evropský sociální fond v ČR
EVROPSKÁ UNIE
MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY
OP Vzdělávání pro konkurenceschopnost
INVESTICE DO ROZVOJE VZDĚLÁVÁNÍ

PSK3-18

Název školy:	Vyšší odborná škola a Střední průmyslová škola, Božetěchova 3
Autor:	Ing. Marek Nožka
Anotace:	web framework Bottle.py
Vzdělávací oblast:	Informační a komunikační technologie
Předmět:	Počítačové sítě a komunikační technika (PSK)
Tematická oblast:	Operační systém Linux/Unix
Výsledky vzdělávání:	Žák vytvoří jednoduchou webovou aplikaci pomocí Bottle.py
Klíčová slova:	Linux, Unix, Apache, Python, Bottle.py, web framework
Druh učebního materiálu:	Online vzdělávací materiál
Typ vzdělávání:	Střední vzdělávání, 4. ročník, technické lyceum
Ověřeno:	VOŠ a SPŠE Olomouc; Třída: 4L
Zdroj:	Vlastní poznámky, Vilém Vychodil: Linux Příručka českého uživatele

Python jako motor webu -- Bottle II

Statické soubory

Webové aplikace napsané v Pythonu mají sklon „vlastnit“ celý adresář (nebo i celou doménu). Přiřazování konkrétních URL určitému kódu je tedy o něco pružnější a je obvykle spravováno tzv. routingem1.

Někdy se ale přece jen hodí mít některé části webu umístěny ve statických souborech. Typickým příkladem můžou být obrázky nebo CSS. V těchto případech použijeme následující kód.

```
from bottle import static_file
@route('/static/<filename>')
def server_static(filename):
    return static_file(filename, root='/path/to/your/static/files')
```

`--> stáhnout

nebo lépe:

```
@route('/static/<filepath:path>')
def server_static(filepath):
    return static_file(filepath, root='/path/to/your/static/files')
```

`--> stáhnout

Funkce `static_file()` je obecně doporučovanou cestou pro předávání statický souborů. Funkce může automaticky odhadnout MIME daného souboru. Ale můžeme to i přímo určit.

```
from bottle import static_file
@route('/images/<filename:re:.*\.png>')
def send_image(filename):
    return static_file(filename, root='/path/to/image/files', mimetype='image/png')

@route('/static/<filename:path>')
def send_static(filename):
    return static_file(filename, root='/path/to/static/files')
```

`--> stáhnout

Prohlížeče většinu souboru jejichž typ znají zobrazí. Je to například PDF nebo JPEG. Dialog pro stažení lze vynutit pomocí parametru `download`. Ten může obsahuje jméno souboru. Pokud se rozhodneme ho nechat stejné jako na serveru stačí hodnotu nastavit na `True`.

```
@route('/download/<filename:path>')
def download(filename):
    return static_file(filename, root='/path/to/static/files', download=filename)
```

`--> stáhnout

Šablony

Vytvářet stránku vždy přímo v obslužné funkci by bylo hodně nepohodlné. Proto Bottle přichází s jednoduchým šablonovacím systémem. Pokud ale tento vestavěný šablonovací systém nedostačuje je možné použít robustnější řešení jako je mako, jinja2 nebo cheetah.

Bottle šablony hledá v adresáři `./views/` (respektive v seznamu `bottle.TEMPLATE_PATH`).

Šablona může vypadat například takto:

```
%if name == 'World':
    <h1>Hello {{name}}!</h1>
    <p>This is a test.</p>
%else:
    <h1>Hello {{name.title()}}!</h1>
    <p>How are you?</p>
%end
```

`--> stáhnout

A následující kód slouží pro použití šablony.

```
@route('/hello')
@route('/hello/<name>')
def hello(name='World'):
    return template('hello_template', name=name)
```

`--> stáhnout

Alternativní způsob rendrování šablony je decorator `@view`. Obslužná funkce potom musí vracet `dict` s proměnnými a jejich hodnotami.

```
@route('/hello')
@route('/hello/<name>')
@view('hello_template')
def hello(name='World'):
    return dict(name=name)
```

`--> stáhnout

Chybové stránky

Pokud se něco pokazí Bottle automaticky zobrazí chybovou stránky. Chybovou stránku si ale můžeme pro konkrétní HTTP status code vytvořit sami.

Například status 404 říká, že dokument nelze nalézt:

```
from bottle import error
@error(404)
def error404(error):
    return 'Nic tu není, jdi pryč'
```

`--> stáhnout

Pokud z nějakého důvodu chceme chybový stav programově vyvolat poslouží nám k tomu funkce `abort()`.

```
from bottle import route, abort
@route('/restricted')
def restricted():
    abort(401, "Promiň, ale sem nesmíš.")
```

`--> stáhnout

Příklad

```
@error(404)
def notFound(error):
    r='<h1>'+error.status+'</h1>'
    r+='<p>Sorry, Tady nic není</p><hr />'
    r+='<p>'+error.body+'</p>'
    return r

@route('/nic')
def nic():
```

```
    abort(404,'bbbbeeeeeeeeee')
```

`--> stáhnout

Vyzkoušejte si rozdíl v chování pro adresu `/nic` a `/nejakabl bost_nmfi oewkwf jwei oq`

Přesměrování

Funkce `redirect()` vyvolá status `303 See Other` a přesměruje požadavek na jinou stránku.

```
from bottle import redirect
@route('/spatna/url')
def wrong():
    redirect("/spravna/url")
```

`--> stáhnout

HTTP hlavička posílaná klientem

Všechny HTTP hlavičky zaslané klientem (např. `Referer`, `Agent` or `Accept-Language`) jsou dostupné ve slovníkovém objektu `request.headers`.

```
@get('/req')
def req():
    r='<pre>'
    for k in request.headers.keys() :
        r += k+':'+request.headers[k]+'\n'
    r+='</pre>'
    return r
```

`--> stáhnout

HTTP hlavička odpovědi a výchozí kódování

Pokud chceme zasáhnou do HTTP hlavičky, kterou naše aplikace odesílá, použijeme objekt `Response`.

```
@route('/wiki/<page>')
def wiki(page):
    response.set_header('Content-Language', 'en')
    ...
```

`--> stáhnout

Pokud má hlavička obsahovat více stejnojmenných polí použijeme kromě metody `.set_header()` také metodu `add_header.()`.

```
....
response.set_header('Set-Cookie', 'name=value')
response.add_header('Set-Cookie', 'name2=value2')
....
```

`--> stáhnout

Výchozí kódování

Změna výchozího kódování se děje pomocí pole Content-Type. Výchozí hodnotu:

```
Content-Type: text/html; charset=UTF-8
```

... je možné změnit pomocí `Response.content_type` nebo přímo pomocí `Response.charset`:

```
from bottle import response
@route('/iso')
def get_iso():
    response.charset = 'ISO-8859-15'
    return u'This will be sent with ISO-8859-15 encoding.'

@route('/latin9')
def get_latin():
    response.content_type = 'text/html; charset=latin9'
    return u'ISO-8859-15 is also known as latin9.'
```

`--> stáhnout

WSGI/CGI prostředí

Některé informace se WSGI/CGI programu předávají pomocí proměnných prostředí. Je to například `REQUEST_METHOD` nebo `REMOTE_ADDR`.

```
@route('/my_ip')
def show_ip():
    ip = request.environ.get('REMOTE_ADDR')
    # or ip = request.get('REMOTE_ADDR')
    # or ip = request['REMOTE_ADDR']
    return template("Your IP is: {{ip}}", ip=ip)
```

`--> stáhnout

nebo

```
@get('/env')
def env():
    r='<pre>'
    for k in request.environ.keys() :
        r += '<strong>'+k+'</strong>:'+str(request.environ[k])+'\n\n'
    r+='</pre>'
    return r
```

`--> stáhnout

Cookies

Cookie je pojmenovaný kus textu uložené ve webovém prohlížeče. Cookie si můžeme vyžádat přes `request.get_cookie()` a nastavit

nové cookies pomocí `response.set_cookie()`:

```
@route('/hello')
def hello_again():
    if request.get_cookie("visited"):
        return "Vítej zpět! Je jezké, že jsi zase přišel"
    else:
        response.set_cookie("visited", "yes")
        return "Nazdar! Vítam tě tu."
```

`--> stáhnout

Metoda `response.set_cookie()` přijímá řadu dalších pojmenovaných argumentů, které řídí život a chování souborů cookie. Některé z nejběžnějších nastavení jsou popsány v následující tabulce:

`max_age`	Platnost cookie v sekundách. (default: None)
`expires`	Expirace: `datetime` objekt nebo UNIX timestamp. (default: None)
`domain`	Doména pro kterou cookie platí (default: aktuální doména).
`path`	Limituje cookie pro určitou cestu (default: `/`).
`secure`	Limituje cookie pro `HTTPS` spojení (default: off).
`httponly`	Zabrání Jabascriptu na straně klienta ve čtení cookie (default: off).

Pokud není nastaven `expires` ani `max_age`, cookie vyprší na konci relace prohlížeče.

Dále byste měli vzít v úvahu:

- Cookies jsou omezeny na 4 KB textu (většinou).
- Někteří uživatelé nastavují své prohlížeče, aby nepoužívali cookies. Ujistěte se, že vaše aplikace stále funguje i bez cookies.
- Cookies jsou uloženy na straně klienta a nejsou žádným způsobem šifrována. Soubory cookie, může uživatel číst. Některé viry jsou známé čtením cookies. Proto nikdy ukládejte důvěrné informace do cookies.

Nevěřte cookies!

Další odkazy na dokumentaci

- Dynamic Routes
- HTTP Request Methods
- HTML <form> Handling
- Routing Static Files
- Request Routing
- Templates
- Error Pages
- Generating content
- Static Files
- HTTP Errors and Redirects
- The Response Object
- Cookies

- Request Data
- File uploads
- JSON Content
- Plugins
- Development

---

1. Více o routingu se dozvíte v oficiální dokumentaci na http://bottlepy.org/docs/dev/tutorial.html#request-routing a http://bottlepy.org/docs/dev/routing.html ↩