Web Application Development

© Alexander Menshchikov, ITMO 2022

Data Storage and Authentication

Data storage

Where to store data?

On the client side

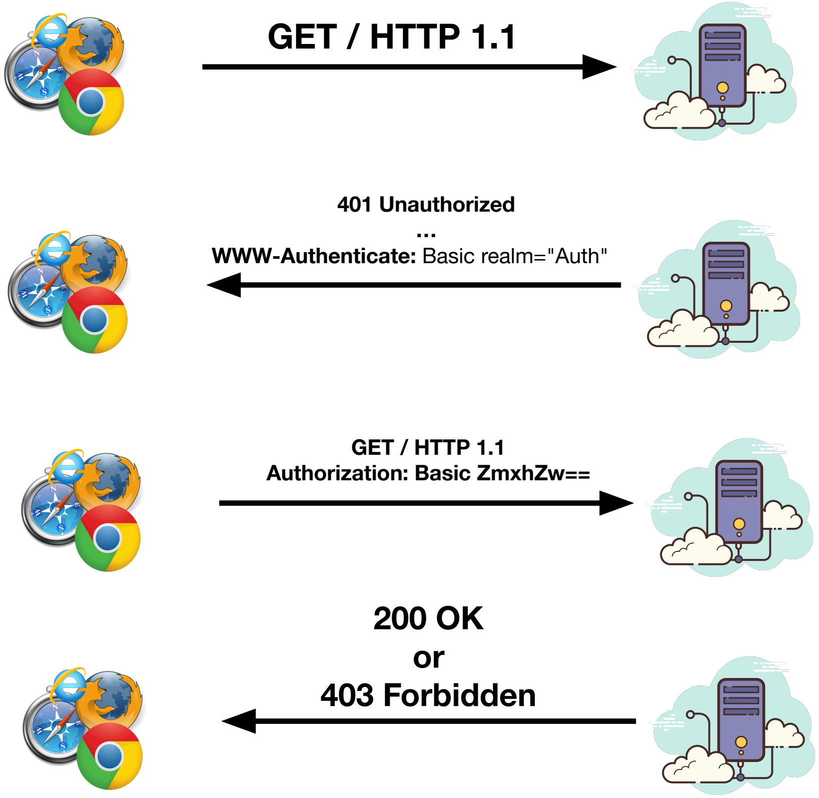
- In HTML
- In Cookie
- In local storage, etc

On the server side

- In global variable
- In file
- In database

Authentication

Basic Authentication



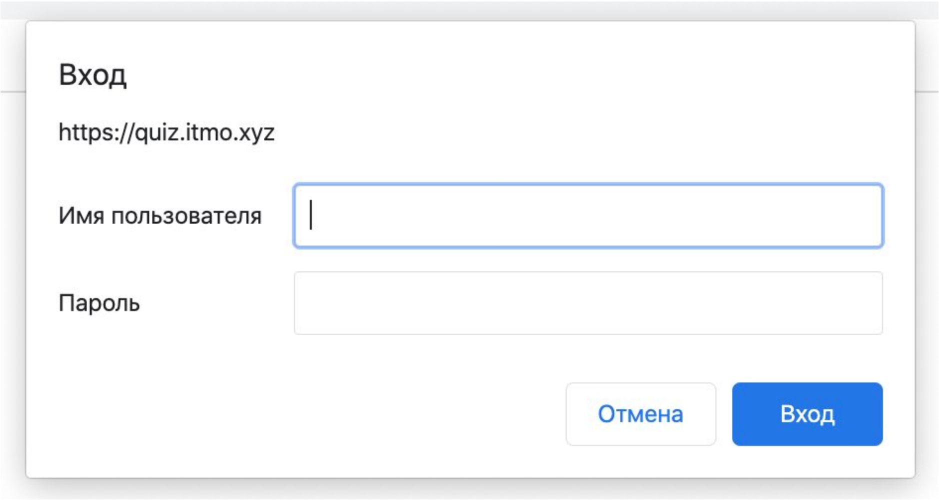
GET / HTTP 1.1

401 Unauthorized
...
WWW-Authenticate: Basic realm="Auth"

GET / HTTP 1.1
Authorization: Basic ZmxhZw==

200 OK
or
403 Forbidden

Basic Authentication



Вход

https://quiz.itmo.xyz

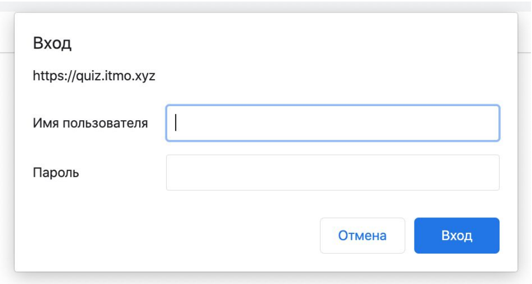
Имя пользователя

Пароль

Basic Authentication

- 👎 Username and password in every request
    - Less vulnerable over HTTPS
- 👎 Password is the same and cannot be changed frequently
- 👍 Simple and fast
- 👍 Can be provided on the web-server side

Base64 and basic auth

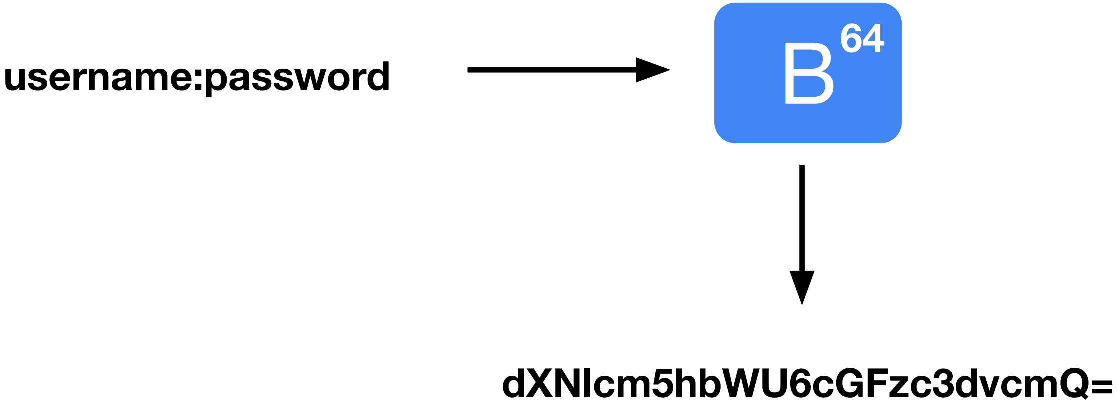


Вход

https://quiz.itmo.xyz

Имя пользователя

Пароль



username:password

dXNlcm5hbWU6cGFzc3dvcmQ=

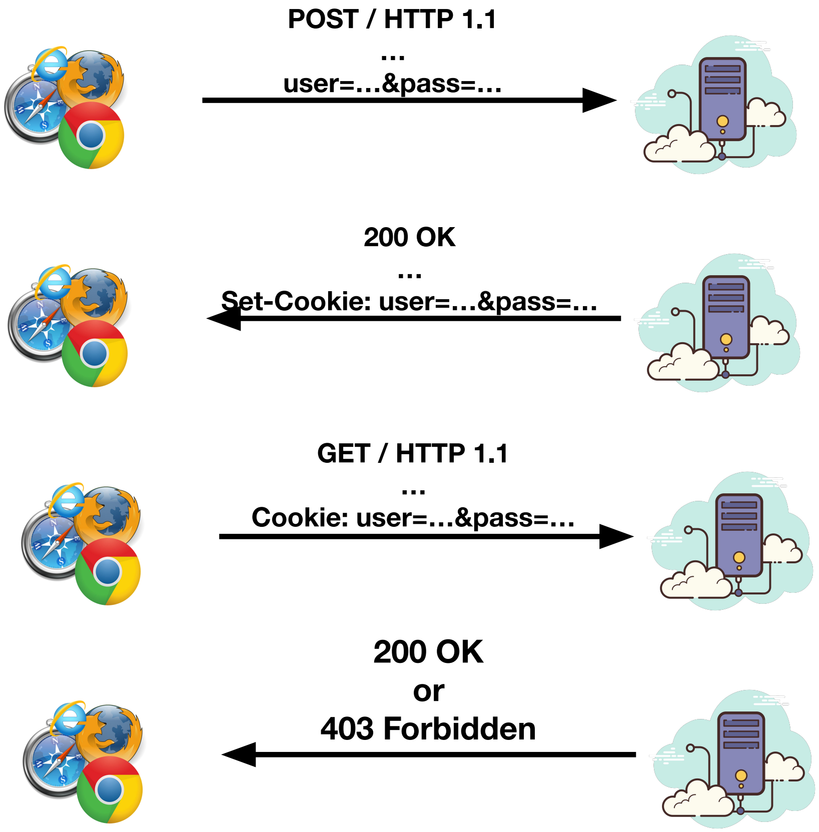
PBKDF2

https://tedboy.github.io/flask/generated/werkzeug.generate_password_hash.html

```
users = {
   "user": generate_password_hash("user"),
   "guest": generate_password_hash("guest")
}
```

```
{'user':
'pbkdf2:sha256:260000$UAbOsqIzJsajPI86$5d1c01542d7338c28874bac891e30c69ca4ec0c001386b2366c7ae5f6f5
4b7cb',

'guest':
'pbkdf2:sha256:260000$6UKfv29pZrTtsbhj$e9e3a0cacdafab9af840985ea82769923ea2048832241695e35aa6108c6c
fcb0'}
```

Cookie based auth



POST / HTTP 1.1
...
user=...&pass=...

200 OK
...
Set-Cookie: user=...&pass=...

GET / HTTP 1.1
...
Cookie: user=...&pass=...

200 OK
or
403 Forbidden

Cookie based auth

👎 Username and password in every request

Can be encrypted

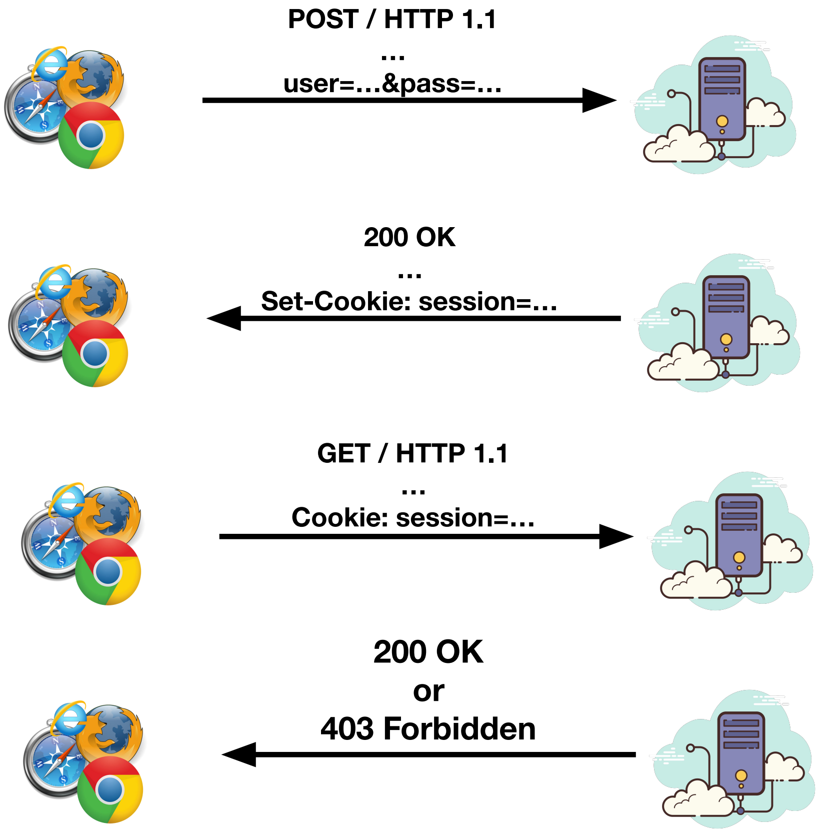
👎 Credentials can be extracted via JavaScript

Can be protected

👍 Still simple and fast

👍 ~Stateless

Session based auth

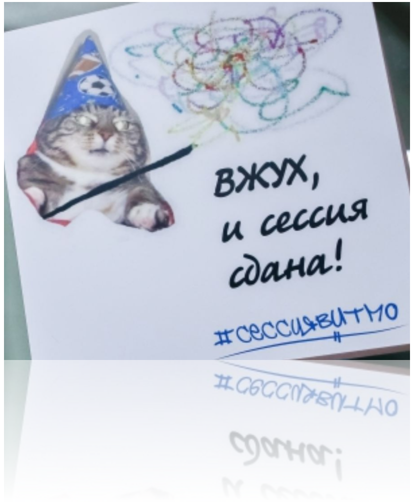


POST / HTTP 1.1
...
user=...&pass=...

200 OK
...
Set-Cookie: session=...

GET / HTTP 1.1
...
Cookie: session=...

200 OK
or
403 Forbidden



Session based auth

👍 **No password transfer**

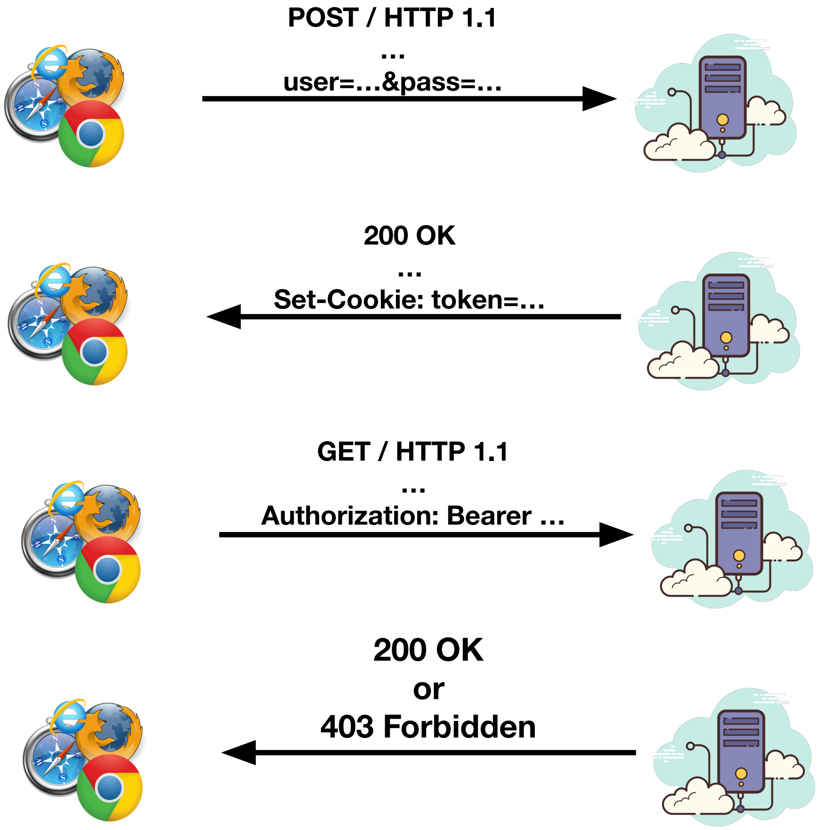
👎 **The same authentication token in every request**

👎 **Token can be extracted via JavaScript**
Can be protected

👍 **Simple**

Not stateless

Token based auth



POST / HTTP 1.1
...
user=...&pass=...

200 OK
...
Set-Cookie: token=...

GET / HTTP 1.1
...
Authorization: Bearer ...

200 OK
or
403 Forbidden

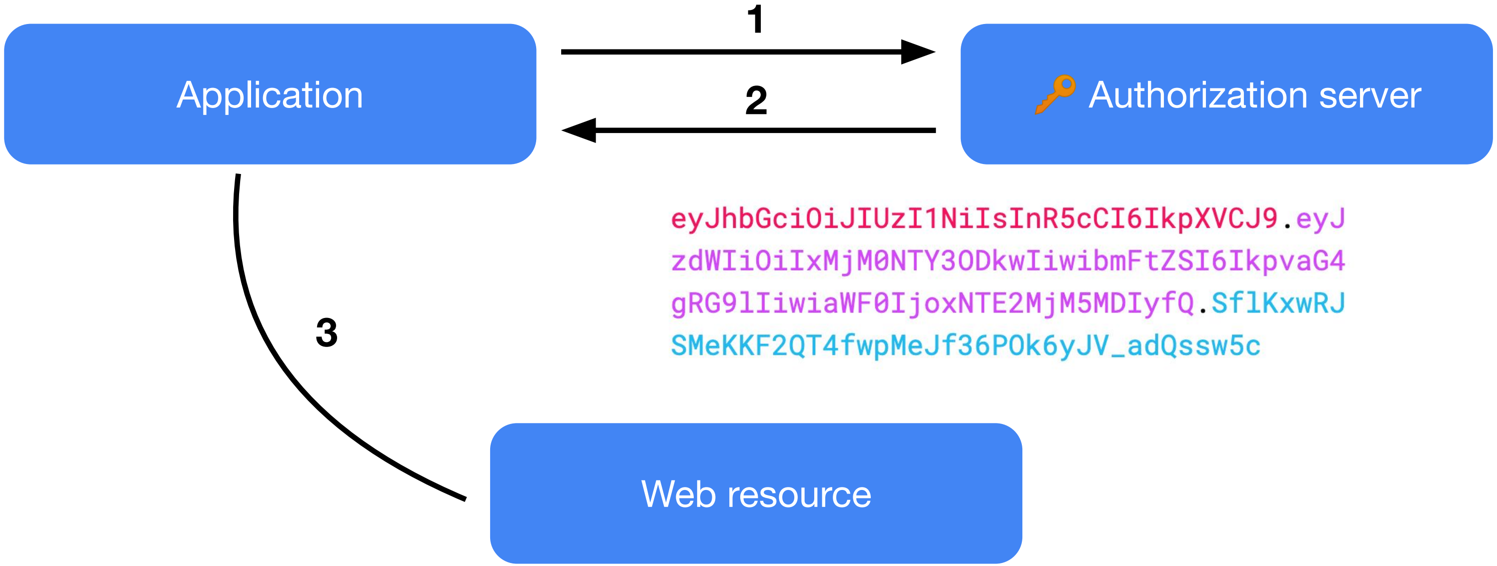
Token based auth

👍 **No password transfer**

👎 **More complicated**

👍 **Stateless**

JWT



```
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ
zdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4
gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJ
SMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c
```

https://jwt.io/

Literature

- HTTP Authentication – https://developer.mozilla.org/en-US/docs/Web/HTTP/Authentication
- Base64 – https://en.wikipedia.org/wiki/Base64
- JWT – https://jwt.io/

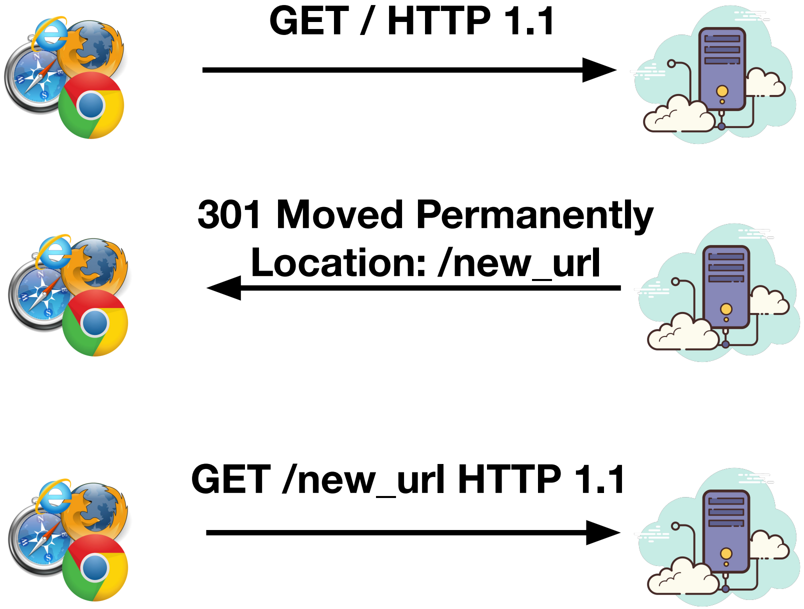
Flask

Redirect

- Expanding the reach
  - Synonyms (www.quiz.itmo.xyz –> quiz.itmo.xyz)
  - Typos (www.quis.itmo.xyz –> quiz.itmo.xyz)
- Moving to a new domain
  - ifmo.su –> codex.so
- Force HTTPS
  - http://wad.itmo.xyz –> https://wad.itmo.xyz

Use cases

Redirect



GET / HTTP 1.1

301 Moved Permanently
Location: /new_url

GET /new_url HTTP 1.1

301 Moved Permanently
308 Permanent redirect

GET

POST

302 Found
307 Temporary Redirect

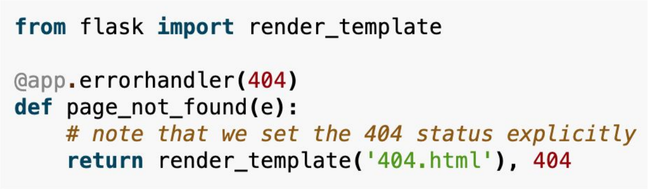
GET

POST

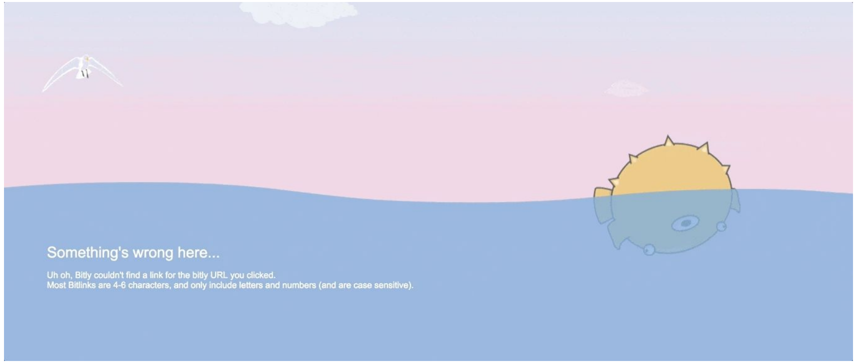
304 Not Modified

Errors

- Common error codes
  - ○ 404 Not Found
  - ○ 403 Forbidden
  - ○ 500 Internal Server Error



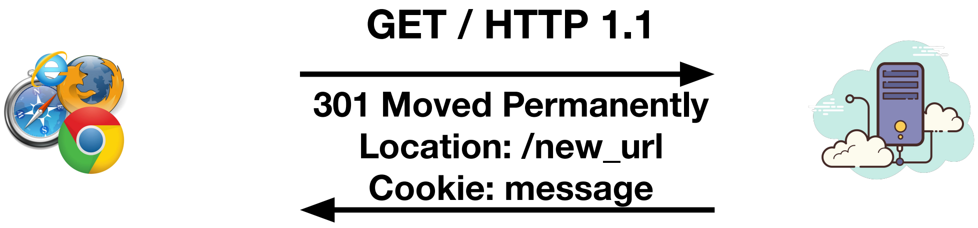
```python
from flask import render_template

@app.errorhandler(404)
def page_not_found(e):
    # note that we set the 404 status explicitly
    return render_template('404.html'), 404
```



Something's wrong here...

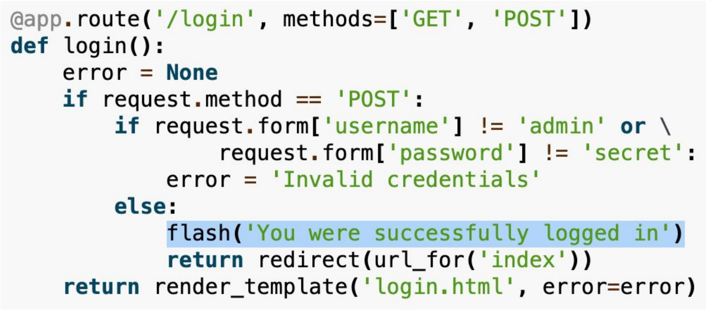
Uh oh, Bitly couldn't find a link for the bitly URL you clicked.
Most Bitlinks are 4-6 characters, and only include letters and numbers (and are case sensitive).

Flash

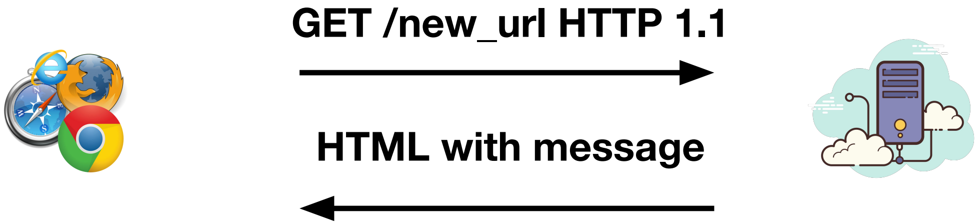


GET / HTTP 1.1

301 Moved Permanently
Location: /new_url
Cookie: message

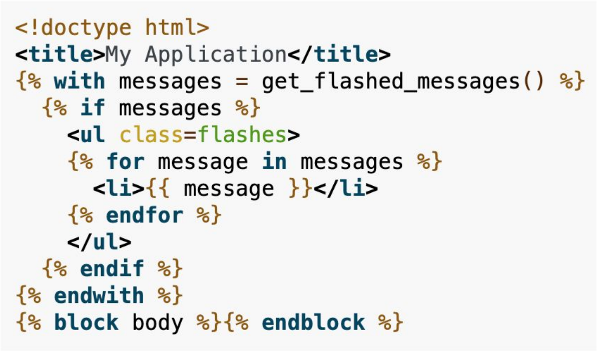


```
@app.route('/login', methods=['GET', 'POST'])
def login():
    error = None
    if request.method == 'POST':
        if request.form['username'] != 'admin' or \
                request.form['password'] != 'secret':
            error = 'Invalid credentials'
        else:
            flash('You were successfully logged in')
            return redirect(url_for('index'))
    return render_template('login.html', error=error)
```



GET /new_url HTTP 1.1

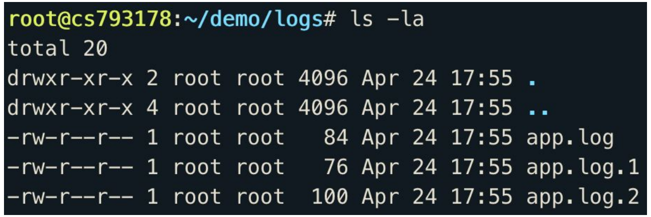
HTML with message



```
<!doctype html>
<title>My Application</title>
{% with messages = get_flashed_messages() %}
  {% if messages %}
    <ul class=flashes>
    {% for message in messages %}
      <li>{{ message }}</li>
    {% endfor %}
    </ul>
  {% endif %}
{% endwith %}
{% block body %}{% endblock %}
```

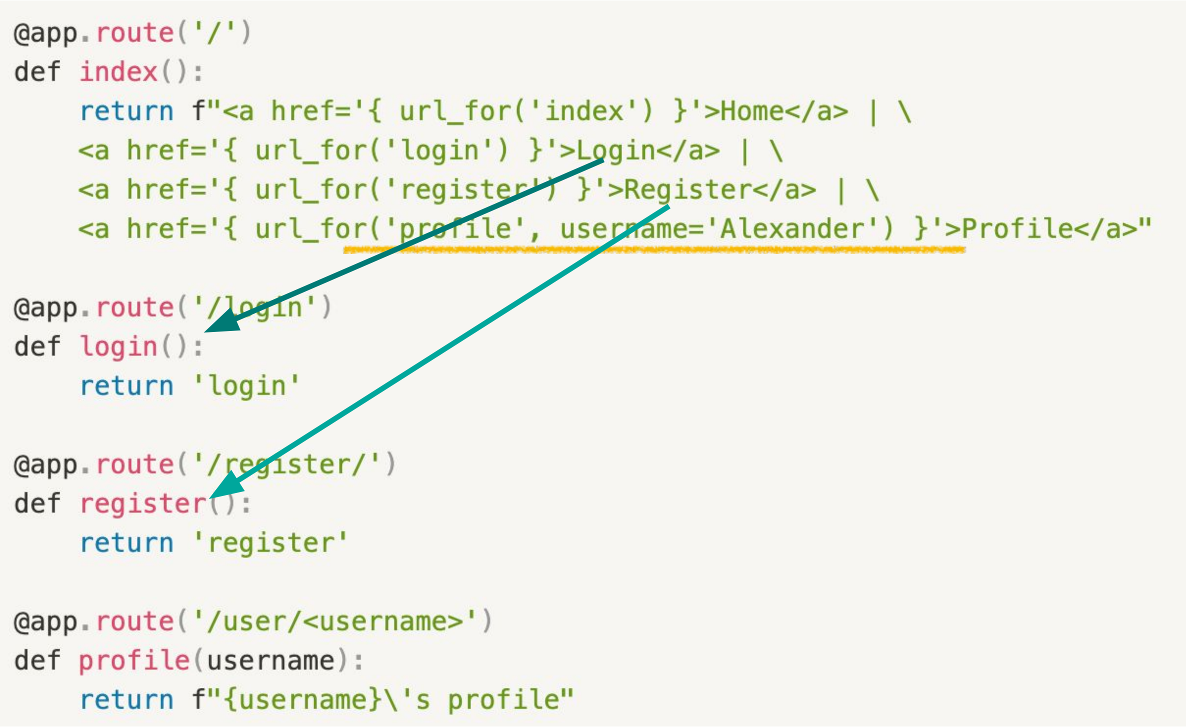
Logging

```
from flask import Flask, render_template, request
from logging.handlers import RotatingFileHandler
import logging


app = Flask(__name__)

handler = logging.handlers.RotatingFileHandler('logs\\app.log', maxBytes=32, backupCount=2)
handler.setLevel(logging.DEBUG)
handler.setFormatter(logging.Formatter('%(asctime)s [in %(pathname)s:%(lineno)d]: %(message)s '))

app.logger.addHandler(handler)
app.logger.setLevel(logging.DEBUG)
app.logger.info('This message goes to stderr and app.log!')
```



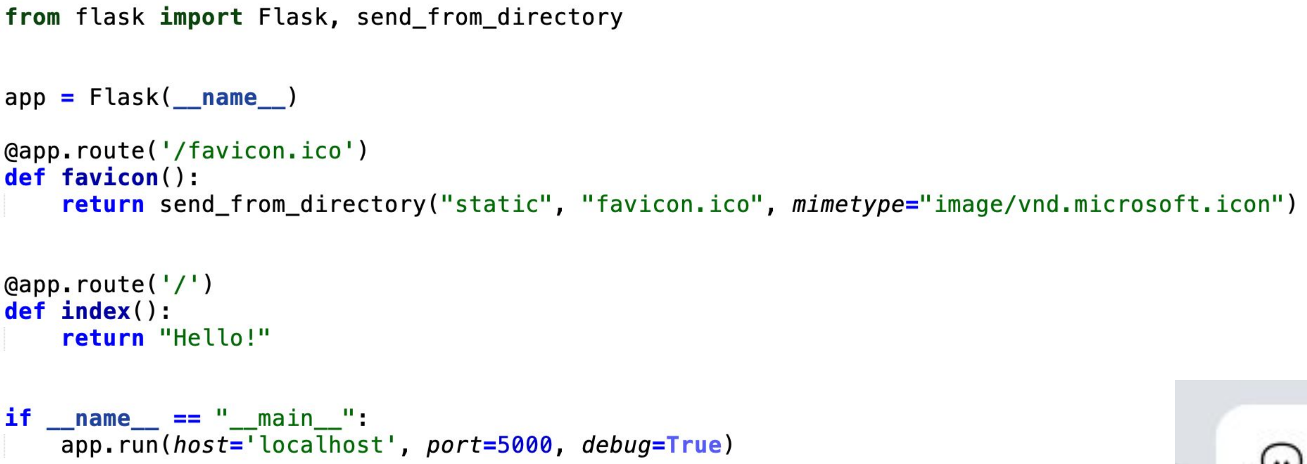
```
root@cs793178:~/demo/logs# ls -la
total 20
drwxr-xr-x 2 root root 4096 Apr 24 17:55 .
drwxr-xr-x 4 root root 4096 Apr 24 17:55 ..
-rw-r--r-- 1 root root   84 Apr 24 17:55 app.log
-rw-r--r-- 1 root root   76 Apr 24 17:55 app.log.1
-rw-r--r-- 1 root root  100 Apr 24 17:55 app.log.2
```

Url for

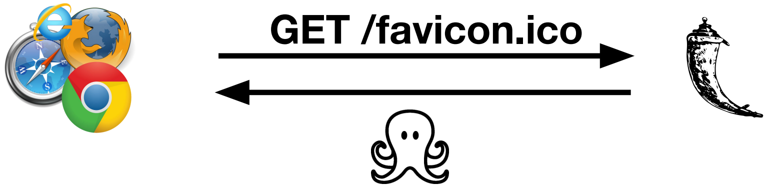


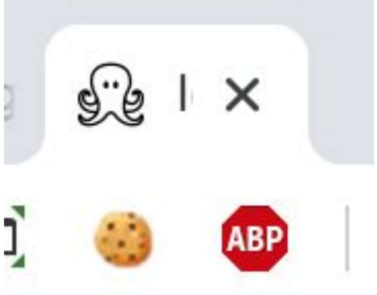
```
@app.route('/')
def index():
    return f"<a href='{ url_for('index') }'>Home</a> | \
    <a href='{ url_for('login') }'>Login</a> | \
    <a href='{ url_for('register') }'>Register</a> | \
    <a href='{ url_for('profile', username='Alexander') }'>Profile</a>"

@app.route('/login')
def login():
    return 'login'

@app.route('/register/')
def register():
    return 'register'

@app.route('/user/<username>')
def profile(username):
    return f"{username}\'s profile"
```

Favicon

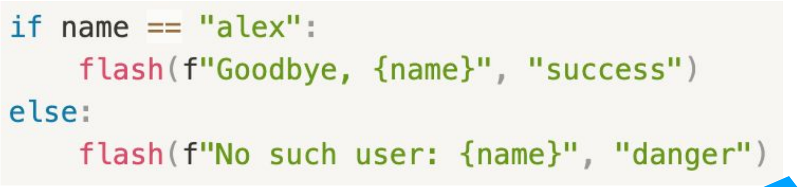


```
from flask import Flask, send_from_directory


app = Flask(__name__)

@app.route('/favicon.ico')
def favicon():
    return send_from_directory("static", "favicon.ico", mimetype="image/vnd.microsoft.icon")


@app.route('/')
def index():
    return "Hello!"


if __name__ == "__main__":
    app.run(host='localhost', port=5000, debug=True)
```

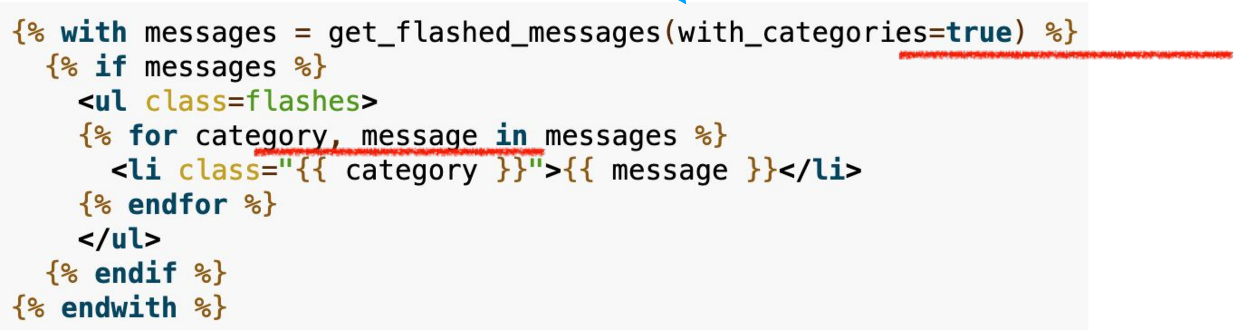




Flash with categories



```
if name == "alex":
    flash(f"Goodbye, {name}", "success")
else:
    flash(f"No such user: {name}", "danger")
```



```
{% with messages = get_flashed_messages(with_categories=true) %}
  {% if messages %}
    <ul class=flashes>
    {% for category, message in messages %}
      <li class="{{ category }}">{{ message }}</li>
    {% endfor %}
    </ul>
  {% endif %}
{% endwith %}
```

File upload

GET /

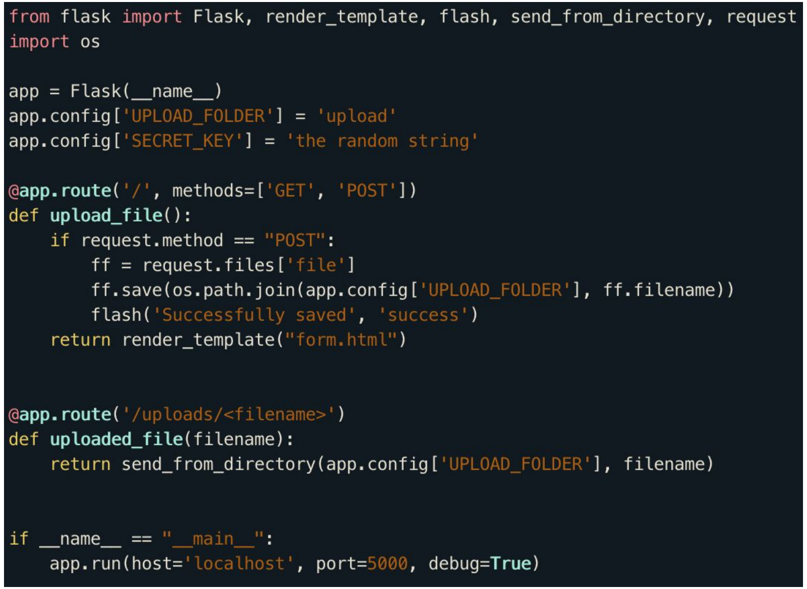
```
<form method=POST enctype="multipart/form-data>"

    <input type=file name=file>

    <input type=submit value=Upload>

</form>
```

POST /

- **Check if file was sent**
- **Check that filename is not empty**
- **Check that file extension is allowed**

GET /uploads/<filename>

File upload

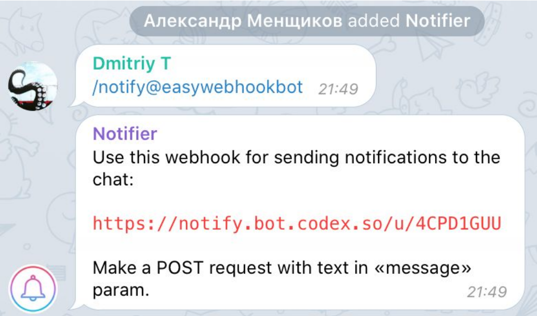


```python
from flask import Flask, render_template, flash, send_from_directory, request
import os

app = Flask(__name__)
app.config['UPLOAD_FOLDER'] = 'upload'
app.config['SECRET_KEY'] = 'the random string'

@app.route('/', methods=['GET', 'POST'])
def upload_file():
    if request.method == "POST":
        ff = request.files['file']
        ff.save(os.path.join(app.config['UPLOAD_FOLDER'], ff.filename))
        flash('Successfully saved', 'success')
    return render_template("form.html")


@app.route('/uploads/<filename>')
def uploaded_file(filename):
    return send_from_directory(app.config['UPLOAD_FOLDER'], filename)


if __name__ == "__main__":
    app.run(host='localhost', port=5000, debug=True)
```

```python
@app.route('/uploads/<filename>')
def uploaded_file(filename):
    return send_from_directory(app.config['UPLOAD_FOLDER'], filename)


@app.route('/', methods=['GET', 'POST'])
def upload_file():
    if request.method == 'POST':
        file = request.files['file']
        file.save(app.config['UPLOAD_FOLDER'] + file.filename)

        flash('Successfully saved', 'success')
        return f"<a href='/uploads/{file.filename}'>uploaded file
{file.filename}</a>"

    return render_template("form.html")
```

Requests

@easywebhookbot



Александр Менщиков added Notifier

Dmitriy T
/notify@easywebhookbot 21:49

Notifier
Use this webhook for sending notifications to the chat:

https://notify.bot.codex.so/u/4CPD1GUU

Make a POST request with text in «message» param. 21:49

```
@app.errorhandler(InternalServerError)
def handle_500(e):
    requests.post("https://notify.bot.codex.so/u/4CPD1GUU", {
        "message": f"*Exception* on the server: `{str(e)}`",
        "parse_mode": "Markdown"
    })
    return str(e), 500
```

Literature

- Redirect – https://flask.palletsprojects.com/en/1.1.x/quickstart/#redirects-and-errors
- Error handlers – https://flask.palletsprojects.com/en/1.1.x/patterns/errorpages/
- Flashing – https://flask.palletsprojects.com/en/1.1.x/patterns/flashing/

Literature

- Logging: https://flask.palletsprojects.com/en/1.1.x/logging/
- Url-building: https://flask.palletsprojects.com/en/1.1.x/quickstart/#url-building
- Favicon: https://flask.palletsprojects.com/en/1.1.x/patterns/favicon/
- Fileupload: https://flask.palletsprojects.com/en/1.1.x/patterns/fileuploads/
- Python Requests: https://github.com/psf/requests