第6天打通运维的任督二脉

一、前言

这次课程主要是解决运维人员对动态网站的模糊概念的问题，还有解决缓存到底是如何使用的疑问。

导致以上问题的原因还有，目前运维人员不了解一个动态网站是如何编写的。

自己没有接触过真东西，今天我们就来带大家来亲自体会一下动态网站的编写过程。

1. 效果图

首先来看几张我们网站的效果图，别看简陋，但是技术原理都是一样的，对于理解动态网站和缓存会有非常大的作用。

首页

服务器信息展示

获取 JSON 数据

2. 什么是 JSON 数据

> JSON(JavaScript Object Notation) 是一种轻量级的数据交换格式。 易于人阅读和编写。同时也易于机器解析和生成。

JSON建构于两种结构：

- “名称/值” 成对的集合（A collection of name/value pairs）。比如

```
'{"name": "shark"}'
```

不同的语言中，它被理解为对象（object），纪录（record），结构（struct），字典（dictionary），哈希表（hash table），有键列表（keyed list），或者关联数组 （associative array）。

- 值的有序列表（An ordered list of values）。 在大部分语言中，它被理解为数组（array）。

比如

```
'[1, 2, "hello"]'
```

这些都是常见的数据结构。事实上大部分现代计算机语言都以某种形式支持它们。这使得同样基于这些结构的不同编程语言之间，可以交互这样一种数据格式。

比如 Python python 中的内置模块 `json` 可以实现在 JSON 数据和Python 对象直接的互相转换

- Python 对象转换为 JSON 叫序列化

```
In [8]: import json

In [9]: d = {"host_name": "qfedu.com"}

In [10]: json.dumps(d)
Out[10]: '{"host_name": "qfedu.com"}'

In [11]:
```

> 注意：JSON 数据中的字符串必须使用 双引号引起来，最外层必须是单引号

- JSON 数据转换为 Python对象 叫反序列化

```
In [12]: json.loads('[1, 2, "hello"]')
Out[12]: [1, 2, 'hello']

In [13]: li = json.loads('[1, 2, "hello"]')

In [14]: type(li)
Out[14]: list

```

- 混合格式

在生产中大部分 JSON 数据都会是下面的混合方式

```
In [15]: server_list =[
    {"host_name": "qfedu.com", "ip": "192.168.1.100"},
    {"host_name": "sharkyun.com", "ip": "1.1.1.1"}
  ]

In [16]: print(json.dumps(server_list,indent=4))
[
    {
        "host_name": "qfedu.com",
        "ip": "192.168.1.100"
    },
    {
        "host_name": "sharkyun.com",
        "ip": "1.1.1.1"
    }
]

In [19]:
```

> `indent=4` 的意思是，不同层级，缩进 4 个空格

二、 网站编写

1. uwsgi 模块实现基本的动态网站首页

uwsgi 可以帮助我们实现一个简单的网站。在 python 中它是一个第三方模块。

1. 1 安装

```
$ pip3 install uwsgi
```

1.2 小试牛刀

我们可以参考官方文档的实例来进行初步的了解

第一步编写一个程序启动文件，并把如下代码添加进行。

`wsgi.py`

```
def application(env, start_response):
    start_response('200 OK', [('Content-Type','text/html')])
    return [b"Hello World"]
```

运行网站

```
uwsgi --http :8081 --wsgi-file wsgi.py
```

之后可以使用浏览器访问

其实，也可以通过配置文件

```
[uwsgi]
http = 127.0.0.1:8081
chdir = /home/foobar/myproject/
wsgi-file = wsgi.py
processes = 4
threads = 2
```

> `processes` 开启 4 个进程 `threads` 每个进程开启 2 个线程

2. url 实现返回不同的网页

函数 `def application(env, start_response)`

- `env` 是一个封装好的字典，里面有一个 key `PATH_INFO` ，这个值就是 url 了

我们可以通过 url 的不同来判断

`views.py`

```
def index():
    file = "/root/python_code/day06/index.html"
    with open(file, mode='rb') as f:
        content = f.read()
        return content
```

`wsgi.py`

```python
import views

def application(env, start_response):
    start_response('200 OK', [('Content-Type','text/html')])
    if env['PATH_INFO'] == '/':
        return [views.index()]
    elif env['PATH_INFO'] == '/server_list':
        return ["server list".encode()]
```

3. 功能细化

> 动态网站的本质是根据请求，获取到数据，之后把数据替换到含有 html 标记语言的普通文本里，最后把含有数据的页面文件返回给浏览器， 浏览器根据标记语言的规范展示数据。

获取数据的方式一般都是从数据库中获取，这个数据库可以关系型数据库，比如 MySQL, 也可以是 非关系型数据库，比如 Redis。

下面我们就通过实现不同的功能，来展示出这都详细的实现过程。

3.1 返回首页

3.1.1 首页的后端添加返回当前日期时间的功能

```python
def handler_index():
    """
    以二进制方式返回首页文件的内容，并替换文件中的变量为当前时间
    """
    dt = time.strftime(r"%Y%-m-%d %H:%M:%S")
    with open('templates/index.html', 'r', encoding='utf-8') as f:
        content = f.read()
        content = content.replace("{{now_dt}}", dt)
    return bytes(content, encoding='utf-8')

```

wsgi.py

```python
import views

headers = {
    'html': ('Content-Type', 'text/html;charset=utf-8'),
    'json': ('Content-Type', 'application/json;charset=utf-8'),
    'css': ('Content-Type', 'text/css'),
    'jpg': ('Content-Type', 'application/x-jpg'),
    'png': ('Content-Type', 'image/png'),
}

def application(env, start_response):
    if env['PATH_INFO'] == '/':
        start_response('200 OK', [('Content-Type','text/html')])
        return [views.handler_index()]
```

```
    elif env['PATH_INFO'] == '/bootstrap/css/bootstrap.css':
        start_response('200 OK', [('Content-Type', 'text/css')])
        return [views.handler_css()]
```

3.1.2 首页的前端实现

index.html

```
<!DOCTYPE html>
<html lang="zh-CN">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="/bootstrap/css/bootstrap.css">
    <title>正经海参搞 IT</title>
</head>

<body>
    <div class="container-fluid">
        <div class="row">

            <h1 class="col-4 offset-4" >
                欢迎光临红浪漫
            </h1>
        </div>

        <div class="row">
```

```
            <div class="col-2 offset-4">这里是正经海参搞 IT</div>
            <div class="col-3" style="background-color: aqua;">当前时间：{{now_dt}}</div>
</body>

</html>

```

3.2 准备数据库

3.2.1 封装mysql 连接工具

```
import pymysql

def mysql_conn():
    """
    创建连接
    返回连接对象，游标对象
    """
    conn = pymysql.connect(host="192.168.1.37",
                           user="dbuser",
                           passwd="QFedu123!",
                           db="shark_db",
                           charset="utf8")
    # 获取游标对象
    cursor = conn.cursor(
      cursor=pymysql.cursors.DictCursor)
    return conn, cursor


def mysql_query(cursor, query_sql):
    cursor.execute(query_sql)
```

```
21
22      return cursor.fetchall()
23
24  def mysql_close(cursor, conn):
25      # 关闭游标对象
26      cursor.close()
27
28      # 关闭连接对象
29      conn.close()
30
31  if __name__ == "__main__":
32      query_sql = "select * from base_info;"
33      conn, cursor = mysql_conn()
34      print(mysql_query(cursor, query_sql))
35      mysql_close(cursor, conn)
36
37
38
39  ###################### 示例数据 ##################
40
41  """
42  dic =  {
43      "base_info": {
44          "host_name": "nginx_server",
45          "kernel": "3.10.0-957.21.3.el7.x86_64",
46          "os": "CentOS Linux release 7.6.1810
    (Core)",
47          'manufacturer': 'Alibaba Cloud',
48          'pod_name': 'Alibaba Cloud ECS',
49          'sn': '0f7e3d86-7742-4612-9f93-
    e3a9e4754199',
50          'cpu_name': 'Intel(R) Xeon(R) Platinum 8163
    CPU @ 2.50GHz',
51          'cpu_num': 2,
52          'cpu_cores_each': 4
```

```
    },
    "mem": [{
            'capacity': '8192 MB',
            'slot': 'DIMM_A3',
            'model': 'DDR3',
            'speed': '1333 MT/s',
            'manufacturer': '00CE04B380CE',
            'sn': '8362A2F8'
        },
        {
            'capacity': 'No Module Installed',
            'slot': 'DIMM_A4',
            'model': 'DDR3',
            'speed': 'Unknown',
            'manufacturer': '',
            'sn': ''
        }
    ]
}"""

```


3.2.2 封装 Redis 工具

```
import redis
rs = redis.StrictRedis(
    host="127.0.0.1",
    password="foo",
    decode_responses=True,
    port=6379,
    db=0
)

if __name__ == "__main__":
    print(rs.keys())

```

3.3 返回服务器列表数据

3.3.1 前端实现

`index.html`

添加如下代码：

```
        <div class="row">
            <a class="col-4 offset-2 btn bg-success" href="/server_list">我要服务器列表</a>
            <a class="col-4 btn bg-warning" href="/api/server">我要服务器 JSON 数据</a>
    </div>
```

`server_list.html`

```
<!DOCTYPE html>
<html lang="zh-CN">
```

```
3
4   <head>
5       <meta charset="UTF-8">
6       <meta name="viewport" content="width=device-
    width, initial-scale=1.0">
7       <link rel="stylesheet"
    href="/bootstrap/css/bootstrap.css">
8       <title>西瓜甜</title>
9   </head>
10
11  <body>
12      <div class="container-fluid">
13          <div class="row">
14                  <h1 class="col-4 offset-4">服务器基础
    信息</h1>
15          </div>
16          <div row>
17              <button class="btn btn-warning col-md2
    offset-2">{{cache}}</button>
18              <a href="/cache" class="btn btn-danger
    col-md-2">验证使用缓存的效果</a>
19
20              <button class="btn btn-warning col-md2">
    {{dt}}</button>
21              <a href="/" class="btn btn-success col-
    md-2">返回首页</a>
22          </div>
23
24          <div class="row">
25              <div class="col-8 offset-2">
26
27
28                  <table class="table table-condensed
    table-hover table-bordered">
29                      <thead class="table-dark">
```

```
                            <tr>
                                <th>id</th>
                                <th>主机名</th>
                                <th>操作系统</th>
                            </tr>
                        </thead>
                        <tbody>
                            {{table_tr}}
                        </tbody>
                    </table>
                </div>
            </div>
        </div>
    </body>

    </html>

```


3.3.2 views.py

```
def server_data():
    conn, cursor = mysql_conn()
    sql = 'select id, host_name, os from base_info;'
    ret = mysql_query(cursor, sql)
    mysql_close(cursor, conn)
    return ret


def server_list():
    tr_tpl = '''
            <tr>
                <td>{id}</td>
                <td>{host_name}</td>
```

```
                <td>{os}</td>
            </tr>
    '''

    ret = server_data()

    tr = ''
    for row in ret:
        tr += tr_tpl.format(**row)

    with open('templates/server_list.html', 'r', encoding='utf-8') as f:
        content = f.read()
        content = content.replace('{{table_tr}}', tab_tr)
        content = content.replace('{{cache}}', '未使用缓存')
        content = content.replace('{{dt}}', "不用计时")
    return bytes(content, encoding='utf-8')
```


3.3 利用缓存

`wsgi.py` 文件中添加如下内容：

```
    elif env['PATH_INFO'] == "/cache":
        start_response('200 OK', [('Content-Type','text/html')])
        return [views.mysql_or_cache()]
```

`views.py` 文件添加如下内容：

```
def mysql_or_cache():
    st = time.time()
    rs = cache.rs
    ex = rs.ttl("server_info")
    if ex >= 1:
        data = rs.get("server_info")
        data = json.loads(data)
    else:
        data = server_data()

        # 添加到缓存
        rs.set("server_info", json.dumps(data), ex=10)
    endt = time.time()
    use_dt = endt - st

    tr_tpl = """
            <tr>
                <td>{id}</td>
                <td>{host_name}</td>
                <td>{os}</td>
            </tr>"""
    file = 'templates/server_list.html'
    tr = ''
    for row in data:
        tr += tr_tpl.format(**row)

    with open(file, 'r') as f:
        content = f.read()
        content = content.replace("{{table_tr}}", tr)
```

```
30          content = content.replace("{{cache}}", "使用缓存")
31          content = content.replace("{{dt}}", str(use_dt))
32          return content.encode()
```

3.4 返回 JSON 数据

在 `wsgi.py` 文件中添加如下代码：

```
1     elif env['PATH_INFO'] == '/api/server':
2         start_response('200 OK', [('Content-Type', 'application/json')])
3         data = views.server_data()
4         return [json.dumps(data, indent=4).encode()]
```