimal('7.16'), datetime.date(2017, 7, 19)))

序号	股票代码	股票简称	涨跌幅	换手率	最新价(元)	前期高点	前期高点日期	添加自选

20.6 mini-web 框架-组装数据为 html 格式

上面可以看到数据格式是不对的，那我们如何让数据按照行业显示呢？请看下面代码

mini_frame.py(更新)

```
 import re
from pymysql import connect

"""
URL_FUNC_DICT = {
    "/index.html": index,
    "/center.html": center
}
"""

URL_FUNC_DICT = dict()


def route(url):
    def set_func(func):
        # URL_FUNC_DICT["/index.py"] = index
        URL_FUNC_DICT[url] = func
        def call_func(*args, **kwargs):
            return func(*args, **kwargs)
        return call_func
    return set_func


@route("/index.html")
def index():
    with open("./templates/index.html",encoding="utf-8") as f:
        content = f.read()
```

```
    # my_stock_info = "哈哈哈哈 这是你的本月名称....."
    # content = re.sub(r"\{%content%\}", my_stock_info, content)
    # 创建 Connection 连接
    conn =
connect(host='192.168.0.111',port=3306,user='root',password='123',datab
ase='stock_db',charset='utf8')
    # 获得 Cursor 对象
    cs = conn.cursor()
    cs.execute("select * from info;")
    stock_infos = cs.fetchall()
    cs.close()
    conn.close()

    tr_template = """<tr>
        <td>%s</td>
        <td>%s</td>
        <td>%s</td>
        <td>%s</td>
        <td>%s</td>
        <td>%s</td>
        <td>%s</td>
        <td>%s</td>
        <td>
            <input type="button" value="添加" id="toAdd" name="toAdd"
systemidvaule="000007">
        </td>
        </tr>
    """

    html = ""
    for line_info in stock_infos:
        html += tr_template %
(line_info[0],line_info[1],line_info[2],line_info[3],line_info[4],line_i
nfo[5],line_info[6],line_info[7])

    # content = re.sub(r"\{%content%\}", str(stock_infos), content)
    content = re.sub(r"\{%content%\}", html, content)

    return content
```

```
@route("/center.html")
def center():
    with open("./templates/center.html",encoding="utf-8") as f:
        content = f.read()

    # my_stock_info = "这里是从 mysql 查询出来的数据。。。"
    # content = re.sub(r"\{%content%\}", my_stock_info, content)
    # 创建 Connection 连接
    conn =
connect(host='192.168.0.111',port=3306,user='root',password='123',datab
ase='stock_db',charset='utf8')
    # 获得 Cursor 对象
    cs = conn.cursor()
    cs.execute("select
i.code,i.short,i.chg,i.turnover,i.price,i.highs,f.note_info from info
as i inner join focus as f on i.id=f.info_id;")
    stock_infos = cs.fetchall()
    cs.close()
    conn.close()

    tr_template = """
        <tr>
            <td>%s</td>
            <td>%s</td>
            <td>%s</td>
            <td>%s</td>
            <td>%s</td>
            <td>%s</td>
            <td>%s</td>
            <td>
                <a type="button" class="btn btn-default btn-xs"
href="/update/300268.html"> <span class="glyphicon glyphicon-star"
aria-hidden="true"></span> 修改 </a>
            </td>
            <td>
                <input type="button" value="删除" id="toDel"
name="toDel" systemidvaule="300268">
            </td>
```

```
            </tr>
        """

        html = ""
        for line_info in stock_infos:
            html += tr_template %
(line_info[0],line_info[1],line_info[2],line_info[3],line_info[4],line_i
nfo[5],line_info[6])

        # content = re.sub(r"\{%content%\}", str(stock_infos), content)
        content = re.sub(r"\{%content%\}", html, content)

        return content


def application(env, start_response):
    start_response('200 OK', [('Content-Type', 'text/html;charset=utf-
8')])

    file_name = env['PATH_INFO']
    # file_name = "/index.py"

    """
    if file_name == "/index.py":
        return index()
    elif file_name == "/center.py":
        return center()
    else:
        return 'Hello World! 我爱你中国....'
    """


    try:
        # func = URL_FUNC_DICT[file_name]
        # return func()
        return URL_FUNC_DICT[file_name]()
    except Exception as ret:
        return "产生了异常：%s" % str(ret)
```