MILLENNIUM

>_ CS305 Final Project

Artanisax IceLocke DeerInForestOvO

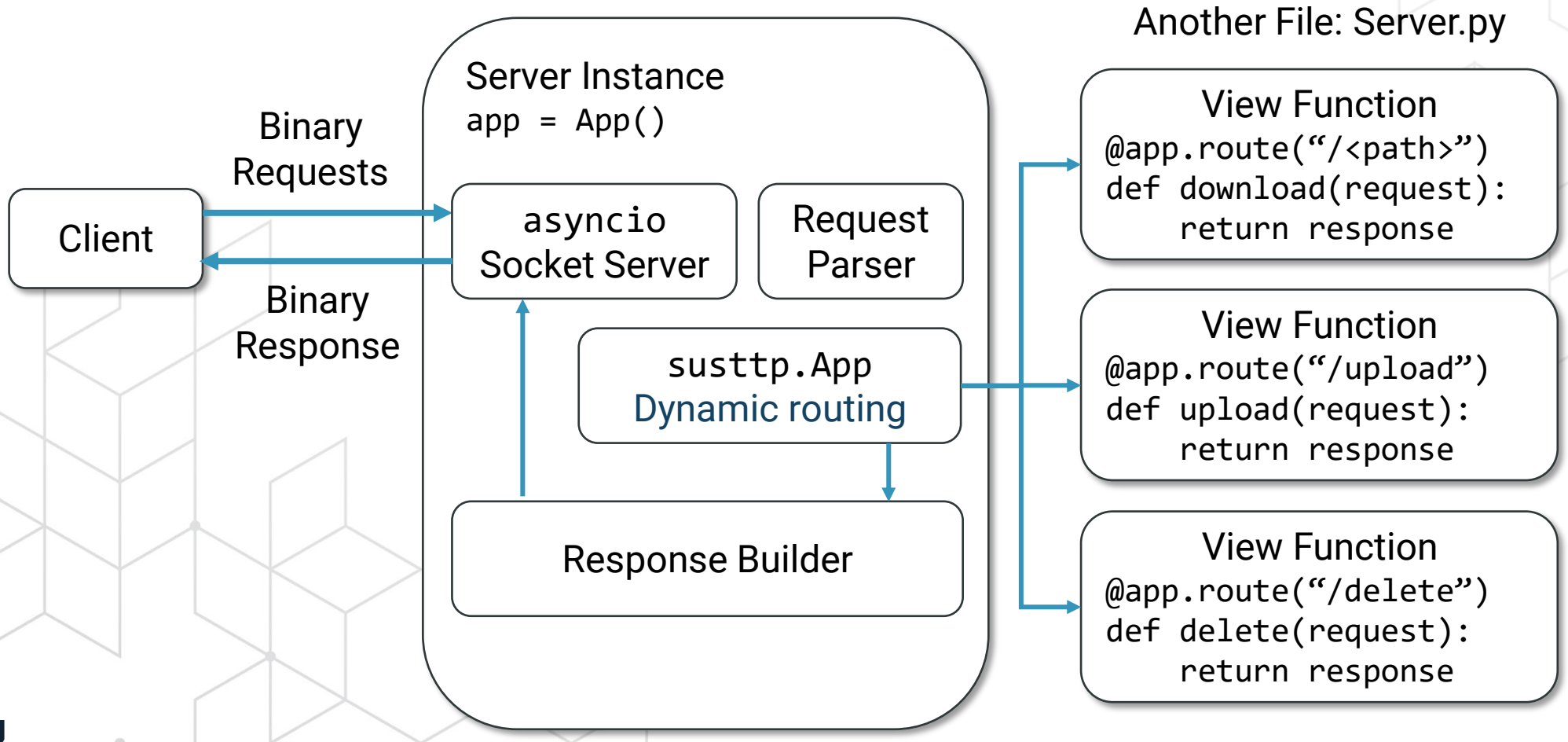
MILLENNIUM

>_ Framework

Structure Design (Basic Part)

Chapter 0x00

Overview: Flask-like framework



MILLENNIUM

Chapter 0x00

Request

- We encapsulate the request into a class `susttp.Request`.

```
self.method = method
self.path = path
self.version = version
self.headers = headers
self.request_param = None
self.path_param = None
self.cookies = None
self.anchor = None
self.body = None
```

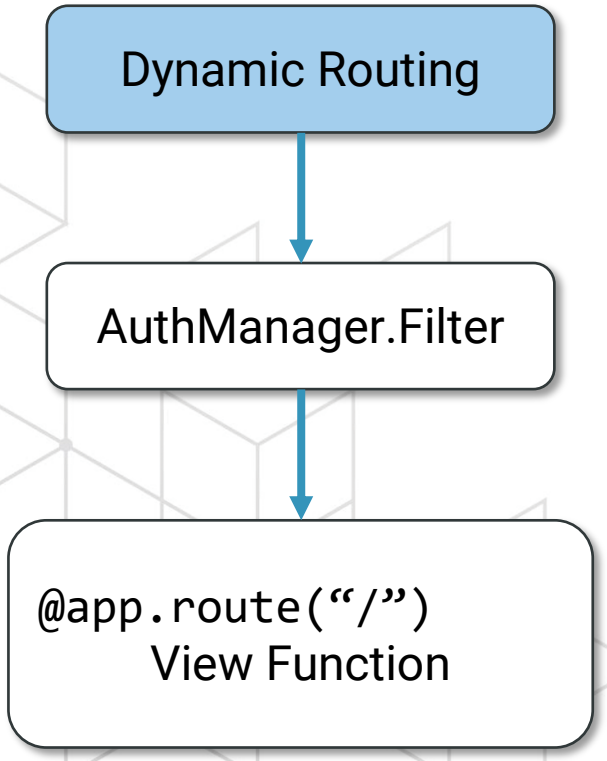
- Then our view functions can feasibly access the request context.

- ~~(However we didn't implement a cooler context manager)~~

MILLENNIUM

Chapter 0x00

Dynamic Routing



- Dynamic routing is very convenient for developing web apps.
- So, we implement a dynamic routing algorithm with a simple Flask-like routing syntax like `@app.route("/<name>")` by `re`

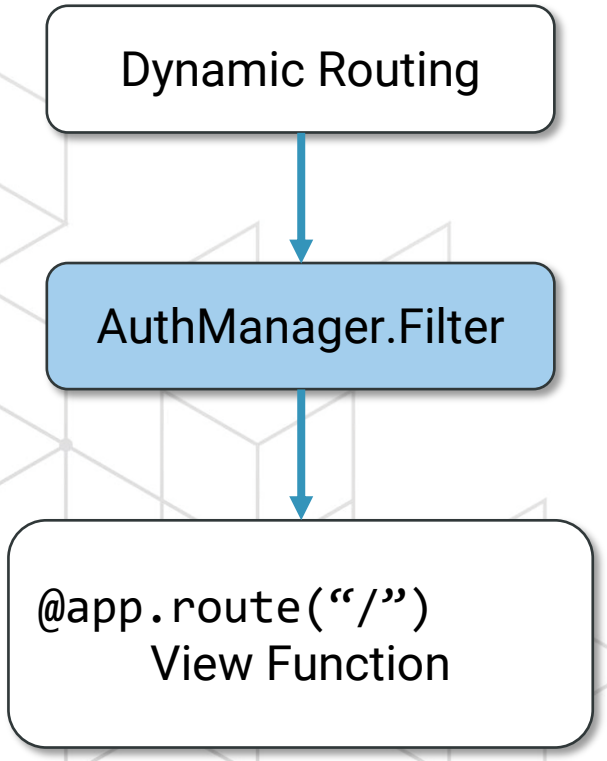
```
@app.route("/<name>")
def get_name(request):
    return Response(body=request.name)
```

- In the core of server, we will parse the request and route to view functions which complies the pre-defined path rules.

MILLENNIUM

Chapter 0x00

Authorization

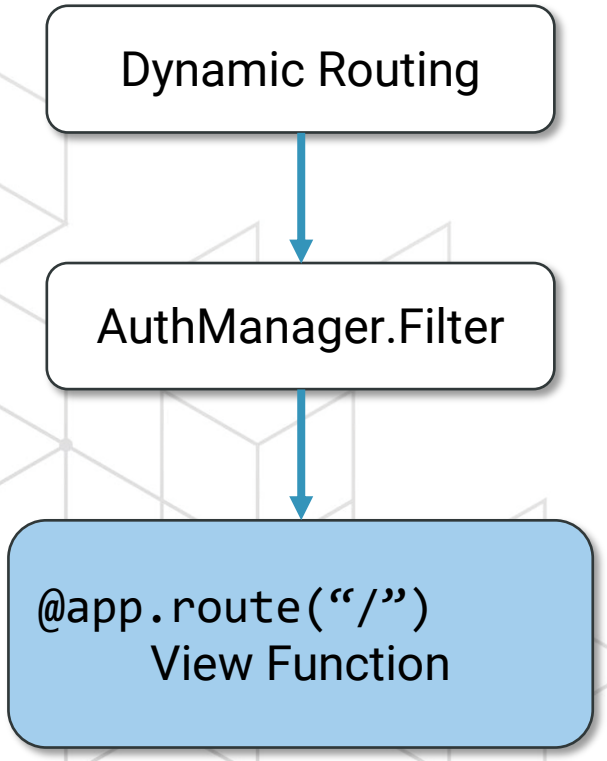


- We integrate authorization functions in a class `AuthManager`.
- Actually it's like a combination of `SecurityFilter` and `AuthenticationManager` in Spring Boot.
- It filters unauthorized requests for view functions that requires authorization, and can get authentication status of a request through it.

MILLENNIUM

Chapter 0x00

View Functions



🤓 👆 Well, actually it's the same concept in Flask.

- We will handle the encapsulate request in the view functions, and return encapsulate response to the server core.
- Then the response will be convert into byte stream and sent back to the client in the core.

MILLENNIUM

Chapter 0x00

Response

- We also encapsulate the response into a class `susttp.Response`.
- Setting status code, response body, chunked / breakpoint response information, etc. is permitted before the real response was built.

- Very easy to use in view functions.

MILLENNIUM

MILLENNIUM

>_ Test Results

Basic Part

Chapter 0x01

Authorization

```python
headers={"Authorization": "Basic Y2xpZW50MToxMjM="}
q=requests.head('http://localhost:8080/',headers=headers)
print(q)
```

在 2024.01.02 09:53:01 于 65ms内执行

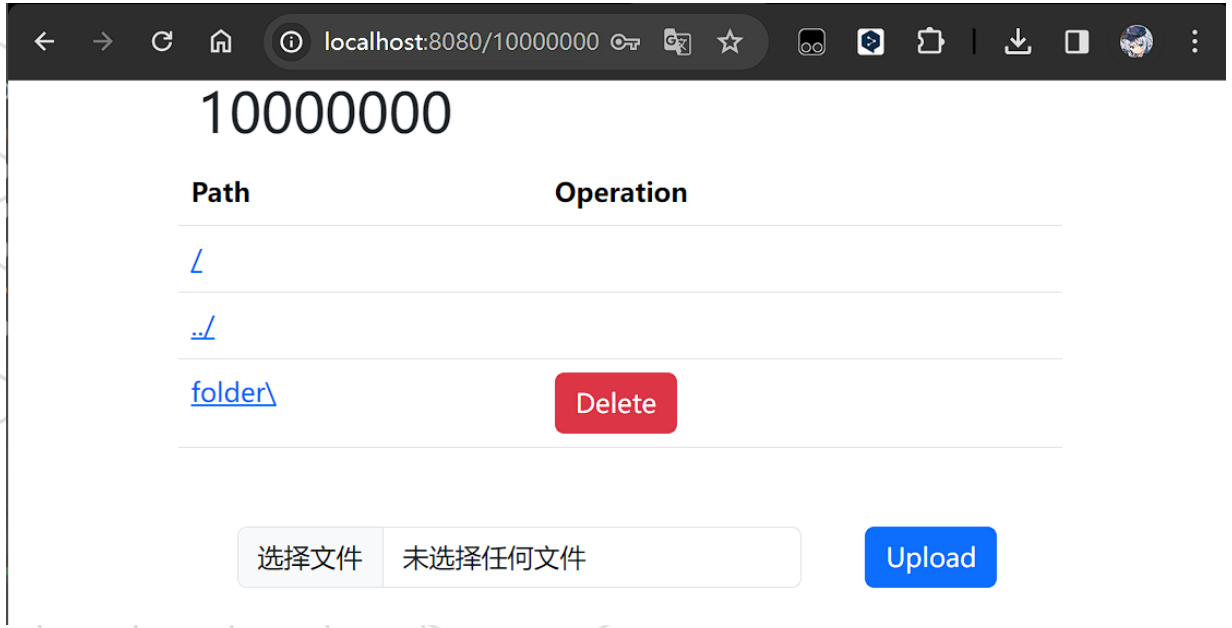
```
<Response [200]>
```

```
▶ Frame 19364: 282 bytes on wire (2256 bits), 282 bytes captured (2256 bits) on interface \Device\NPF_Loopback, id 0
▶ Null/Loopback
▶ Internet Protocol Version 6, Src: ::1, Dst: ::1
▶ Transmission Control Protocol, Src Port: 8080, Dst Port: 60215, Seq: 1, Ack: 186, Len: 218
▼ Hypertext Transfer Protocol
  ▼ HTTP/1.1 200 OK\r\n
    ▶ [Expert Info (Chat/Sequence): HTTP/1.1 200 OK\r\n]
      Response Version: HTTP/1.1
      Status Code: 200
      [Status Code Description: OK]
      Response Phrase: OK
    Server: SUSTTP Server\r\n
    Date: Tue, 02 Jan 01:53:01 GMT\r\n
    Content-Type: text/html\r\n
  ▶ Content-Length: 4053\r\n
    Set-Cookie: session-id=3b1632e618a987431707e0039bf80eb8db688fa9d59a42eea1b9a011406fb7fa; Path=/\r\n
    \r\n
    [HTTP response 1/1]
    [Time since request: 0.007038000 seconds]
    [Request in frame: 19362]
    [Request URI: http://localhost:8080/]
```

MILLENNIUM

Chapter 0x01

Get Request




```
Test get request

headers={"Authorization": "Basic Y2xpZW50MToxMjM="}
q=requests.get('http://localhost:8080/',headers=headers)
print(q)
在 2024.01.02 09:56:39 于 58ms内执行

<Response [200]>
```

MILLENNIUM

Chapter 0x01

Post Request

Test post request

```
headers={"Authorization": "Basic Y2xpZW50MToxMjM="}
q=requests.post('http://localhost:8080/',headers=headers)
print(q)
```

在 2024.01.02 10:25:44 于 83ms内执行

```
<Response [405]>
```

```
▶ Frame 60834: 64 bytes on wire (512 bits), 64 bytes captured (512 bits) on interface \Device\NPF_Loopback, id 0
▶ Null/Loopback
▶ Internet Protocol Version 6, Src: ::1, Dst: ::1
▶ Transmission Control Protocol, Src Port: 8080, Dst Port: 52992, Seq: 229, Ack: 205, Len: 0
▶ [2 Reassembled TCP Segments (228 bytes): #60832(228), #60834(0)]
▼ Hypertext Transfer Protocol
  ▼ HTTP/1.1 405 Method Not Allowed\r\n
    ▶ [Expert Info (Chat/Sequence): HTTP/1.1 405 Method Not Allowed\r\n]
      Response Version: HTTP/1.1
      Status Code: 405
      [Status Code Description: Method Not Allowed]
      Response Phrase: Method Not Allowed
    Server: SUSTTP Server\r\n
    Date: Tue, 02 Jan 02:25:44 GMT\r\n
    Content-Type: text/plain; charset=utf-8\r\n
    Set-Cookie: session-id=aa24b128e5c210575fb733791b999bfc960449ba84470d70a4b9b8d2d23bb8c9; Path=/\r\n
    \r\n
```

MILLENNIUM

Chapter 0x01

Integrated Tests（1）

```
#1.1
headers={"Authorization": "Basic Y2xpZW50MToxMjM="}
session = requests.Session()
session.headers.update({'Connection': 'keep-alive'})

response1 = session.get('http://127.0.0.1:8080',headers=headers)
response2 = session.get('http://127.0.0.1:8080',headers=headers)

print(response1)
print(response2)
```

在 2023.11.20 20:25:26 于 28ms内执行

```
<Response [200]>
<Response [200]>
```

```
#1.2
#1.1
headers={"Authorization": "Basic Y2xpZW50MToxMjM="}
session1 = requests.Session()
session1.headers.update({'Connection': 'keep-alive'})
session2 = requests.Session()
session2.headers.update({'Connection': 'keep-alive'})

response1 = session1.get('http://127.0.0.1:8080',headers=headers)
response2 = session2.get('http://127.0.0.1:8080',headers=headers)


print(response1)
print(response2)
```

在 2024.01.02 10:37:59 于 69ms内执行

```
<Response [200]>
<Response [200]>
```

MILLENNIUM

Chapter 0x01

Integrated Tests (2)

```python
#2
url1='http://127.0.0.1:8080/?SUSTech-HTTP=1'

headers={"Authorization": "Basic Y2xpZW50MToxMjM="}
r=requests.get(url=url1, headers=headers)
print(r)
print(r.content.decode())
```

在 2024.01.02 10:43:31 于 150ms内执行

```
<Response [200]>
["a.py", "a.txt", "client1/"]
```

```python
#2
url2='http://127.0.0.1:8080/?SUSTech-HTTP=0'

headers={"Authorization": "Basic Y2xpZW50MToxMjM="}
r=requests.get(url=url2, headers=headers)
print(r.content.decode())
```

在 2024.01.02 10:40:53 于 75ms内执行

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>SUSTTP Demo</title>
```

```python
#2
headers={"Authorization": "Basic Y2xpZW50MToxMjM="}
r=requests.get(url='http://127.0.0.1:8080/a.txt', headers=headers)
print(r.content.decode())
```

在 2024.01.02 10:44:13 于 64ms内执行

```
sadfsdfasdf
```

```python
#3.1
files = {"firstFile": open('tmp/a.txt', "rb")}


data={}
headers={"Authorization": "Basic Y2xpZW50MToxMjM="}
r=requests.post(url='http://127.0.0.1:8080/upload?path=client1/',data=data,
print(r)
r=requests.post(url='http://127.0.0.1:8080/upload?path=client2/',data=data,
print(r)
```

在 2024.01.02 10:44:10 于 71ms内执行

```
<Response [200]>
<Response [403]>
```

MILLENNIUM

Chapter 0x01

Integrated Tests(3)

```
#3.2
url='http://127.0.0.1:8080/delete?path=client1/a.py'

headers={"Authorization": "Basic Y2xpZW50MToxMjM="}
r=requests.post(url=url, headers=headers)
print(r)
在 2023.11.20 20:25:30 于 13ms内执行

<Response [404]>
```

```
#4
url='http://127.0.0.1:8080/'

headers={"Authorization": "Basic Y2xpZW50MToxMjM="}
r=requests.get(url=url, headers=headers)
print(r.cookies.values()[0])
headers={"Cookie":'session-id='+r.cookies.values()[0]}
q=requests.get('http://localhost:8080/',headers=headers)
print(q)
# print(q.cookies)
在 2023.11.20 20:25:32 于 2s 47ms内执行

70bf4c60d3d7d6c1e8cf36665528385838935253 2b34b0050b26983fd685ccc8
<Response [200]>
```

```
#5
headers={"Authorization": "Basic Y2xpZW50MToxMjM="}
r=requests.get(url='http://127.0.0.1:8080/client1/a.txt?chunked=1', headers=headers)
print(r)
在 2023.11.20 20:25:32 于 16ms内执行

<Response [200]>
```

MILLENNIUM

Chapter 0x01

Corner Cases? (Requests)

Robustness of parameters

```
#2
url1='http://127.0.0.1:8080/?SUSTech-HTTP'

headers={"Authorization": "Basic Y2xpZW50MToxMjM="}
r=requests.get(url=url1, headers=headers)
print(r)
print(r.content.decode())
```

在 2024.01.02 10:58:42 于 340ms内执行

```
<Response [200]>
<!DOCTYPE html>
<html lang="en">
```

Robustness of path

```
headers={"Authorization": "Basic Y2xpZW50MToxMjM="}
r=requests.post(url='http://127.0.0.1:8080/upload?path=/client1/'
print(r)
r=requests.post(url='http://127.0.0.1:8080/upload?path=client1',d
print(r)
```

在 2024.01.02 10:57:14 于 78ms内执行

```
<Response [200]>
<Response [200]>
```

MILLENNIUM

Chapter 0x01

Corner Cases? (Requests)

Filename with spaces

```
#2
headers={"Authorization": "Basic Y2xpZW50MToxMjM="}
r=requests.get(url='http://127.0.0.1:8080/hihihi hihihi.txt', headers=headers)
print(r.content.decode())
在 2024.01.02 10:51:53 于 141ms内执行

hihihi hihihi
```

MILLENNIUM

Chapter 0x01

Corner Cases? (Auth)

Authentication entry point

Our AuthManager supports a customized entry point for failed authentication requests.

e.g. Bad (Lack of) credential field, wrong password, etc.

```
In 17
headers={"Authorization": "Basic Y2xpZW50MToxMjM"}
q=requests.head('http://localhost:8080/',headers=headers)
print(q)

在 2024.01.02 10:47:09 于 86ms内执行

<Response [401]>
```

```
IceLocke +1
@app.auth_manager.entry_point()
def authenticate(request: req.Request):
    response = resp.unauthorized_response()
    response.body = login_template.render().encode('utf-8')
    response.headers['Content-Type'] = 'text/html'
    return response
```

MILLENNIUM

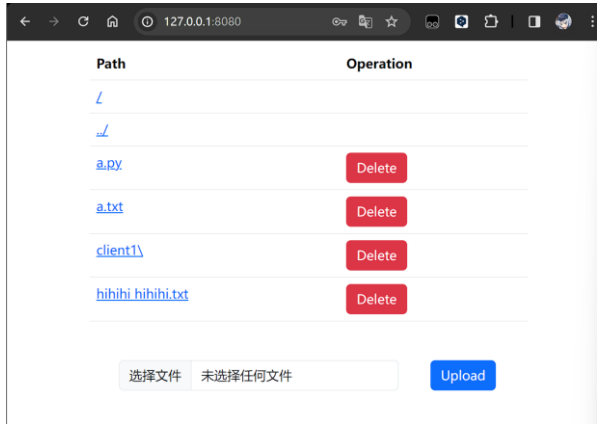
Chapter 0x01

Corner Cases? (Auth)

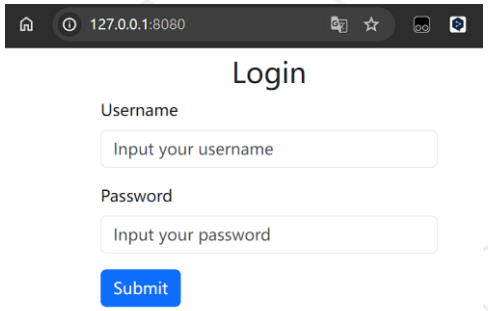
Session flush

Re-login of the same account with flush the `session-id` i.e. a user can’t interact with the system if `session-id` is not shared

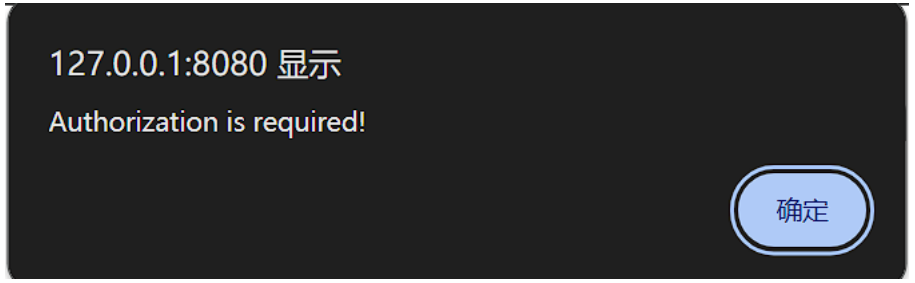
Client 1 @ PC1



Client 1 @ PC2



Another request of Client 1 @ PC1



MILLENNIUM

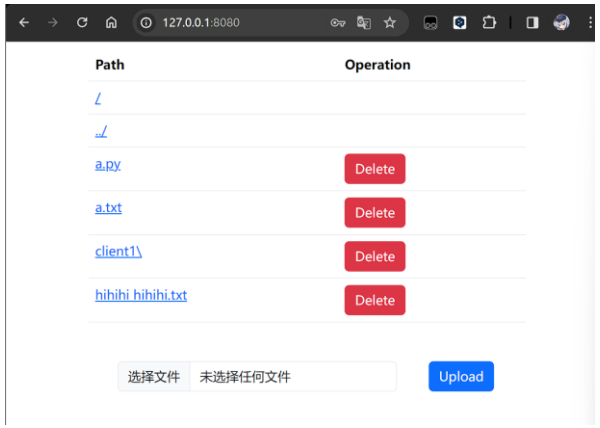
Chapter 0x01

Corner Cases? (Auth)

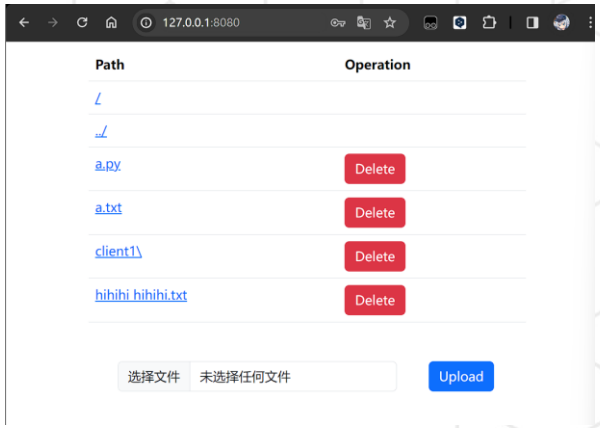
Properties of Set-Cookie

The identity of cookie is not only determined by the key, but also by the path and expiration. So we should manually set the path.

Client 1 GET /client



Client 1 GET /



The cookie path should be set to root path of the server

MILLENNIUM

MILLENNIUM

>_ Bonus

Breakpoint, Encryption & Frontend

Chapter 0x02

Breakpoint

- Breakpoint are mainly supported by dealing with `Range` in request headers.
- First our server parser the `Range` header into a list of tuple indicating the segments that the client requests.
- When generating response body, our server will chop the file into subframes using "`multipart/byteranges; boundary=3d6b6a416f9b5`".

MILLENNIUM

Chapter 0x02

Breakpoint Test Demo

```python
#Breakpoint Transmission
url='http://127.0.0.1:8080/client1/a.txt'

data={}
headers={"Authorization": "Basic Y2xpZW50MToxMjM=",
         "Range": "0-1,1-2,2-3"}
r=requests.get(url=url, data=data, headers=headers)
print(r.content.decode())
```

在 2023.11.20 20:25:32 于 13ms内执行

```
sa
--3d6b6a416f9b5
Content-Type: text/plain
Content-Range: bytes 1-2/11

ad
--3d6b6a416f9b5
Content-Type: text/plain
Content-Range: bytes 2-3/11

df
--3d6b6a416f9b5--
```

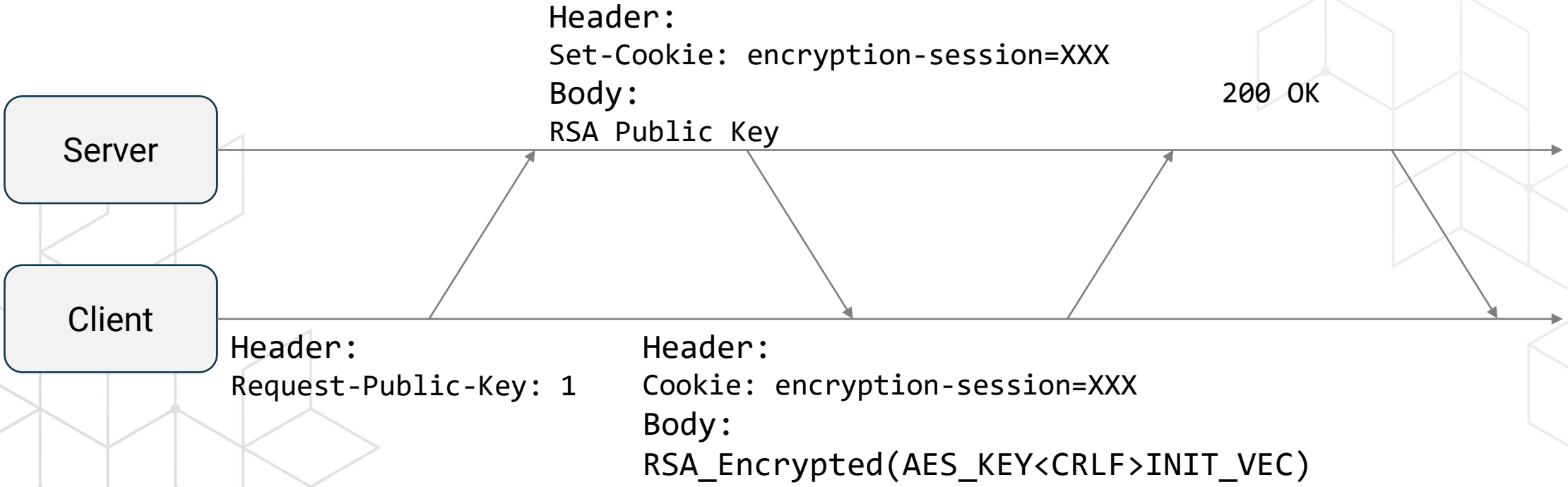
HTTP	261	GET /client1/a.txt HTTP/1.1
TCP	44	8080 → 61355 [ACK] Seq=1 Ack=218 Win=2160896 Len=0
HTTP	593	HTTP/1.1 206 Partial Content (text/plain) (text/plain) (text/plain)
TCP	44	61355 → 8080 [ACK] Seq=218 Ack=550 Win=2160640 Len=0
TCP	44	8080 → 61355 [FIN, ACK] Seq=550 Ack=218 Win=2160896 Len=0

MILLENNIUM

Chapter 0x02

Encryption

- We designed a four-way handshake to establish an encrypted connection.



- After that, all the subsequent bodies in both request and response are encrypted according to `encryption-session`.

MILLENNIUM

Chapter 0x02

Encryption Test Demo

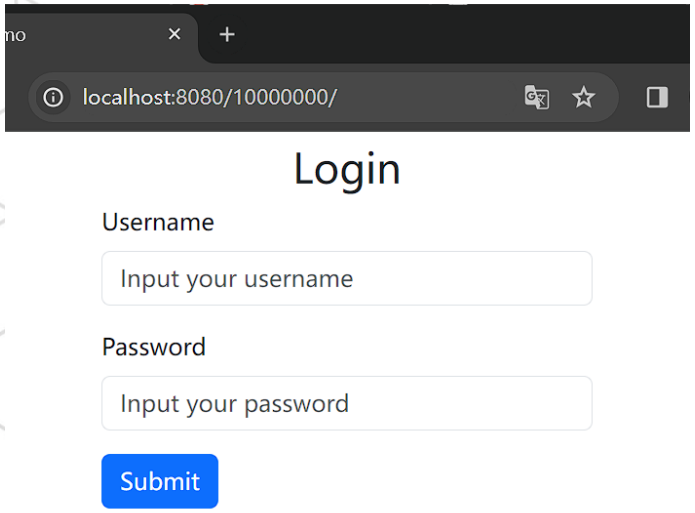
```
1个用法  Artanisax
def main():
    client = Client( url: 'localhost',  port: 8080,  user: 'Artanisax')
    client.handshake()
```

```
C:\Users\IceLocke\PycharmProjects\CS305-Final-Project\venv\Scripts\python.exe
b'Public key: 2142df7cc4f4f2be\r\n1c4d97f51b45c274'
Encryption handshake completed.
b'Encrypted body: \x07\x91\x8c\xcda\xec\x87\xabA\xb9\xa7\xcbD\'\x08\xb8\x9e\x
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>SUSTTP Demo</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstr
```

MILLENNIUM

Chapter 0x02

Frontend



- We use Jinja, Bootstrap 6 and jQuery to implement the frontend with better look.
- We have provide a login page and dynamic content updating (with jQuery ajax) to achieve better experience.

Link to our demo

MILLENNIUM

MILLENNIUM

>_ Thanks for Your Attention!

:>