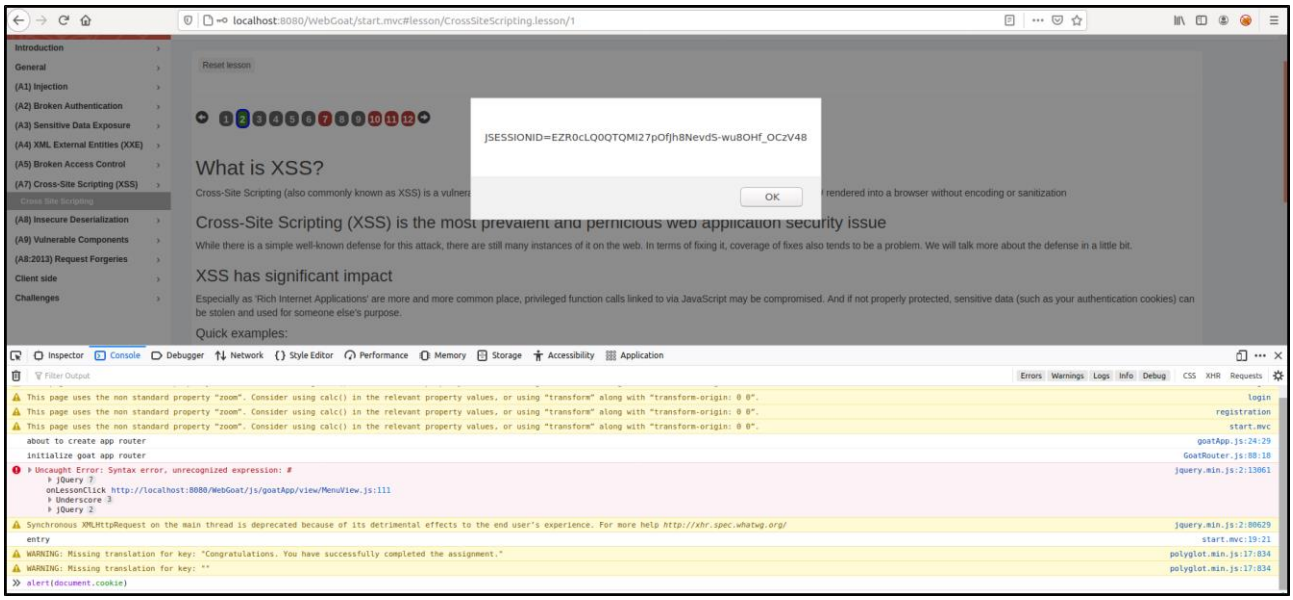
Vinci Nicolò 220229

PART 1: WEBGOAT EXERCISES

Exercise 2

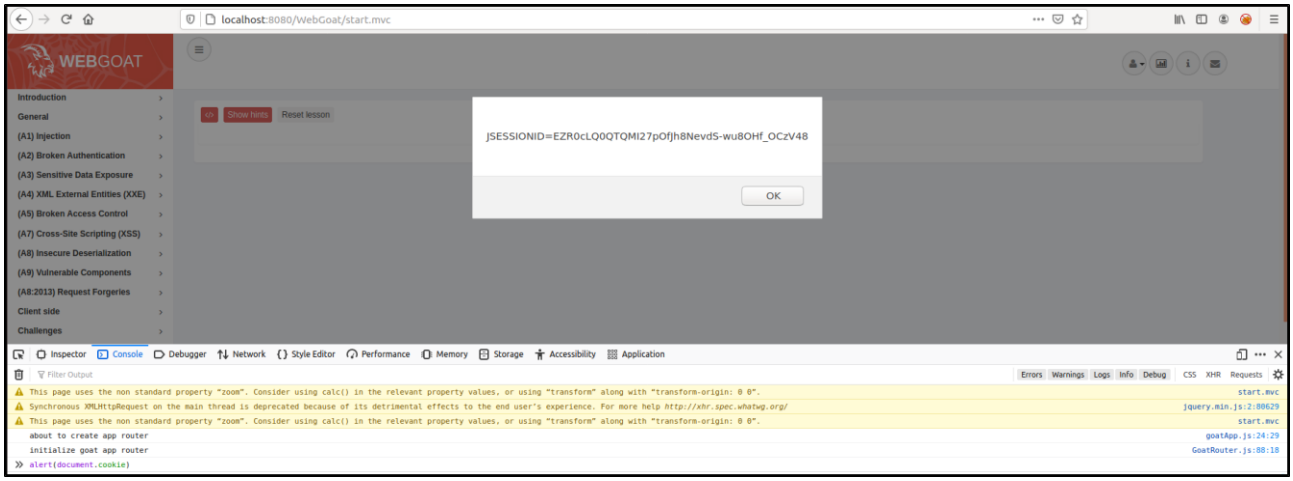
url: http://localhost:8080/WebGoat/start.mvc#lesson/CrossSiteScripting.lesson/1

cookie: JSESSIONID=EZR0cLQ0QTQMI27pOfJh8NevdS-wu8OHf_OCzV48

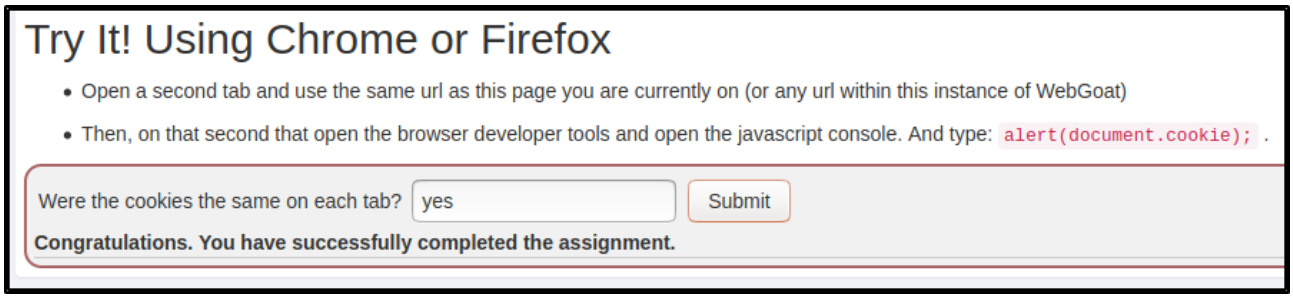


url: http://localhost:8080WebGoat/start.mvc

cookie: JSESSIONID=EZR0cLQ0QTQMI27pOfJh8NevdS-wu8OHf_OCzV48



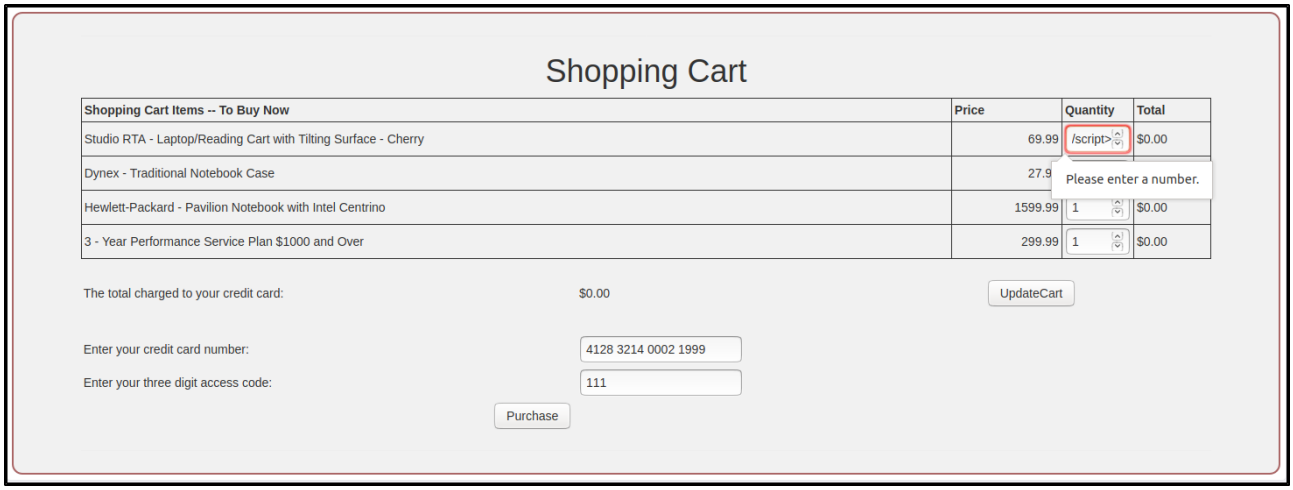
So, both cookies are the same for two different URLs of WebGoat. I conclude that I have found the authentication cookie for my WebGoat session.



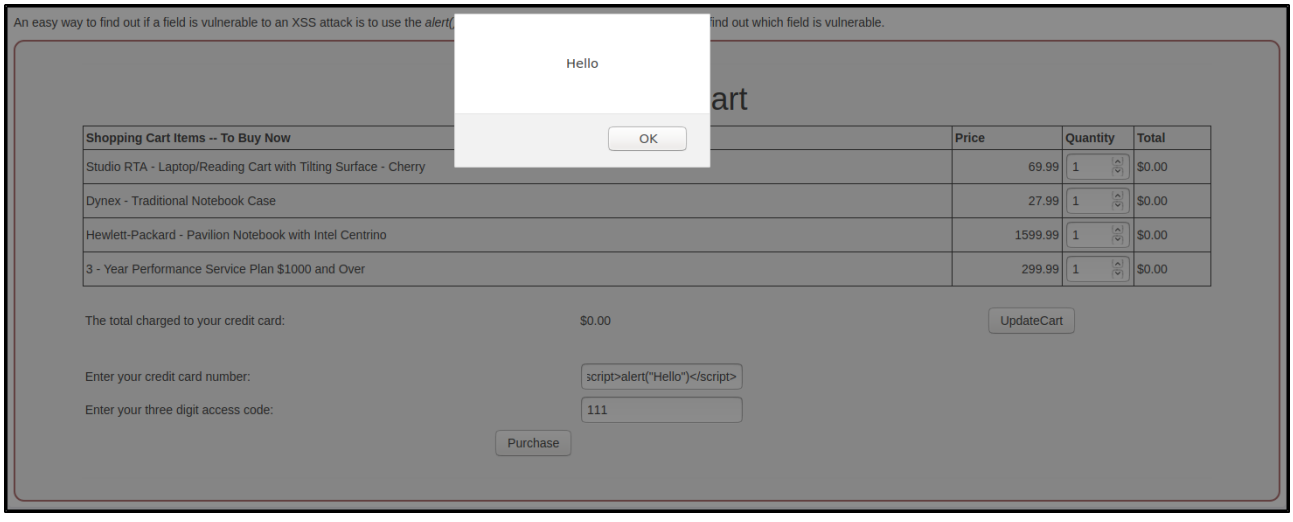
Exercise 7

In this exercise, I have to find a vulnerable XSS field. First of all, I write <script>alert(“Hello”)</script> in all possible field.

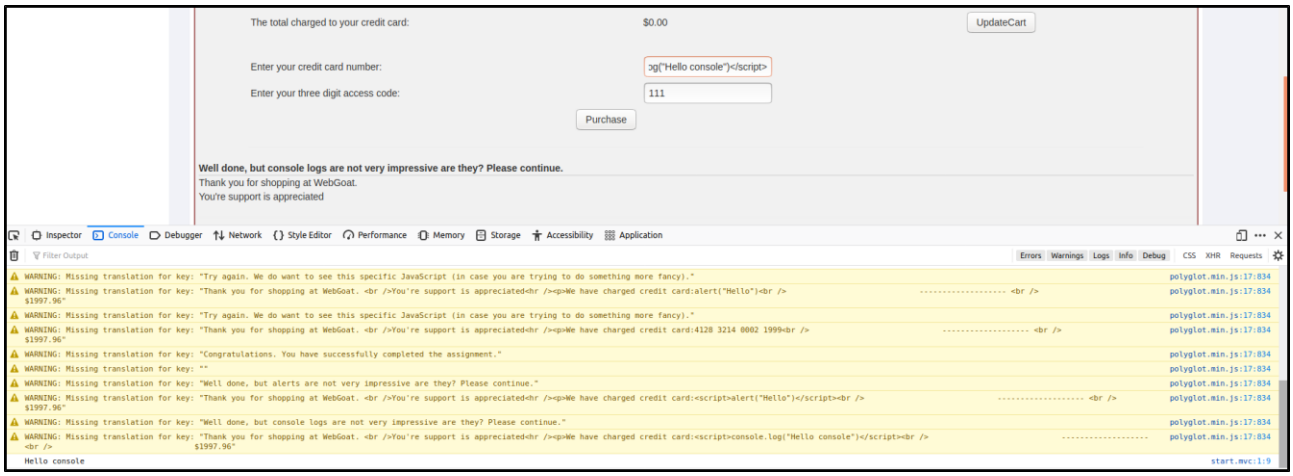
An example of a non-vulnerable field.



An example of a vulnerable field.

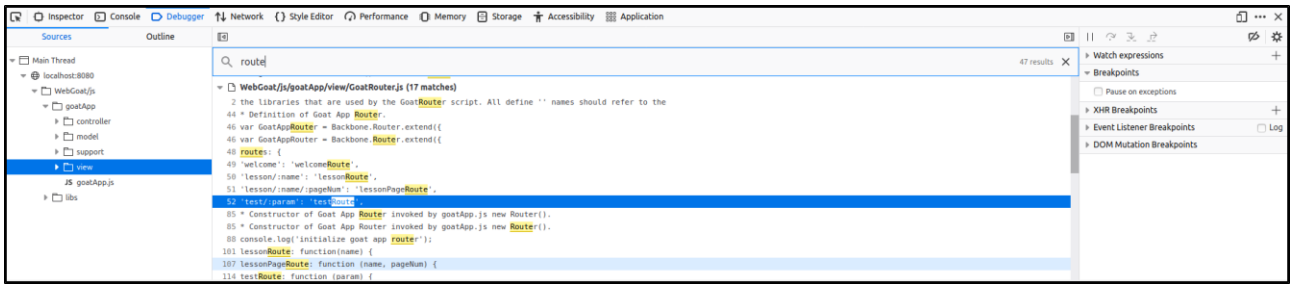


Moreover, I try with <script>console.log(“Hello console”)</script> and I check the console to verify if a field is vulnerable or not.

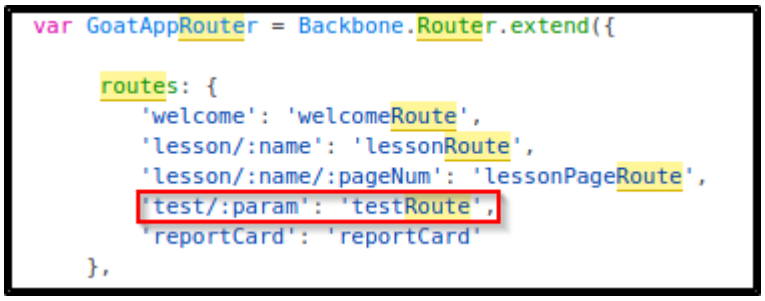


Exercise 10

To find the test route, I look in the source code. I search the word “test”, but there are a lot of results. Hence, I search “route” and I find the source code that manages the route of the page.



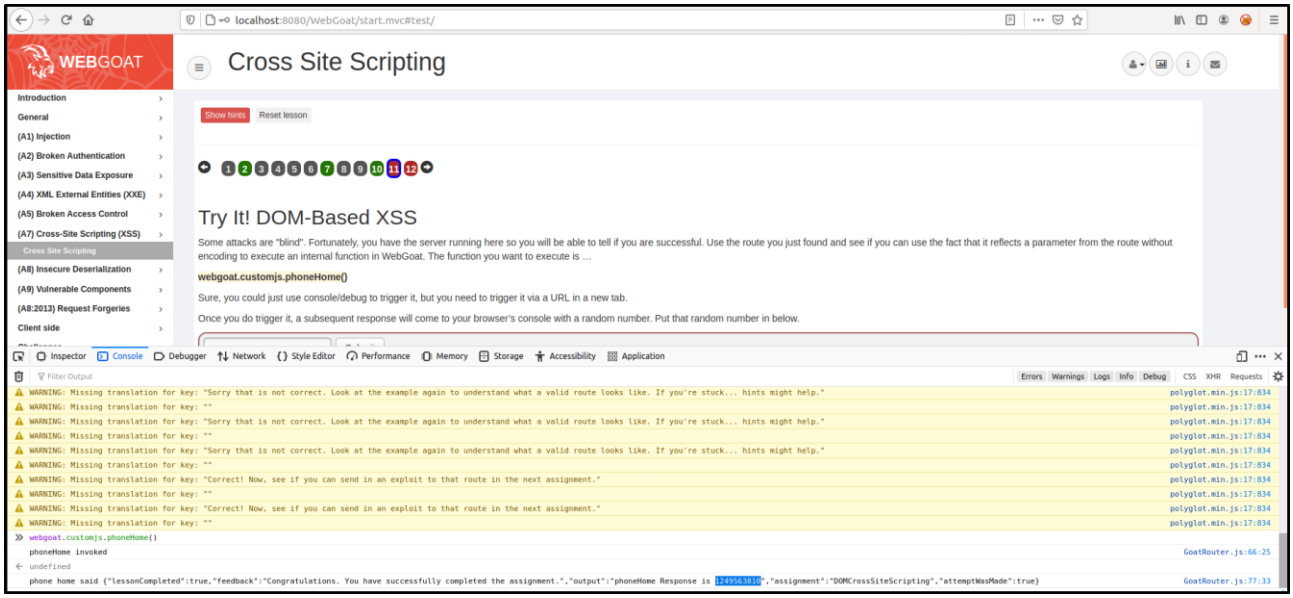
I find this source code and there is a route called “test/ :param” that is a test route probably.



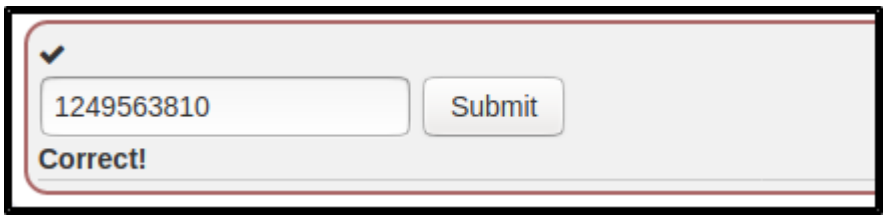
In this case, the base route is “start.mvc#”, so if I type “start.mvc#test/” in the submit field, I finish the exercise.

Exercise 11

I can do this exercise in different ways. The fastest is to go to the test page, thanks to the route “start.mvc#test/” which is found before. Then, I open the console and I simply invoke the function “webgoat.customjs.phoneHome()”. In the response I can find the number to complete the challenge.



Eventually, I put the number and I submit.



Another way is to try to execute the function directly from the URL, typing “start.mvc#test/<script>webgoat.customjs.phoneHome()<%2Fscript>”. Then I can find the response in the console.

Exercise 12

1. Are trusted websites immune to XSS attacks?

- [ ] Solution 1: Yes they are safe because the browser checks the code before executing.
- [ ] Solution 2: Yes because Google has got an algorithm that blocks malicious code.
- [ ] Solution 3: No because the script that is executed will break through the defense algorithm of the browser.
- [x] Solution 4: No because the browser trusts the website if it is acknowledged trusted, then the browser does not know that the script is malicious.

2. When do XSS attacks occur?

- [ ] Solution 1: Data enters a web application through a trusted source.
- [ ] Solution 2: Data enters a browser application through the website.
- [x] Solution 3: The data is included in dynamic content that is sent to a web user without being validated for malicious content.
- [ ] Solution 4: The data is excluded in static content that way it is sent without being validated.

3. What are Stored XSS attacks?

- [x] Solution 1: The script is permanently stored on the server and the victim gets the malicious script when requesting information from the server.
- [ ] Solution 2: The script stores itself on the computer of the victim and executes locally the malicious code.
- [ ] Solution 3: The script stores a virus on the computer of the victim. The attacker can perform various actions now.
- [ ] Solution 4: The script is stored in the browser and sends information to the attacker.

4. What are Reflected XSS attacks?

- [ ] Solution 1: Reflected attacks reflect malicious code from the database to the web server and then reflect it back to the user.
- [x] Solution 2: They reflect the injected script off the web server. That occurs when input sent to the web server is part of the request.
- [ ] Solution 3: Reflected attacks reflect from the firewall off to the database where the user requests information from.
- [ ] Solution 4: Reflected XSS is an attack where the injected script is reflected off the database and web server to the user.

5. Is JavaScript the only way to perform XSS attacks?

- [ ] Solution 1: Yes you can only make use of tags through JavaScript.
- [ ] Solution 2: Yes otherwise you cannot steal cookies.
- [ ] Solution 3: No there is ECMAScript too.
- [x] Solution 4: No there are many other ways. Like HTML, Flash or any other type of code that the browser executes.

PART 2a: EXPLOITING OF THE CODING EXERCISES

Exercise 1

First of all, I run Flask on localhost:5000/ and I go to the only route “hello” that is accepted a parameter.

To write my name in bold is enough to inject “Nicolò in the parameter “name”.

Exercise 2

My solution for this exercise is to modify the “name” parameter in the following way:

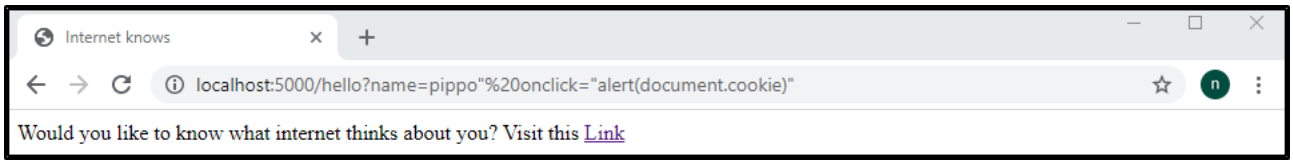
pippo"<button type="button" onclick="alert(‘You have a Virus!’)">Click me</button><!—

To keep the search alive, I put a random word before the button. The “<!—” needs to comment the rest of the html code, so I can have only the “Click me” and the word “pippo” will be searched.

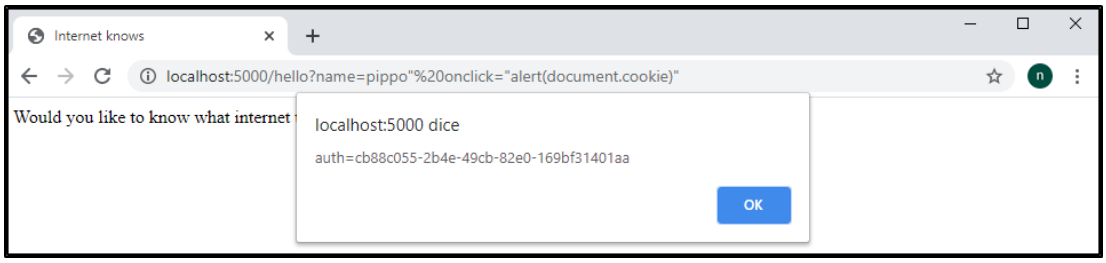
Exercise 3

For the last exercise, I add a JavaScript event “onclick” to the link so that I display the authentication cookie. I write in the “name” parameter:

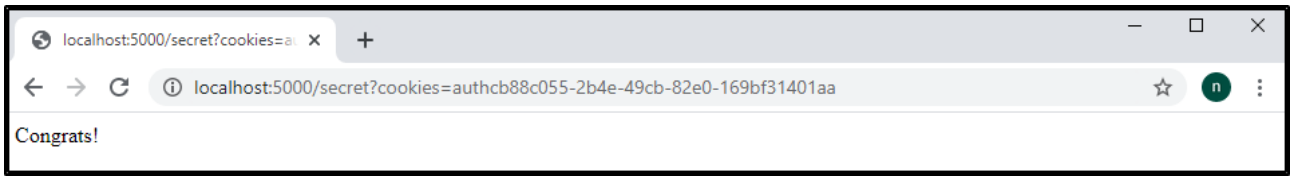
pippo" onclick="alert(document.cookie)"



When I click on “Link”, I can see the actual cookie:



Then, the word “pippo” will be searched. At the end, I pass the cookie to the secret page and I have the “Congrats!”.



PART 2b: FIXING CODE

To fix all the exercises, I add the escape function to parse the user input. I attach the fixed exercises.

EXERCISE 1

```
from flask import Flask, request
from html import escape
app = Flask(__name__)

@app.route("/hello")
def hello():
    name = request.args.get('name')
    content = """
    <html>
        <head><title>Hello Website</title></head>
        <body>
            Hello {}
        </body>
    </html>
    """.format(escape(name))
    return content

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

EXERCISE 2

```
from flask import Flask, request
from html import escape
app = Flask(__name__)

@app.route("/hello")
def hello():
    name = request.args.get('name')
    content = """
    <html>
        <head><title>Internet knows</title></head>
        <body>
            Would you like to know what internet thinks about you? Visit this <a hr
ef="https://www.bing.com/search?q={}" attribute="aaa">Link</a>
        </body>
    </html>
    """.format(escape(name))
    return content

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

EXERCISE 3

```
from flask import Flask, request, make_response
import uuid
from html import escape
app = Flask(__name__)

your_cookie = None

@app.route("/hello")
def hello():
    global your_cookie
    name = request.args.get('name')
    content = """
    <html>
        <head><title>Internet knows</title></head>
        <body>
            Would you like to know what internet thinks about you? Visit this <a hr
ef="https://www.bing.com/search?q={}" attribute="aaa">Link</a>
        </body>
    </html>
    """.format(escape(name))
    resp = make_response(content)
    your_cookie = str(uuid.uuid4())
    resp.set_cookie('auth', your_cookie)
    return resp

@app.route("/secret")
def secret():
    global your_cookie
    print("your_cookie", your_cookie)
    cookies = request.args.get('cookies')
    if your_cookie and your_cookie in cookies:
        print("Congrats!")
        return "Congrats!", 200
    else:
        return "Wrong cookie", 403

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