Веб-сервер на Flask
Введение

Наше первое приложение

```
fat@ptest:~/flask$ cat hello.py
from flask import Flask, request, session, g, redirect, url_for, abort, render_template, flash

app=Flask(__name__)
app.config.from_object(__name__)
app.config.update(dict(
        MESSAGE="Hello, world"
    ))

@app.route('/')
def hello():
   return app.config['MESSAGE']
```

Запускаем

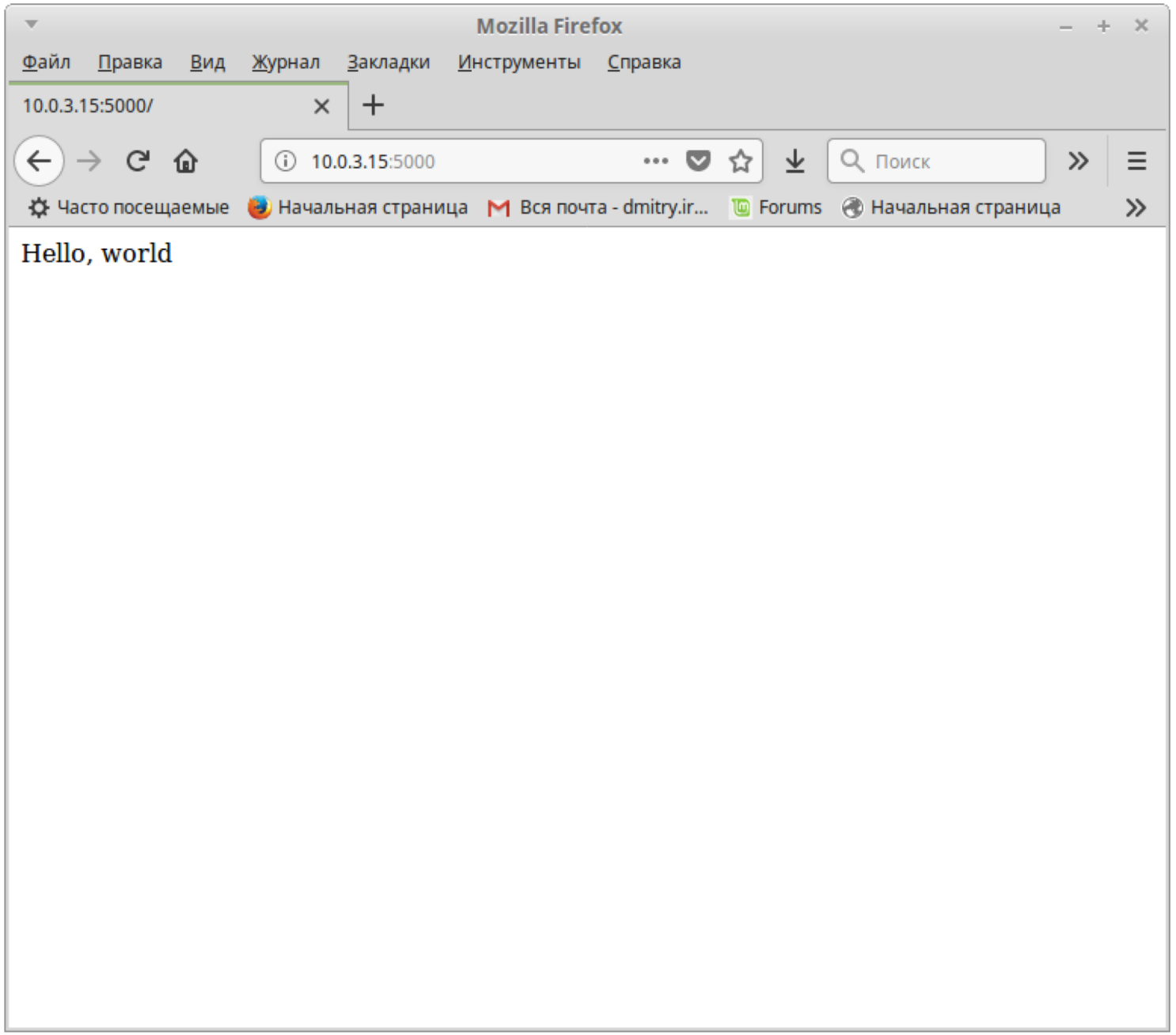
fat@ptest:~/flask$ FLASK_APP=hello.py python3 -m flask run \ --host=0.0.0.0

* Serving Flask app "hello"

* Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)

10.0.3.1 - - [29/Mar/2018 06:17:10] "GET / HTTP/1.1" 200 -

Смотрим


Mozilla Firefox
Файл Правка Вид Журнал Закладки Инструменты Справка
10.0.3.15:5000/
10.0.3.15:5000
Поиск
Часто посещаемые
Начальная страница
Вся почта - dmitry.ir...
Forums
Начальная страница
Hello, world

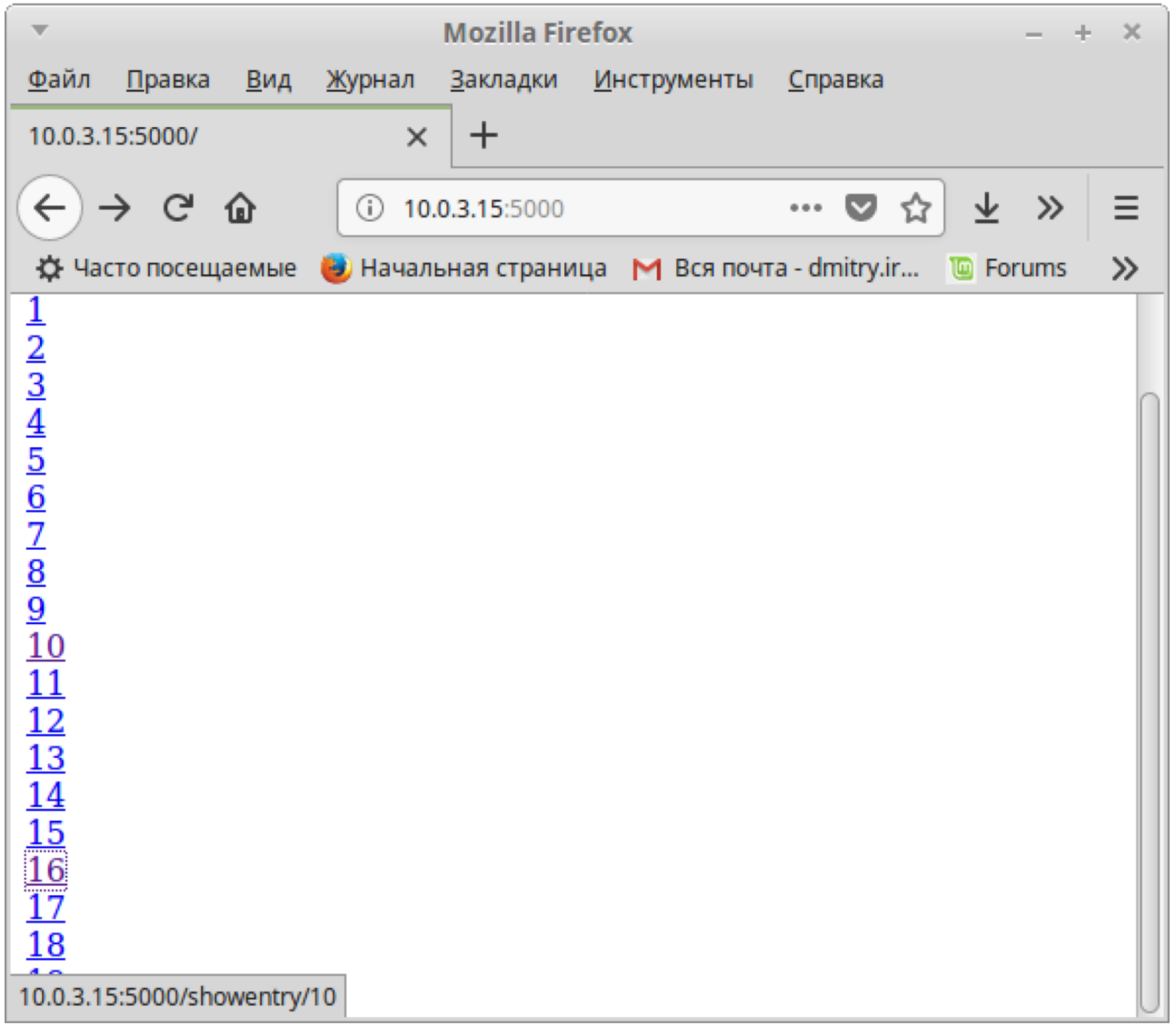
Давайте что-нибудь поинтереснее

```
import os
import dbm
from flask import Flask, request, session, g, redirect, url_for, abort, render_template, flash

app=Flask(__name__)
app.config.from_object(__name__)
app.config.update(dict(  DATABASE=os.path.join(app.root_path, "storage")    ))

@app.route('/')
def show_entries():
  e=''
  with dbm.open(app.config['DATABASE'],'r') as db:
    for i in sorted(db.keys(), key=int):
        e=e + ('<br>'+i.decode('utf-8'))
  return e
```

Или еще интереснее

```
@app.route('/')
def show_entries():
  e=''
  with dbm.open(app.config['DATABASE'],'r') as db:
    for i in sorted(db.keys(), key=int):
      d=i.decode('utf-8')
      e=e + ('<br>'+'<a href="'+url_for('showentry', id=d)+'">'+d+'</a>')
  return e

@app.route('/showentry/<id>')
def showentry(id):
  with dbm.open(app.config['DATABASE'], 'r') as db:
    e=db[id]
  return e
```

Что получилось?



Еще интересные вещи

```
from flask import abort, redirect, url_for, render_template

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

@app.route('/login')
def login():
    abort(401)
    this_is_never_executed()

@app.errorhandler(404)
def page_not_found(error):
    return render_template('page_not_found.html'), 404
```

Обработка форм

```
@app.route('/login', methods=['POST', 'GET'])
def login():
    error = None
    if request.method == 'POST':
        if valid_login(request.form['username'],
                       request.form['password']):
            return log_the_user_in(request.form['username'])
        else:
            error = 'Invalid username/password'
    # the code below is executed if the request method
    # was GET or the credentials were invalid
    return render_template('login.html', error=error)
```

Куки

```
from flask import request
@app.route('/')
def index():
    username = request.cookies.get('username')
    # use cookies.get(key) instead of cookies[key] to not get a
    # KeyError if the cookie is missing.
---------------------------------
from flask import make_response
@app.route('/')
def index():
    resp = make_response(render_template(...))
    resp.set_cookie('username', 'the username')
    return resp
```

Сесии

```
@app.route('/')
def index():
    if 'username' in session:
        return 'Logged in as %s' % escape(session['username'])
    return 'You are not logged in'

@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        session['username'] = request.form['username']
        return redirect(url_for('index'))
    return render_template("login.html")

@app.route('/logout')
def logout():
    # remove the username from the session if it's there
    session.pop('username', None)
    return redirect(url_for('index'))
```

Еще сессии

- Базовый движок Flask все данные сессии хранит в куке
  - Можно хранить только строки
  - Можно столкнуться с ограничением на размер
  - Все, кто действительно хочет, может их просмотреть
  - Это просто Base64-encoded JSON
- Кука подписывается (не шифруется!!!) секретным ключом
  - app.secret_key = 'A0Zr98j/3yX R~XHH!jmN]LWX/,?RT’
  - Параметр конфигурации SECRET_KEY
- Есть расширения, например, Flask-Session, которые предоставляют более продвинутые сессии с хранением данных на сервере

Глобальные переменные

- Переменная g
- В нее можно добавлять атрибуты как в объект, и эти атрибуты будут доступны во всех контекстах обработки запроса

```
def get_db():
"""Opens a new database connection if there is none yet for the current application context. """
    if not hasattr(g, 'sqlite_db’):
        g.sqlite_db = connect_db()
    return g.sqlite_db
```

Срок жизни «глобальных» переменных

- g не такая уж глобальная
- Ее срок жизни = сроку жизни контекста приложения, который ~= сроку обработки одного запроса.
- В многопоточных UWSGI серверах своя g на каждую нить

```
@app.teardown_appcontext
def close_db(error):
"""Closes the database again at the end of the request."""
   if hasattr(g, 'sqlite_db’):
      g.sqlite_db.close()
```

Цикл обработки запроса

- before_request() – глобальный хук вызываемый перед всеми запросами
- app.route() – обработчик запроса конкретной URL
- after_request() – может изменить параметры ответа перед его отправкой
- teardown_request() – исполняется всегда, даже если предыдущие функции выкинули исключение
- app.teardown_appcontext()

Шаблоны Jinja2

fat@ptest:**~/flask/templates**$ cat layout.html

```
<!doctype html>
<html><head><title>Наше Flask приложение</title></head>
<body>
<h1>{% block title %}{% endblock %}</h1>
{% for message in get_flashed_messages() %}
<div class=flash>{{ message }}</div>
{% endfor %}
{% block body %}{% endblock %}
</body>
</html>
```

Шаблоны-наследники

fat@ptest:**~/flask/templates**$ cat show_entries.html

```
{% extends "layout.html" %}
{% block title %}Список постов{% endblock %}
{% block body %}
<ul class=entries>
{% for entry in entries %}
<li><a href="{{ url_for('showentry', id=entry) }}">{{ entry }}</a></li>
{% else %}
  <li><em>No entries</em></li>
{% endfor %}
</ul>
{% endblock %}
```

Как это использовать

```
@app.route('/')
def show_entries():
  with dbm.open(app.config['DATABASE'],'r') as db:
    e= (i.decode('utf-8') for i in sorted(db.keys(), key=int))
  return render_template('show_entries.html', entries=e)

@app.route('/showentry/<id>')
def showentry(id):
  with dbm.open(app.config['DATABASE'], 'r') as db:
    e=db[id].decode('utf-8')
  return render_template('entry.html', id=id, body=e)
```