Уеб програмиране с Django

"Програмиране с Python", ФМИ
12.05.2011 г.

Цел за днес

Цел за днес

+ хайку =

(5, 7, 5)

Цел за днес

twitter + хайку =

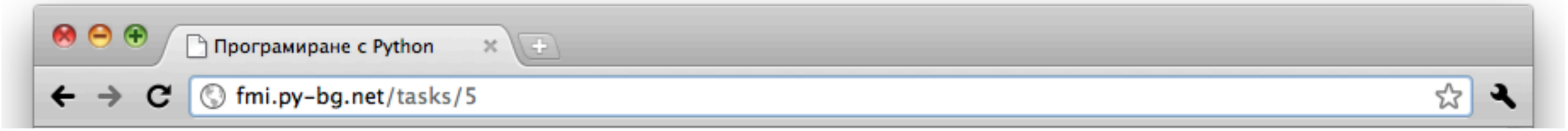
(5, 7, 5)

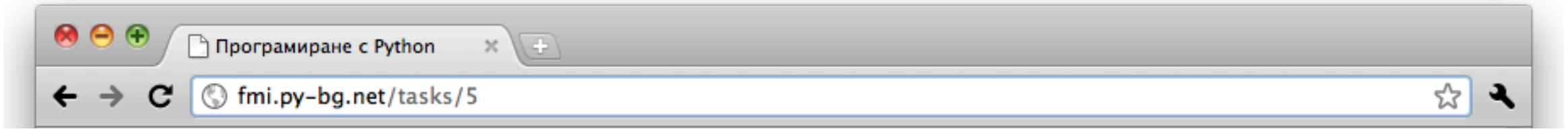
Цел за днес

Полъх на риган.
Бистър пролетен повей.
Бира в “Торонто”.

“Как работи интернеДа”

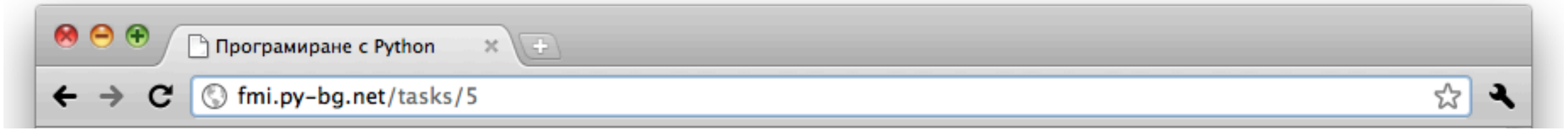
обзор


Програмиране с Python
fmi.py-bg.net/tasks/5

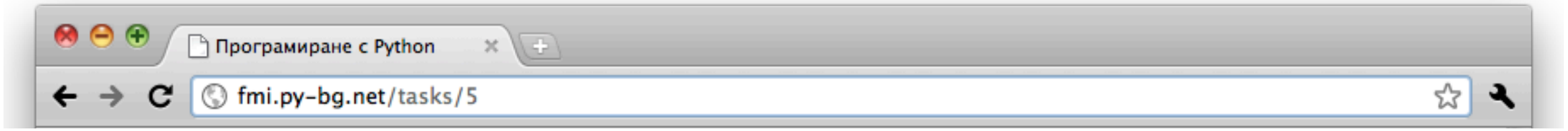


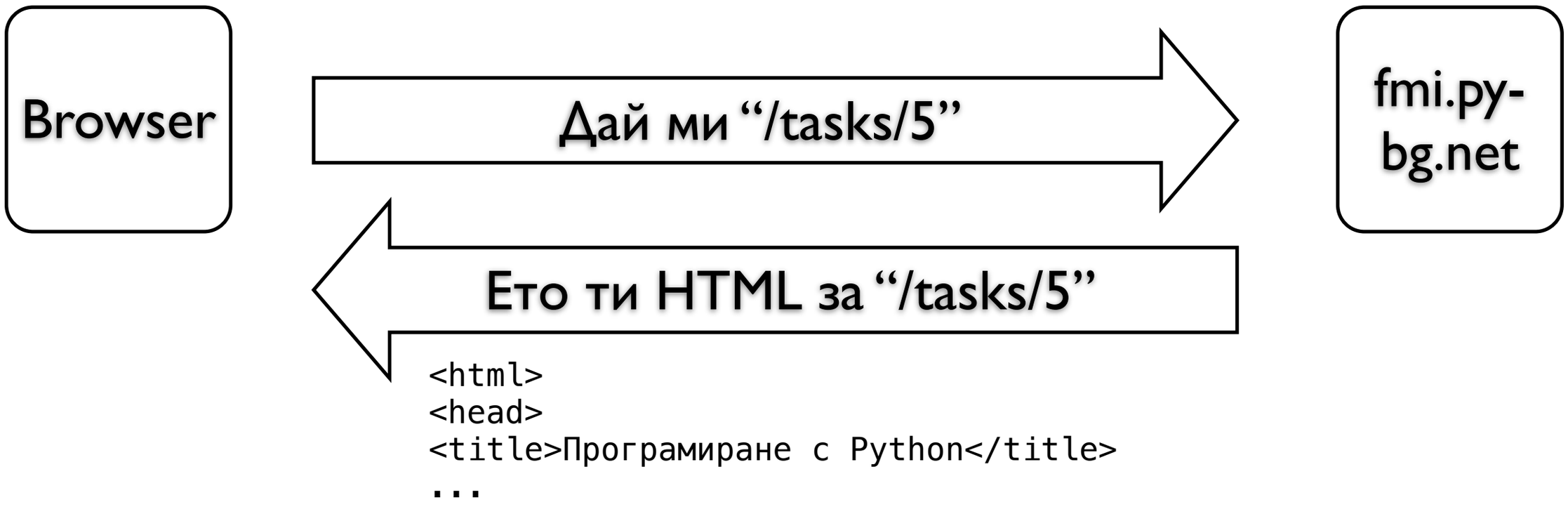
Browser

fmi.py-bg.net

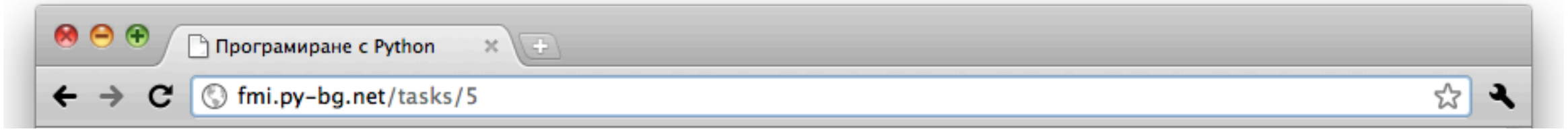

Програмиране с Python
fmi.py-bg.net/tasks/5

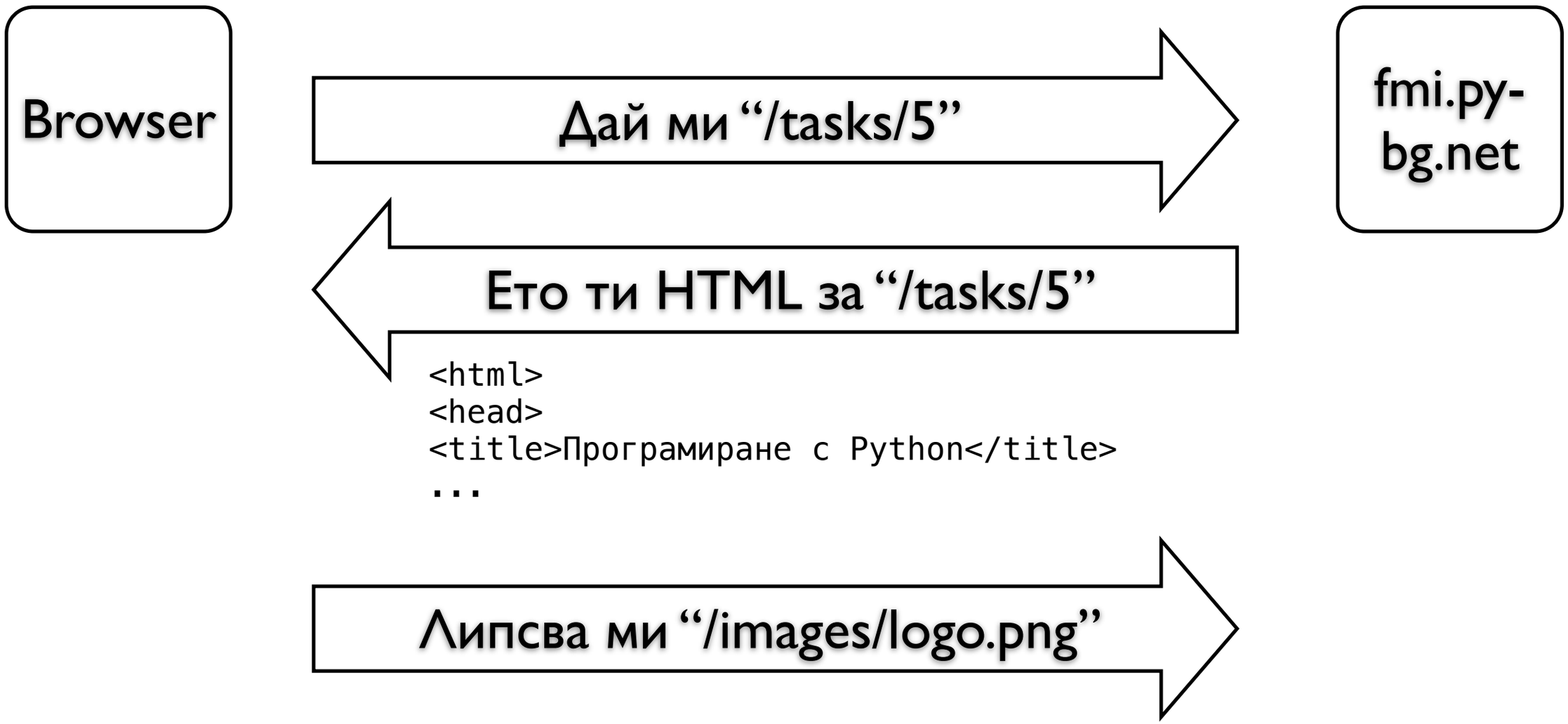

Browser
Дай ми “/tasks/5”
fmi.py-bg.net

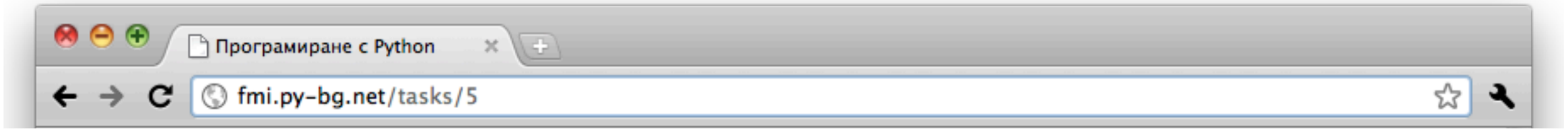


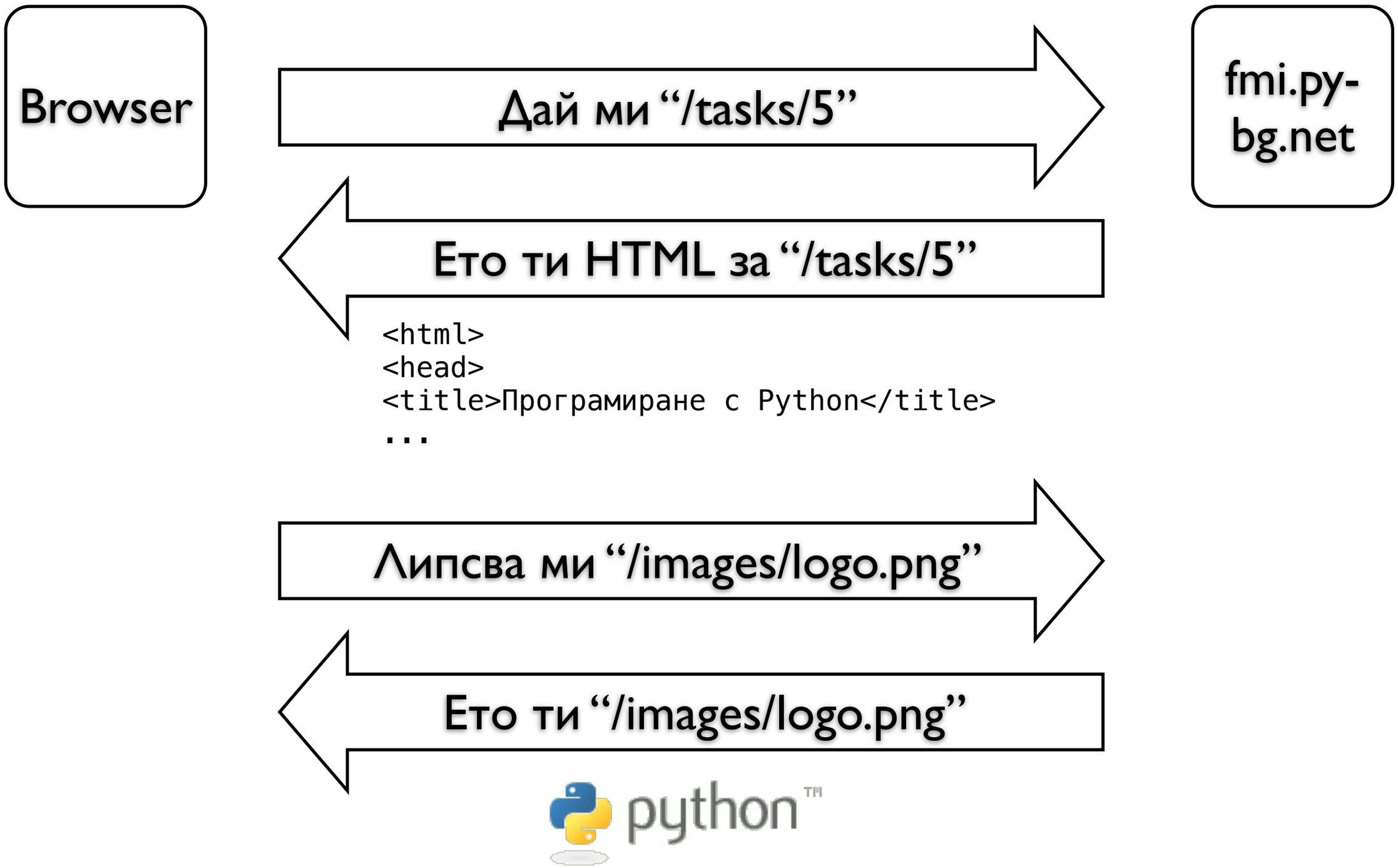



```
<html>
<head>
<title>Програмиране с Python</title>
...
```


Програмиране с Python
fmi.py-bg.net/tasks/5


Browser
Дай ми "/tasks/5"
fmi.py-bg.net
Ето ти HTML за "/tasks/5"
<html>
<head>
<title>Програмиране с Python</title>
...
Липсва ми "/images/logo.png"


Програмиране с Python
fmi.py-bg.net/tasks/5


Browser
Дай ми "/tasks/5"
fmi.py-bg.net
Ето ти HTML за "/tasks/5"
<html>
<head>
<title>Програмиране с Python</title>
...
Липсва ми "/images/logo.png"
Ето ти "/images/logo.png"
python™

python™

Пета задача

Краен срок 12.05.2011 17:00

(списък с редакциите от публикуването до момента - тук)

Лисп е език за програмиране от края на 50'те / началото на 60'те години. Има минималистичен синтаксис и е повлиял много съвременни езици. Освен това има много разновидности и диалекти. Сред тях са Common Lisp, Scheme, Logo, Emacs Lisp и др.

Преди последната версия на Scheme идентификаторите бяха case insensitive; в последната версия са case sensitive. Напишете функциите `tokenize`, `identifiers`, `case_sensitive`; целта на функциите е частично да парснат описаното по-надолу *подмножество* на Scheme и да решат проблема, който преминаването към case sensitive синтаксис създава.

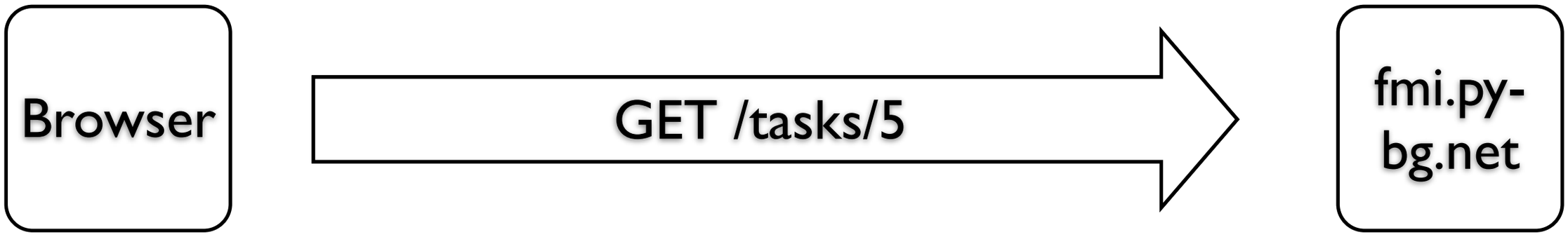
Описание на подмножеството на Scheme:

- Идентификаторите са низове съставени от букви, цифри и специални символи, като започват с буква.
- Специалните символи са: ! $ % & * + - . / : < = > ? @ ^ _ ~
- В допълнение, всеки самостоятелен специален символ също е идентификатор.

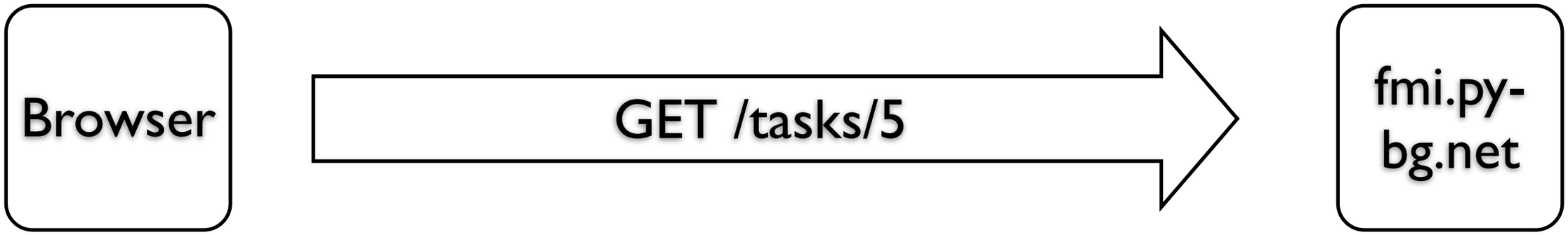
Токен (token) наричаме някое от следните неща:

- идентификатори
- последователности от символи оградени с двойни кавички
- числа (последователности от цифри, като те могат да съдържат най-много

HTTP методи



HTTP методи



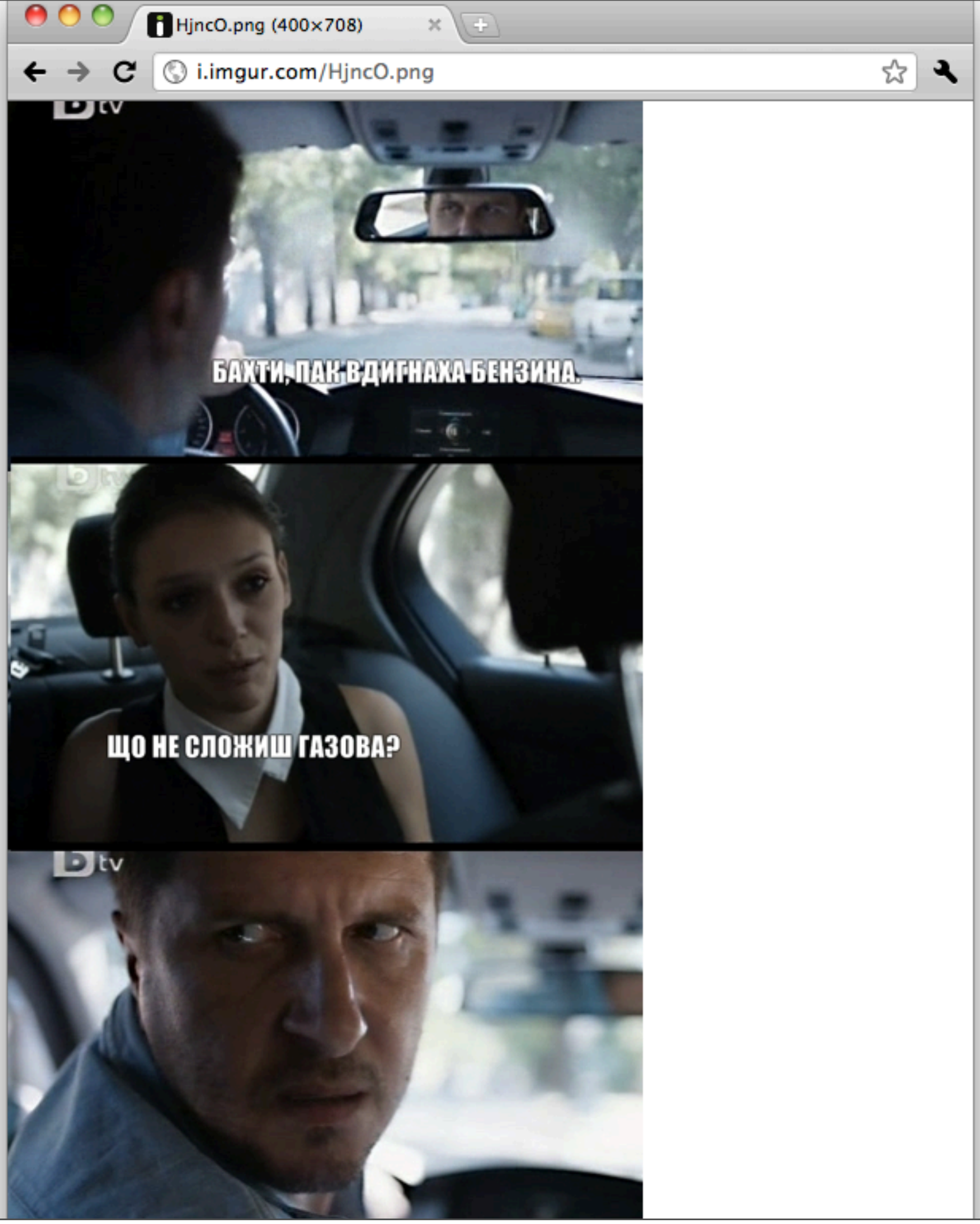

```
GET(url) => html
```

за “дърпане” на данни

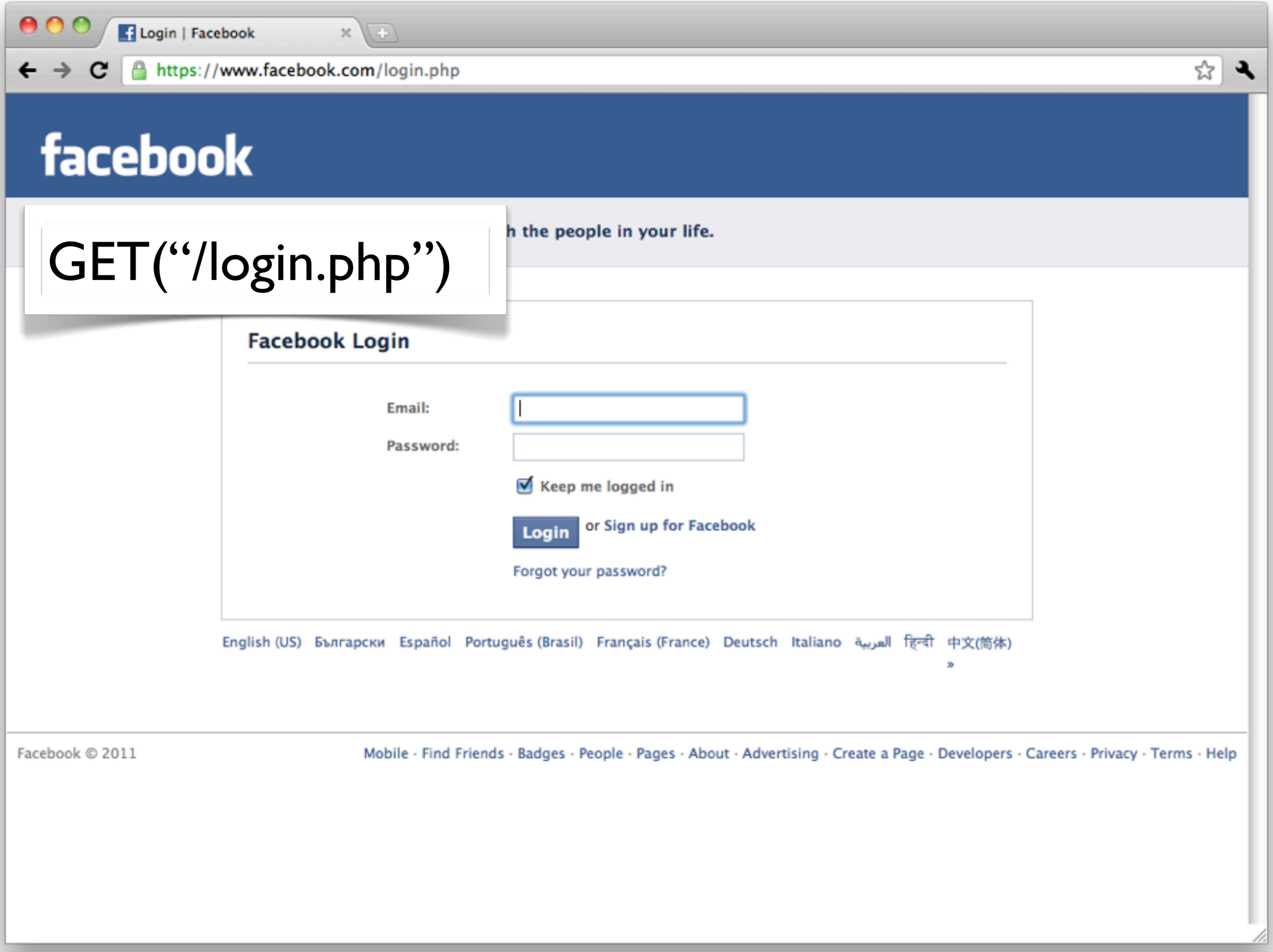
```
POST(url, kwargs) => html
```

за “пращане” на данни

HTTP методи



GET("/HjncO.png")



GET("/login.php")

POST("/login.php",
login = "gotin92@abv.bg",
password = "supertaina")

Login | Facebook

https://www.facebook.com/login.php

r life.

Email: gotin92@abv.bg

Password: •••••

Keep me logged in

Login or Sign up for Facebook

Forgot your password?

English (US) Български Español Português (Brasil) Français (France) Deutsch Italiano العربية हिन्दी 中文(简体)
»

Facebook © 2011

Mobile · Find Friends · Badges · People · Pages · About · Advertising · Create a Page · Developers · Careers · Privacy · Terms · Help

```
<!DOCTYPE html>
<html>
<head>
    <title>Моята първа уеб страница</title>
    <script src="myjavascript.js"></script>
</head>
<body>
    <h1>Добре дошли!</h1>
    <img src="pony.jpg" alt="моята снимка">
    <p>За мен:</p>
    <ul>
        <li>Аз съм пони</li>
        <li>Обичам да паса трева</li>
        <li>Любимият ми филм
          е "Отнесени от вихъра"</li>
    </ul>
</body>
</html>
```

```html
<!DOCTYPE html>
<html>
<head>
    <title>Моята първа уеб страница</title
    <script src="myjavascript.js"></script
</head>
<body>
    <h1>Добре дошли!</h1>
    <img src="pony.jpg" alt="моята снимка"
    <p>За мен:</p>
    <ul>
        <li>Аз съм пони</li>
        <li>Обичам да паса трева</li>
        <li>Любимият ми филм
          е "Отнесени от вихъра"</li>
    </ul>
</body>
</html>
```

самостоятелен таг

```
<img src="pony.jpg" alt="моята снимка">
```

атрибут

атрибут

```
<p>За мен:</p>
```

отварящ таг

затварящ таг

JavaScript

- Интерактивност вътре страниците
- Изпълнява се в браузера
- Може да комуникира със сървера
- Gmail

CSS

- Контролира външния вид на HTML елементите
- Разделение на съдържание и стил

HTML 5!


django

Django: framework за уеб приложения

Прави скучните неща вместо нас.

Кара ни да пишем структуриран код.

Позволява rapid prototyping.

Convention vs. configuration

Ако следваш правилата,
почти всичко работи без конфигурация.

http://docs.djangoproject.com

Python 2.x!

```
$ pip install django
```

или

```
$ easy_install django
```

препоръчваме и
django-annoying, django-extensions и
werkzeug

```
$ django-admin.py startproject myproject
```

или

```
http://github.com/aandr/django-starter
```

```
myproject/
  __init__.py
  settings.py
  local_settings.py
  urls.py
  manage.py
  media
  static
  templates
```

общи настройки

настройки за този компютър

скрипт за системни команди

картинки, JS, CSS

HTML templates

```
$ python manage.py startapp calculator
```

* не бъркайте startapp и startproject

```
myproject/
    calculator/
        __init__.py
        models.py
        tests.py
        views.py
    settings.py
    local_settings.py
    urls.py
    manage.py
    media
    static
    templates
```

Models, Views, Templates (MVT)

Models, Views, Templates (MVT)

Ако сте ползвали MVC:

Models = Models

Views = Controllers

Templates = Views

(Data) Model

View

Template

Дефинира типовете и структурата на данните

Persistence (четене и писане в бази данни)

Примери:
- потребител
- продукт в магазин
- съобщение

(Data) Model

Дефинира типовете и структурата на данните

Persistence (четене и писане в бази данни)

Примери:
- потребител
- продукт в магазин
- съобщение

View

Бизнес логика

Вход/изход между модела и потребителя

Примери:
- изпращане на съобщение
- добавяне на продукт към кошница
- търсене сред потребители

Template

(Data) Model

Дефинира типовете и структурата на данните

Persistence (четене и писане в бази данни)

Примери:
- потребител
- продукт в магазин
- съобщение

View

Бизнес логика

Вход/изход между модела и потребителя

Примери:
- изпращане на съобщение
- добавяне на продукт към кошница
- търсене сред потребители

Template

Представяне на данните

Без логика

HTML

Примери:
- изпращане на съобщение
- добавяне на продукт към кошница
- търсене сред потребители

Моделът помни.

Темплейтът го показва.

А view-то мисли.

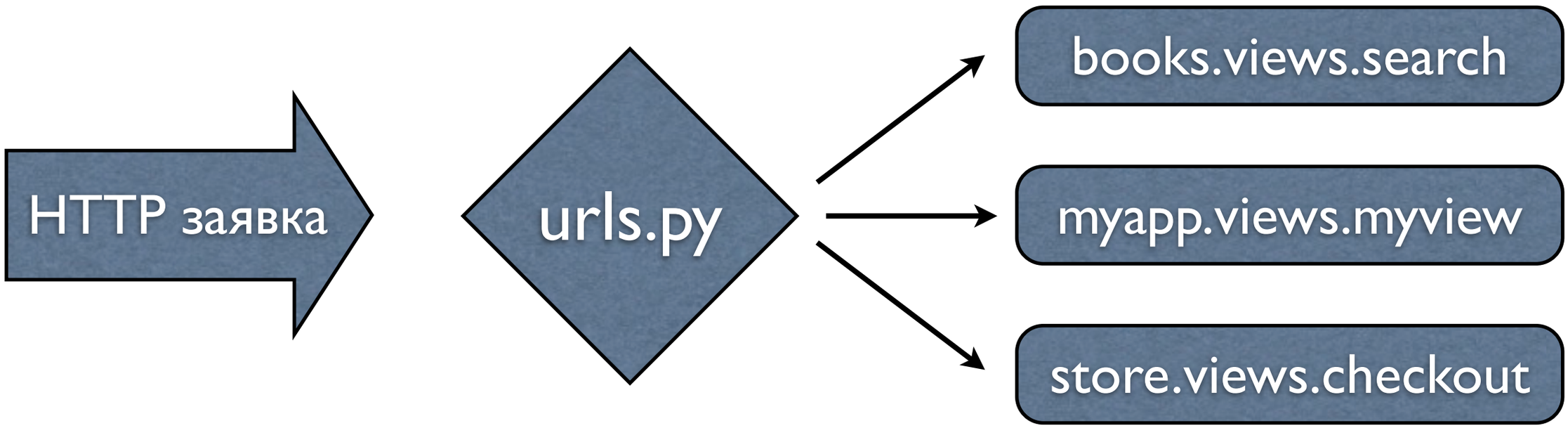

```
def myview(request):
    ...
    return http_response
```

```python
from django.template.response import
                    TemplateResponse

def calculator(request):
    if request.method == 'POST':
        a = int(request.POST['a'])
        b = int(request.POST['b'])
        result = a + b

    return TemplateResponse(request,
        'calculator.html', locals())
```

Повече от едно view?



```
# urls.py
from django.conf.urls.defaults import patterns, include, url

urlpatterns = patterns('',
    url(r'^calculator$', ‘calculator.views.calculator'),
    ...
    )

# /calculator.py => calculator.views.calculator
```

* не слагайте “/” в началото на регулярния израз

```
$ python manage.py runserver
```

Модели

Потребител

- Username
- Име
- Фамилия
- Email
- Парола
- Администатор?
- Дата

Модели

Потребител

- Username
- Име
- Фамилия
- Email
- Парола
- Администатор?
- Дата

Хайку

- Потребител
- Текст
- Дата

```
class User(models.Model):
    username = models.CharField(max_length=30, unique=True)
    first_name = models.CharField(max_length=30, blank=True)
    last_name = models.CharField(max_length=30, blank=True)
    email = models.EmailField(blank=True)
    password = models.CharField(max_length=128)
    is_superuser = models.BooleanField(default=False)
    date_joined = models.DateTimeField(
                        default=datetime.datetime.now)
```

```python
class User(models.Model):
    username = models.CharField(max_length=30, unique=True)
    first_name = models.CharField(max_length=30, blank=True)
    last_name = models.CharField(max_length=30, blank=True)
    email = models.EmailField(blank=True)
    password = models.CharField(max_length=128)
    is_superuser = models.BooleanField(default=False)
    date_joined = models.DateTimeField(
                        default=datetime.datetime.now)
```

* ползвайте на готово:

```python
from django.contrib.auth.models import User
```

```
unique=True # не може да се повтаря

blank=True # не е задължително

default="Иван" # стойност по подразбиране
```

```
models.CharField(max_length = 256)

models.TextField()

models.IntegerField()

models.BooleanField()

models.DateTimeField()

models.FileField()
```

и още 20-тина:
http://docs.djangoproject.com/en/dev/ref/models/fields/

Създаване на запис:

```
spiro = User(username='spiro95',
             first_name='Spiridon')
spiro.lastname = 'Karaivanov'
spiro.save()
```

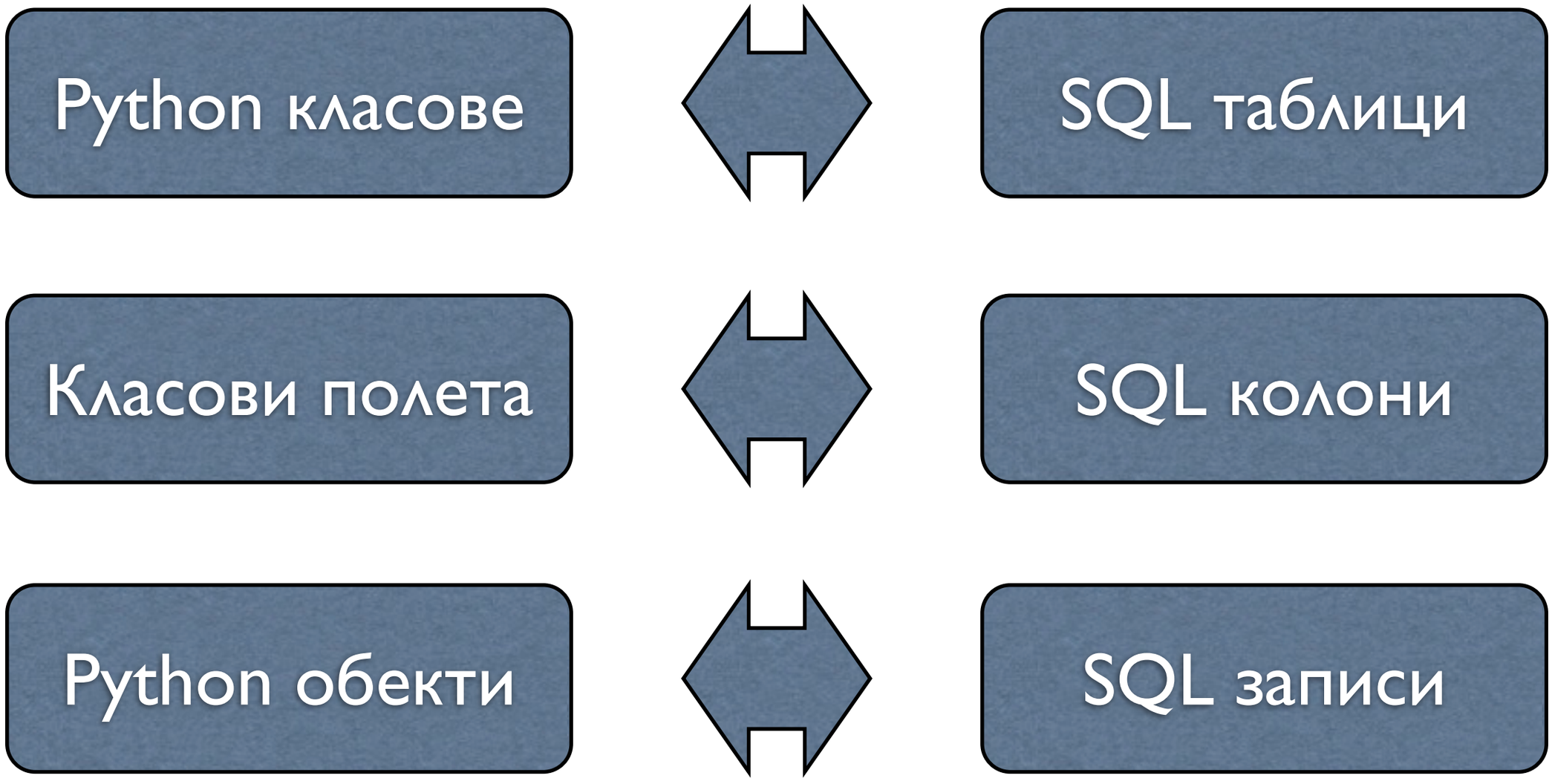
Търсене на много записи:

```
users = User.objects.filter(first_name='Spiridon')
print(users) # 10
print(users[5].lastname) # Stefanov
```

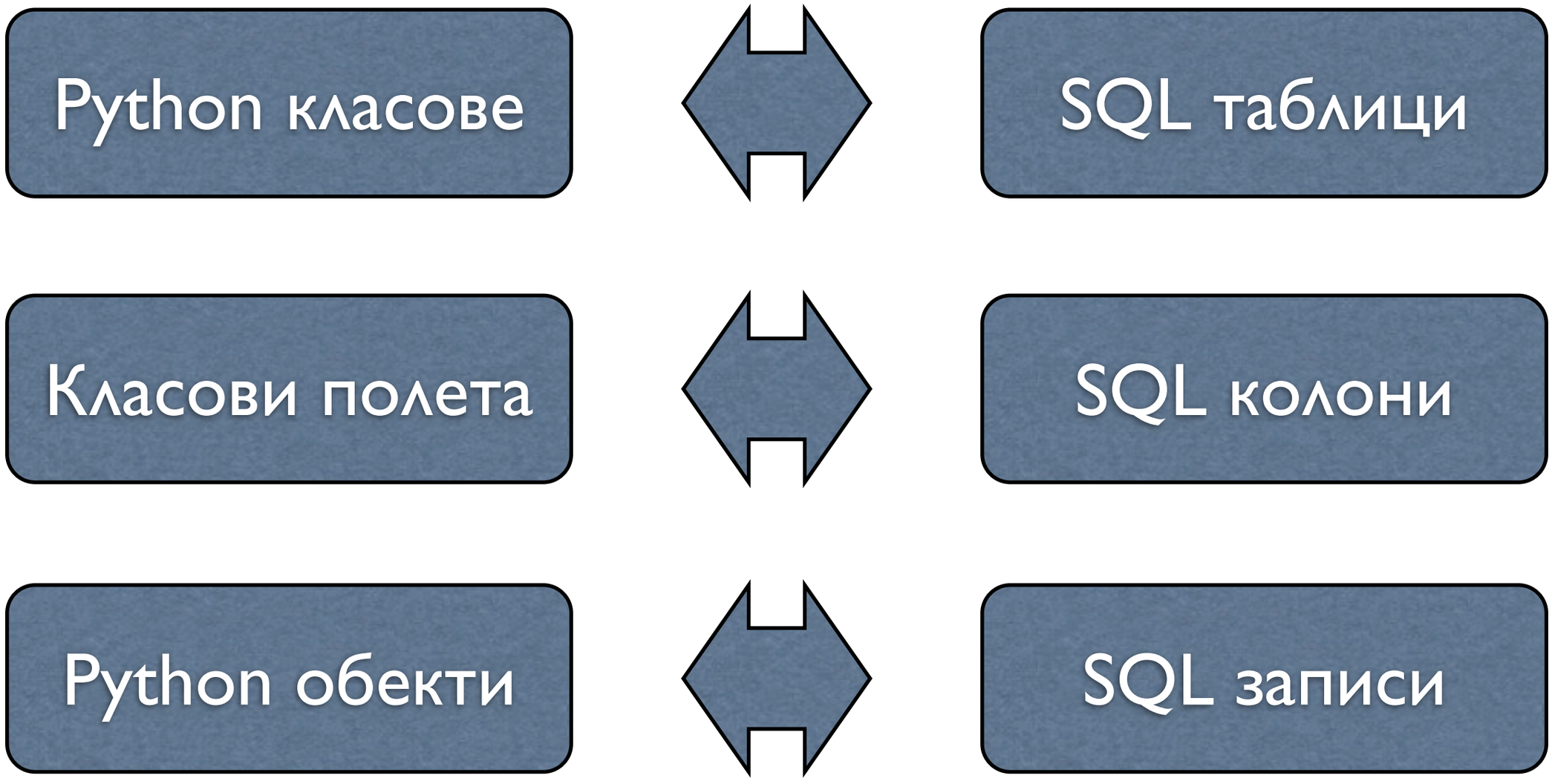
Търсене на един запис и промяна:

```
spiro2 = User.objects.get(username='spiro95')
print(spiro2.lastname) # Karaivanov
spiro2.lastname = 'Grozdev'
spiro2.save()
```

Object-Relational Mapper (ORM)



Object-Relational Mapper (ORM)



Един код за различни бази данни.
SQLite по подразбиране.

уловки?

Релационен модел

- Полетата могат да са само скаларни типове
- За сложни типове, трябва да създадем нов клас + таблица
- Всеки обект има първичен ключ (myobject.id)
- За да реферират друг обект, трябва да ползвате id-то му
- Връзка към друг обект = foreign key
- Връзките са двустранни (haiku.user и user.haiku_set)

```python
# haikus/models.py
from django.db import models
from django.contrib.auth.models import User
class Haiku(models.Model):
    user = models.ForeignKey(User)
    text = models.TextField()
    created = models.DateTimeField(default =
                                   datetime.now)
```

```python
# пример
user = User.objects.get(id = 20)
haiku = Haiku(user = user, text = '...')
haiku.save()
print(haiku.user_id) # 20
print(haiku.user.firstname) # Spiridon
print(user.haiku_set.all()) # [<Haiku: ...>]
```

```python
# haikus/models.py
from django.db import models
from django.contrib.auth.models import User
class Haiku(models.Model):
    user = models.ForeignKey(User)
    text = models.TextField()
    created = models.DateTimeField(default =
                                   datetime.now)

# пример
user = User.
haiku = Haik
haiku.save()
print(haiku.user_id) # 20
print(haiku.user.firstname) # Spiridon
print(user.haiku_set.all()) # [<Haiku: ...>]
```

Django взима user_id, дърпа от таблицата users записа с това id и го връща незабележимо.

```
$ python manage.py syncdb
```

* syncdb не променя вече съществуващи таблици, за това не бързайте да го викате докато не избистрите моделите си; python manage.py sqldiff помага

```
$ python manage.py shell_plus
```

* shell_plus изисква django_extensions; ако нямате, ползвайте shell

Структура на edno57

Начална

Страница на потребител

Ново хайку

Структура на edno57

Начална	Страница на потребител	Ново хайку
Показва последните 50 хайкута	Показва хайкута на потребителя и форма за ново хайку	Проверява хайкуто и го записва

Структура на edno57

Начална	Страница на потребител	Ново хайку
Показва последните 50 хайкута	Показва хайкута на потребителя и форма за ново хайку	Проверява хайкуто и го записва
/	/*username* (напр. /kiril)	/add

Структура на edno57

Начална	Страница на потребител	Ново хайку
Показва последните 50 хайкута	Показва хайкута на потребителя и форма за ново хайку	Проверява хайкуто и го записва
/	/*username* (напр. /kiril)	/add

Login	Logout
/login	/logout

```python
from django.template.response import TemplateResponse
from django.contrib.auth.decorators import login_required
from django.contrib.auth.models import User
from django.shortcuts import *
from haikus.models import Haiku

def homepage(request):
    latest_haikus = Haiku.objects.all().order('-created')[0:50]
    return TemplateResponse(request, "homepage.html", locals())
```

```python
from django.template.response import TemplateResponse
from django.contrib.auth.decorators import login_required
from django.contrib.auth.models import User
from django.shortcuts import *
from haikus.models import Haiku

def homepage(request):
    latest_haikus = Haiku.objects.all().order('-created')[0:50]
    return TemplateResponse(request, "homepage.html", locals())

def user_page(request, username):
    user = get_object_or_404(User, username = username)
    haikus = user.haiku_set.all()
    is_me = request.user == user
    return TemplateResponse(request, "userpage.html", locals())
```

```
@login_required
def add_haiku(request):
    if request.method == "POST":
        haiku = Haiku(user = request.user,
                      text = request.POST.get('text'))
        haiku.save()

    return redirect('user-page', username = request.user.username)
```

Проверява дали потребителят е логнат в системата

```
@login_required
def add_haiku(request):
    if request.method == "POST":
        haiku = Haiku(user = request.user,
                      text = request.POST.get('text'))
        haiku.save()

    return redirect('user-page', username = request.user.username)
```

Templates

- Приемат речник от view-то (наречен context)
- Елементите от речника стават глобални променливи
- Форматират го в текст (най-често HTML)
- Изрази:

```
{{ title }}
{{ user.first_name }}
```

- Елементарна логика:

```
{% if a > 5 %}a е голямо!{% endif %}
{% for el in mylist %}{{ el.name }}{% endfor %}
```

```python
def myview(request):
    facts = ['fact 1',
             'fact 2',
             'fact 3']
    title = "Facts about me"
    return TemplateResponse(request,
             "template.html",
             {'facts': facts,
              'title': title })
```

```
<!DOCTYPE html>
<html>
<head>
    <title>{{ title }}</title>
</head>
<body>
    <h1>{{ title }}</h1>
    <p>За мен:</p>
    <ul>
    {% for fact in facts %}
        <li>{{ title }}</li>
    {% endfor %}
    </ul>
</body>
</html>
```

Топлата вода

```
django.contrib.
    auth # потребители, права, login, logout
    admin # административен панел
    comments # коментари (стил “блог”) с модерация
    flatpages # статични страници (about, FAQ, help)
    gis # GeoDjango - картография, географски примитиви
    messages # нотификации към потребителя
                  (напр. “Операцията е успешна.”)
    syndication # генериране на RSS feeds
    ...
```

Още: http://www.djangopackages.com/

http://github.com/aandr/edno57

http://fmi.py-bg.net/topics/469

Ресурси

- http://docs.djangoproject.com/
- http://docs.djangoproject.com/en/dev/intro/tutorial01/
- HTML & CSS:
  - http://philip.greenspun.com/seia/html
  - http://diveintohtml5.org/
  - http://www.tizag.com/cssT/