配置Web服务器，编写简单页面，分析交互过程

一. 实验要求

1. 搭建Web服务器（自由选择系统），并制作简单的Web页面，包含简单文本信息（至少包含专业、学号、姓名）和自己的LOGO。
2. 通过浏览器获取自己编写的Web页面，使用Wireshark捕获浏览器与Web服务器的交互过程，并进行简单的分析说明。
3. 提交实验报告。

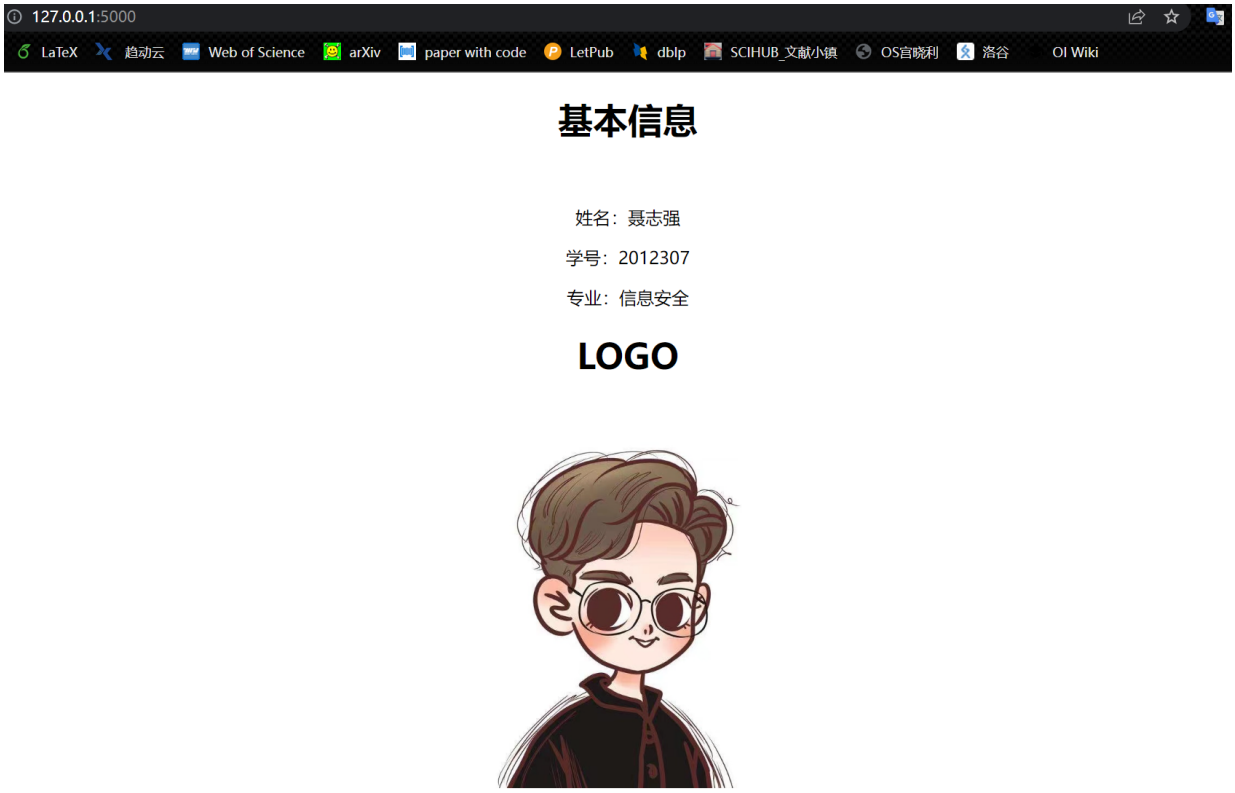
二. Web服务器搭建

1. 代码

```python
from flask import Flask, render_template

# 创建Flask类实例app
app = Flask(__name__)

# 通过装饰器设置函数URLs访问路径
@app.route('/')

# 定义设置网站首页的处理函数
def hello_world():
    return render_template("web.html")

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

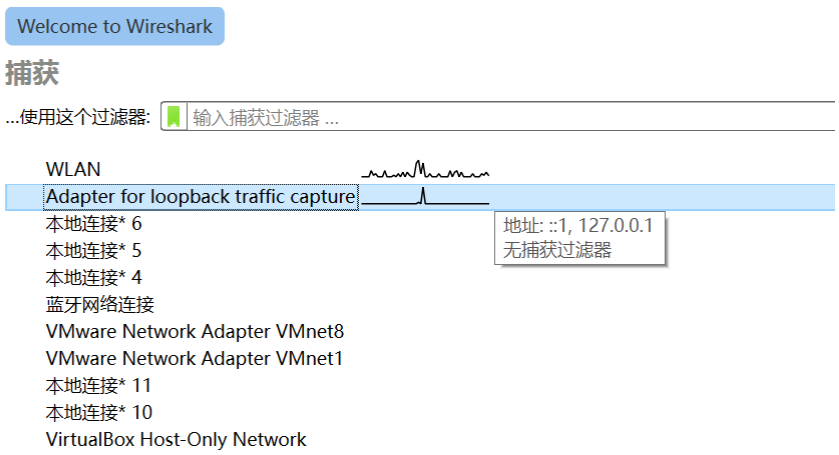
```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>配置Web服务器</title>
</head>
<body>
    <h1 style="text-align: center">基本信息</h1>>
    <p style="text-align: center">姓名：聂志强</p>
    <p style="text-align: center">学号：2012307</p>
    <p style="text-align: center">专业：信息安全</p>
    <h1 style="text-align: center"> LOGO </h1>>
    <p style="text-align : center"><img src="../static/computer-network.jpg" style="zoom:50%;"></p>
</body>
</html>
```

2. **Windows系统上使用Pycharm搭建web服务器，效果如下图所示**



三. Wireshark捕获交互过程

1. **通过本机 127.0.0.1 访问页面，因此在 Wireshark 中选择回环网卡进行监听**



2. **TCP连接状态标识**

   - **SYN** 表示建立连接
   - **FIN** 表示关闭连接
   - **ACK** 表示响应
   - **PSH** 表示有 DATA数据传输

- **RST** 表示连接重置

3. 三次握手

三次握手是建立一个TCP连接时，需要客户端和服务器总共发送3个包。三次握手的目的是连接服务器指定端口，建立连接，并同步连接双方的序列号和确认号，交换窗口大小信息

```
127.0.0.1          TCP        56 51737 → 5000 [SYN] Seq=0 Win=65535 Len=0 MSS=65495 WS=256 SACK_PERM
127.0.0.1          TCP        56 5000 → 51737 [SYN, ACK] Seq=0 Ack=1 Win=8192 Len=0 MSS=65495 WS=256 SACK_PERM
127.0.0.1          TCP        44 51737 → 5000 [ACK] Seq=1 Ack=1 Win=2619648 Len=0
```

- **第一次握手**

  客户端将 TCP 报文标志位 SYN 置为 1，随机产生一个序号值 seq=x（如图 x=0），保存在 TCP 首部的序列号字段里，指明客户端打算连接的服务器的端口，并将该数据包发送给服务器端，发送完毕后，客户端进入 SYN_SENT （同步已发送状态），等待服务器端确认。TCP规定，SYN报文段（SYN=1的报文段）不能携带数据，但需要消耗掉一个序号。

- **第二次握手**

  服务器收到客户端的SYN报文段，如图中同意连接，则发出确认报文，确认报文中ACK=1，SYN=1，确认号ACKnum=x+1（客户端的seq+1），同时，为自己初始化一个序列号seq=y，服务器端将上述所有信息放到一个报文段（即SYN+ACK报文段）中，一并发送给客户端，此时，TCP服务器进程进入SYN-RCVD（同步收到）状态。这个报文也不能携带数据，但是同样要消耗一个序号。

- **第三次握手**

  客户单收到服务器的SYN+ACK报文段，再次发送确认包（ACK），SYN标志位0，ACK标志位为1，确认号ACKnum = y+1（服务器端seq+1），这个报文段发送完毕以后，客户端和服务器都进入ESTABLISHED（已建立连接）状态，完成TCP三次握手。

4. 四次挥手

所有的请求-连接完成后，便可以通过四次挥手断开连接，这里既可以是浏览器首先发送断开连接请求，也可以是服务器发送请求，取决于应用层实现，TCP 只关心是否连接。可以看到， 我们这次是服务器首先请求断开连接，我们就一个断开连接分析其过程：

```
127.0.0.1          TCP        44 5000 → 51738 [FIN, ACK] Seq=665 Ack=749 Win=262400 Len=0
127.0.0.1          TCP        44 51738 → 5000 [ACK] Seq=749 Ack=666 Win=2619136 Len=0
127.0.0.1          TCP        44 51738 → 5000 [FIN, ACK] Seq=749 Ack=666 Win=2619136 Len=0
127.0.0.1          TCP        44 5000 → 51738 [ACK] Seq=666 Ack=750 Win=262400 Len=0
```

- **第一次挥手**

  服务器端设置seq=x, 向客户端发送一个FIN报文段，此时服务器端进入FIN_WAIT_1状态，这表示服务器端没有数据要发送给客户端了。

- **第二次挥手**

  客户端收到服务器端发送的FIN报文段，向服务器端回一个ACK报文段，ACKnum=x+1，服务器端进入FIN_WAIT_2状态，客户端告诉服务器端，我“同意”你的关闭请求。

- **第三次挥手**

  客户端向服务器端发送FIN报文段，请求关闭连接，同时客户端进入LAST_ACK状态。

- **第四次挥手**

  服务器端收到客户端发送的FIN报文段，向客户端发送ACK报文段，然后服务器端进入TIME_WAIT状态，客户端收到服务器端的报文段以后，就关闭连接；此时，服务器端等待2MSL后依然没有收到回复，则证明已正常关闭，那么，服务器端也可以关闭连接了，进入CLOSED状态。

5. **获取网页：（GET 请求报文，HTTP 响应报文）[将Wireshark过滤条件设为 http]**

建立连接之后，通过两次 HTTP 的“请求-响应”，分别获取一个网页文本和一个图片

	Destination	Protocol	Length	Info
0.0.1	127.0.0.1	HTTP	792	GET / HTTP/1.1
0.0.1	127.0.0.1	HTTP	553	HTTP/1.0 200 OK (text/html)
0.0.1	127.0.0.1	HTTP	734	GET /static/computer-network.jpg HTTP/1.1
0.0.1	127.0.0.1	HTTP	451	HTTP/1.0 200 OK (JPEG JFIF image)

- 客户端通过 HTTP 协议的 GET 请求网站，使用 HTTP1.1 协议
- 服务器返回 ACK 表示已收到上面的请求，然后通过 HTTP 协议发送 OK，并将文本文件 (html) 回送给浏览器
- 客户端返回 ACK 表示已收到上面的回送消息，并调用 JQuary 等在本地进行页面渲染，发现页面中包含一个图片，于是再次通过 HTTP 协议的 GET 请求图片；
- 服务器返回 ACK 表示已收到上面的请求，通过 HTTP 协议发送 OK，并将图片 (jpg)回送给客户端，客户端返回 ACK 表示已收到上面的回送消息，并调用 JQuary 等对图片进行本地渲染，在浏览器上显示完整的页面，整个“请求-响应”阶段结束。

6. **Get请求报文分析**

```
Hypertext Transfer Protocol
  GET / HTTP/1.1\r\n
  Host: 127.0.0.1:5000\r\n
  Connection: keep-alive\r\n
  sec-ch-ua: "Chromium";v="106", "Google Chrome";v="106", "Not;A=Brand";v="99"\r\n
  sec-ch-ua-mobile: ?0\r\n
  sec-ch-ua-platform: "Windows"\r\n
  Upgrade-Insecure-Requests: 1\r\n
  User-Agent: Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/106.0.0.0 Safari/537.36\r\n
  Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9\r\n
  Sec-Fetch-Site: none\r\n
  Sec-Fetch-Mode: navigate\r\n
  Sec-Fetch-User: ?1\r\n
  Sec-Fetch-Dest: document\r\n
  Accept-Encoding: gzip, deflate, br\r\n
  Accept-Language: zh-CN,zh;q=0.9,en-US;q=0.8,en;q=0.7\r\n
  Cookie: SL_G_WPT_TO=zh; SL_GWPT_Show_Hide_tmp=1; SL_wptGlobTipTmp=1\r\n
  \r\n
  [Full request URI: http://127.0.0.1:5000/]
  [HTTP request 1/1]
  [Response in frame: 35]
```

- Get请求报文由请求行、首部行、空行和实体主体 4 个部分组成
- 请求行包含三个字段：方法字段（GET）、URL字段和HTTP版本字段
- Host：127.0.0.1:5000指明了对象所在主机，即连接的目标主机
- Connection：对于 HTTP 连接的处理，keep-alive 表示保持连接，如果是在响应报文中发送页面完毕就会关闭连接，状态变为 close。如果Connection的值为close，则浏览器告诉服务器不希望麻烦地使用持续连接，他要求服务器在发送完被请求地对象后就关闭这条连接

- User-Agent：指明用户代理，向服务器发送请求的浏览器类型，服务器可以有效地为不同类型的用户代理实际发送相同对象的不同版本
- Accept-Languae：表示用户想得到该对象的语言版本
- Cookie有关信息
- 空行即为回车换行
- 使用GET方法时实体体为空，而使用POST方法时才使用该实体体。当用户提交表单时，HTTP客户通常使用POST方法，用户可以向服务器请求一个Web页面，但Web页面特定的内容依赖于用户在表单字段中输入的内容，实体体中包含的就是用户在表单字段中的输入值

7. HTTP响应报文分析

```
Hypertext Transfer Protocol
  HTTP/1.0 200 OK\r\n
  Content-Type: text/html; charset=utf-8\r\n
  Content-Length: 509\r\n
  Server: Werkzeug/2.0.3 Python/3.9.12\r\n
  Date: Thu, 27 Oct 2022 17:37:18 GMT\r\n
  \r\n
  [HTTP response 1/1]
  [Time since request: 0.011771000 seconds]
  [Request in frame: 29]
  [Request URI: http://127.0.0.1:5000/]
  File Data: 509 bytes
Line-based text data: text/html (15 lines)
  <!DOCTYPE html>\n
  <html lang="en">\n
  <head>\n
      <meta charset="UTF-8">\n
      <title>配置Web服务器</title>\n
  </head>\n
  <body>\n
      <h1 style="text-align: center">基本信息</h1>>\n
      <p style="text-align: center">姓名：聂志强</p>\n
      <p style="text-align: center">学号：2012307</p>\n
      <p style="text-align: center">专业：信息安全</p>\n
      <h1 style="text-align: center"> LOGO </h1>>\n
      <p style="text-align : center"><img src="../static/computer-network.jpg" style="zoom:50%;"></p>\n
  </body>\n
  </html>
```

- HTTP 响应报文由状态行、首部行、空行和实体体 4 个部分组成
- 状态行由 HTTP 协议版本字段、状态码和相应状态信息（状态码的描述文本） 3 个部分组成
- Data：指示服务器产生并发送该响应报文 的日期和时间，这个时间不是对象创建或最终修改的时间，而是服务器从它的文件系统中检索到该对象，插入到响应报文，并发送该响应报文的时间。
- Sever：指示该报文是由Werkzeug服务器产生的
- Cotent-Length：被发送对象中的字节数
- Cotent-Type：指示了实体体中的对象是HTML文本
- 每行均有一个回车和换行符
- 实体体是报文的主要部分，包含了所请求的对象本身

- 状态码及其相应短语指示了请求结果，状态码以及描述文本有常见的如下几种：

状态码	解释	含义
200	OK	请求成功
301	Moved Permanently	重定向，请求的对象被移走，新的位置在 Location 中给出
304	Not Modified	对象未修改，无需再次传输请求的内容
400	Bad Requst	服务器不能解释请求报文
404	Not Found	服务器找不到请求的文档
505	HTTP Version Not Supported	服务器不支持相应的 HTTP 版本

四. 思考

1. **为什么需要三次握手？两次不行吗？**

   为了防止第一次握手时的请求报文段，在发送到某个地方由于链路拥堵而超时抵达服务器，这时服务器还会响应，向客户端发送确认报文段，而客户端已经超时，所以不会再等下去。而只有客户端及时收到服务器发送的确认报文段，而且及时响应服务器。

2. **为什么TIME_WAIT状态需要经过2MSL（最大报文段生存时间）才能返回到CLOSE状态？**

   - 按理说四个报文发完，就直接进入CLOSE状态，但是，我们顾虑的是在传输过程中超时或丢失(不可靠)ACK，所以TIME_WAIT状态就是用来重发可能丢失的ACK报文。
   - 还有就是，防止类似于“三次握手”中提到的“已经失效的连接请求报文段”出现在本连接中。客户单发送完最后一个确认报文后，在这个2MSL时间中，就可以使本连接持续的时间内锁产生的所有报文段都从网络中消失。这样新的连接中就不会出现旧连接的请求报文。

3. **什么连接的时候是三次握手，关闭的时候却是四次握手？**

   因为当Server端收到Client端的SYN连接请求报文后，可以直接发送SYN+ACK报文。其中ACK报文是用来应答的，SYN报文是用来同步的。但是关闭连接时，当Server端收到FIN报文是，很可能并不会立即关闭SOCKET,所以只能先回复ACK报文，告诉Client端，“你发的FIN报文我收到了”，只有等到我Server端所有的报文发送完了，我才能发送FIN报文，因此不能一起发送。故需要四步握手。