Flask micro-framework

محمدرضا کمالی‌فرد

پنجمین همایش سراسری نرم افزارهای آزاد/متن باز
زنجان - ایران

شهریور ۹۳

چکیده

Flask یک میکرو microframework وب پایتون است. که از Werkzeug و Jinja 2 استفاده می‌کند.
هسته ساده و سبکی دارد و برای توسعه ساخته شده است. امکانات پایه مورد نیاز یک framework مانند Router و Request ، Response و Debug را پیاده کرده و به طور پیش فرض از Jinja 2 برای Template Engine انتخاب کرده اما استفاده از هیچ چیز به همراه فلسک اجباری نیست و می‌توان فلسک را با استفاده از افزونه ها مختلف توسعه داد و امکنات را به آن اضافه کرد.
با استفاده از فلسک با استفاده از فلسک توسعه دهنده امکان این را دارد که به سادگی از تمام سیستم های پیگاه داده موجود (SQL, NOSQL)هر نوع سیستم و روش authentication ای که نیاز دارد استفاده کند چیزی بیشتر از نیازش را وارد فریم ورک نکند.
در واقع کار با فلسک مانند درست کردن لگو است از ترکیب بخش مختلف برنامه مورد نظر ساخته می‌شود و فریم ورک به شکل مورد نیاز برنامه ما تغییر می‌کند که این abstraction در فلسک بسیار دلنشین و پایتونیک هست.
فلسک هنوز به نسخه ۱ نرسیده اما شکل گسترده در محصولات واقعی و شرکت ها و برنامه های کاربردی استفاده می‌شود و این مقاله با توجه به لزوم معرفی فلسک به مخاطبان زیکانف نوشته شده است.

کلمات کلیدی

فلسک،microframework ،Flask، برنامه نویسی، وب، Web application، برنامه نویسی وب، پایگاه داده، Database

فریم ورک وب(Web Framework)

فریم ورک وب مجموعه فریم ورکی هست که برای کمک به ساختن وب‌سایت های پویا، برنامه های تحت وب، سرویس های وب استفاده می‌شود. هدف از استفاده از این فریم ورک ها این است که کارهای تکراری مربوط به توسعه نرم‌افزار وب توسط این فریم ورک ها انجام شود و به سرعت و کیفیت نرم‌افزار افزوده شود.

وب فریم ورک ها به طور معمول امکاناتی زیر را ارائه می‌دهند:

- الگوی طراحی (Design Pattern)
- URL Routing:

  امکاناتی برای ارتباط دادن Request های رسیده از کاربر به بخشی از کد که باید برای آن صفحه اجرا شود.
- Request and Response Objects:

  بسته بندی و رساندن اطلاعات بین درخواست های کاربر و مرورگر کاربر
- Template Engine:

  امکان جدا سازی کد ها و Business برنامه وب از HTML هایی که باید به مرورگر کاربر ارسال شود.
- Database Object Relational Mapper :

  امکاناتی برای دسترسی آسان به دیتابیس های Relational به صورت Object هایی در برنامه و تبدیل مجدد داده ها و ذخیره سازی اطلاعات در دیتابیس اصلی رابطه ای.
- Development Web Server:

  ایجاد یک HTTP سرور روی ماشین ای که توسعه نرم‌افزار روی آن انجام می‌شود برای بالا بردن سرعت دیدن نتیجه کار و توسعه سریع‌تر نرم‌افزار.

فلسک (Flask)

فلسک یک میکرو فریم ورک وب متن‌باز است. که از Werkzeug به عنوان ابزار WSGI و Jinja 2 به عنوان Template Engine استفاده می‌کند.

میکرو فریم ورک بودن فلسک به این دلیل نیست که فلسک برخی از قابلیت های فریم ورک های وب را ندارد فلسک میکرو فریم ورک است به این خاطر که هسته بسیار ساده ای دارد و بسیاری از بخش ها یک فریم ورک مثل جنگو داخل آن پیاده سازی نشده است ولی این هسته ساده قابلیت گسترش دارد.

در فلسک سیستم دیتابیس ORM از ابتدا وجود ندارد، راهکاری از ابتدا برای پردازش و اعتبارسنجی فرم ها وجود ندارد، فلسک برای توسعه دهندگان نرم‌افزار از پیش انتخابی نکرده و دست آن‌ها را باز گذاشته با راحتی هر افزونه ای که می‌خواهند را به فلسک اضافه کنند و از آن ها استفاده کنند.

فلسک با استفاده از کد های Flexible پایتون یک ساختار ساده و قابل گسترش برای توسعه برنامه های تحت وب ایجاد می‌کند.

فلسک فریم ورک ایده آل برای توسعه سریع نرم‌افزار های تحت وب و سرویس های وب است و سرعت توسعه نرم‌افزار را بسیار بالا می‌برد.

یک برنامه ساده با فلسک:

```
from flask import Flask
app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello World!"

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

این برنامه ساده ترین برنامه با فلسک است که در یک فایل نوشته می‌شود.

اگر این کد را با پایتون اجرا کنیم سرور توسعه‌ای فلسک روی http://localhost:5000 اجرا می‌شود و به درخواست هایی که به / بروند مقدار `Hello World` را بر می‌گرداند.

بررسی Flask

Flask Router:

فلسک از decorator برای تعیین روتر استفاده می‌کند.
برنامه های وب امروزی از URL های خوب و زیبا استفاده می‌کنند و به هر چیزی در برنامه می‌توان با استفاده از آدرس آن دسترسی داشت.
با استفاده از فلسک بسیار راحت می‌توان آدرس هایی که می‌خوایم رو بسازیم و این کار برای انسان هم قابل خواندن و بسیار ساده است

```
@app.route('/')
def index():
    return 'Index Page'

@app.route('/hello')
def hello():
return 'Hello World'
```

با استفاده از این روتر ها می‌توانیم بخشی از کد را که باید با انتخاب این آدرس توسط کاربر اجرا شود را مشخص کنیم
آدرس های پیچیده تری هم می‌توان استفاده کرد.
می‌توان یک متغییر را به آدرس URL اضافه کرد و به صورت <variable_name> علامت گذاری کرد و این متغییر به پس از ورود آن بخش از آدرس توسط کاربر به Function مربوط به این Router انتقال پیدا می‌کند. برای این متغییر ها می‌توان نوع هم معین کرد به عنوان مثال int یه متغییر به صورت عددی است.

```
@app.route(’/user/<username>’)
def show_user_profile(username):
    # show the user profile for that user
    return ’User %s’ % username
@app.route(’/post/<int:post_id>’)
def show_post(post_id):
    # show the post with the given id, the id is an integer
    return ’Post %d’ % post_id
```

با استفاده از route آدرس ها رو به view function ها ارتباط دادیم و عکس این عمل هم در فلسک به آسانی قابل انجام است
با استفاده از url_for می‌توان به آدرس مربوط به یک view function دسترسی داشت.

```
>>> from flask import Flask, url_for
>>> app = Flask(__name__)
>>> @app.route('/')
... def index(): pass
...
>>> @app.route('/login')
... def login(): pass
...
>>> @app.route('/user/<username>')
... def profile(username): pass
...
>>> with app.test_request_context():
... print url_for('index')
... print url_for('login')
... print url_for('login', next='/')
... print url_for('profile', username='itmard')
...
/
/login
/login?next=/
/user/itmard
```

می‌توان method را هم در زمان تعریف route تعیین کرد

```
from flask import request

@app.route(’/login’, methods=[’GET’, ’POST’])
def login():
    if request.method == ’POST’:
        do_the_login()
    else:
        show_the_login_form()
```

Request Object:

برنامه وب باید به اطلاعات فرستاده شده از سوی کاربر (Request Data) دسترسی داشته باشد. این کار در فلسک با استفاده از request انجام می‌شود که به صورت global تعریف شده است و برای thread safe بودن از context manager استفاده می‌شود و در طول پردازش یک request این نام به Object مورد پردازش اشاره می‌کند.

```
@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)
```

Template Engine:

ایجاد HTML داخل کد های پایتون به دلایل امنیتی و جدا سازی کد و Business برنامه از اطلاعات ثابت مناسب نیست. به این دلیل HTML را به وسیله Template Engine ایجاد می‌کنیم.فلسک از Jinja2 به عنوان Template Engine استفاده می‌کند. برای پردازش Template باید از render_template استفاده کنیم.

```
from flask import render_template
@app.route('/hello/')
@app.route('/hello/<name>')
def hello(name=None):
    return render_template('hello.html', name=name)
```

فلسک به دنبال hello.html در فولدر templates می‌گردد.

```
App as a module:
/application.py
/templates
    /hello.html
App as a package:
/application
    /__init__.py
    /templates
        /hello.html
```

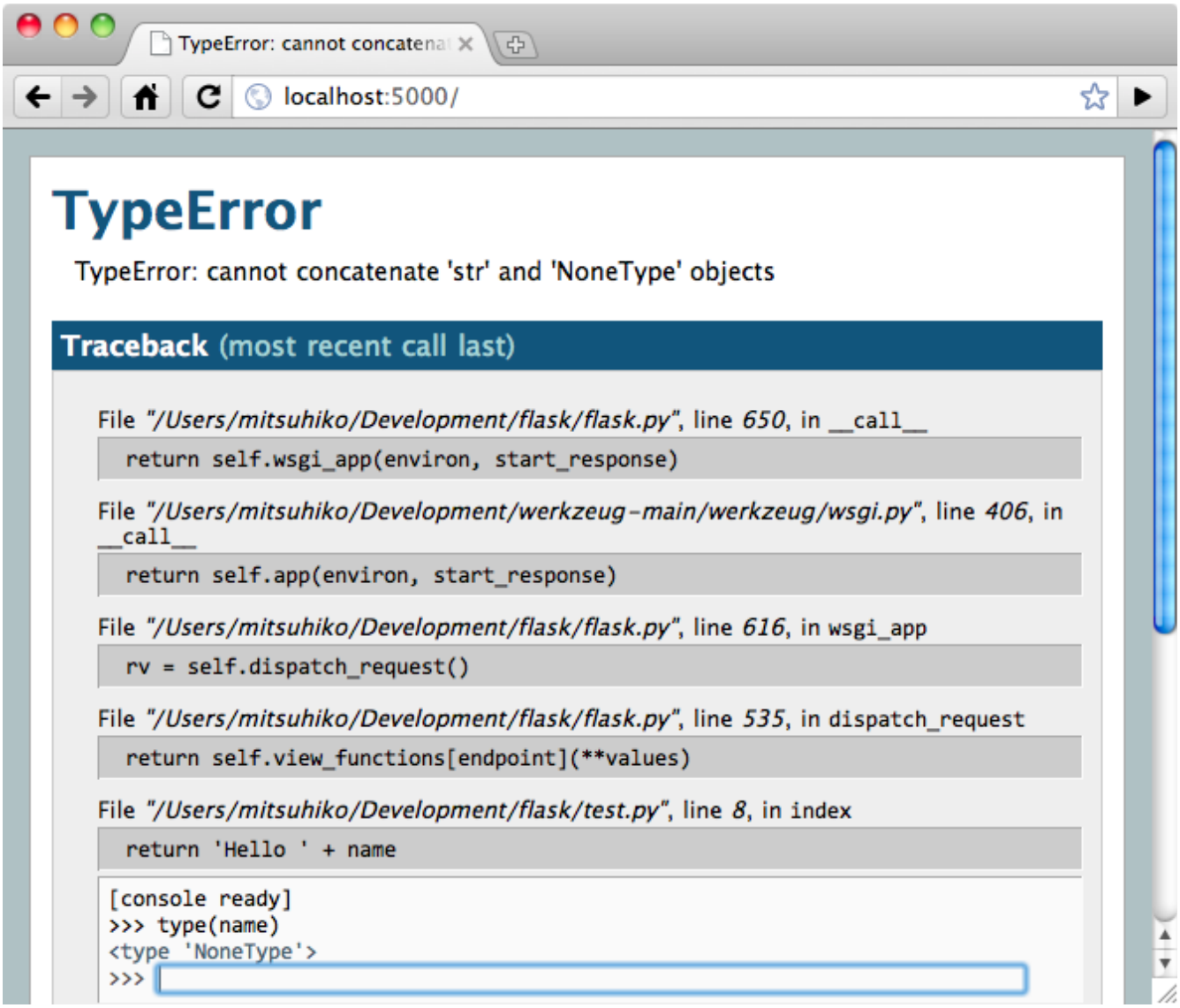
داخل فایل hello.html به جای html معمولی از فرمت jinja استفاده می‌شود

```
<!doctype html>
<title>Hello from Flask</title>
{% if name %}
    <h1>Hello {{ name }}!</h1>
{% else %}
    <h1>Hello World!</h1>
{% endif %}
```

داخل template می‌توانیم از if و else استفاده کنیم و به متغیر هایی که به render_template ارسال شده هم دسترسی داریم مانند main.

Development Server:

یک سرور محلی که بتوانیم برنامه را روی آن در هنگام توسعه ببینیم برای این کار در فلسک اگر از run استفاده کنیم یک سرور لوکال اجرا می‌شود و به صورت local در دسترس است اما با ایجاد هر تغییر روی فایل های برنامه باید آن را به صورت دستی Restart کنیم برای حل این مشکل امکاناتی برای Debug کردن برنامه وجود دارد می‌توان Debug Mode را فعال کرد تا با تغییر فایل ها سرور به صورت خودکار Restart شود همچنین Error ها به صورت کامل و زیبا با TraceBack به نمایش در بیایند و بتوان در داخل پنجره Error کد هایی را اجرا کرد و به Debug برنامه پرداخت.
برای فعال کردن Debug mode باید از app.debug = True باشد.



در console ای که بالا ارائه شده است می‌توانیم در فایل مروبطه که در بخش Trace مشخص شده است دستوراتی را وارد کرده و جواب آن ها را به صورت Interactive ببینم که این قابلیت Debug برنامه را بسیار سریع و ساده می‌کند.

ORM در فلسک:

فلسک از پیش هیچ سیستم برای ساپورت از دیتابیس سیستم ها ندارد اما به سادگی با استفاده از افزونه می‌توان قابلیت استفاده از انواع دیتابیس ها را به آن افزود به عنوان مثال با استفاده از Flask-SQLAlchemy می‌توان از ORM بسیار کارا و توانا SQLALCHEMY در فلسک استفاده کرد.

اضافه کردن این افزونه ها به فلسک بسیار ساده است

```
from flask import Flask
from flask.ext.sqlalchemy import SQLAlchemy

app = Flask(__name__)
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:////tmp/test.db'
db = SQLAlchemy(app)


class User(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    username = db.Column(db.String(80), unique=True)
    email = db.Column(db.String(120), unique=True)

    def __init__(self, username, email):
        self.username = username
        self.email = email

    def __repr__(self):
        return '<User %r>' % self.username
```

نمونه بالا یک app فلسک است که از SQLALCHEMY استفاده می‌کند. و به سادگی می‌توان در آن جداول دیتابس را به صورت کلاس تعریف کرده و Entity ها را به صورت متغییر های کلاس تعریف کرد. این قابلیت امکان مناسبی برای بهره گیری از قابلیت شی گرایی پایتون در استفاده از دیتابیس های رابطه را به کاربر می‌دهد.
به سادگی می‌توان انواع روابط را در این سیستم پیاده سازی کرد و همچنین جداول دیتابیس را در فایل های مختلف تعریف کرد.همچنین علاوه بر SQLALCHEMY گزینه های دیگری مانند pewee نیز وجود دارند که کاملا قابل استفاده و افزوده شدن به فلسک هستند.

استفاده از دیتابس های NOSQL نیز در فلسک بسیار ساده است. با اضافه کردن Flask-MongoEngine به سادگی می‌توان از دیتابیس Mongo-DB استفاده کرد و نیاز به تغییرات زیادی نسبت به دیتابیس رابطه ای وجود ندارد.
برای اضافه کرد این قابلیت به فلسک کافی است از این کد استفاده کنیم:

```
from flask import Flask
from flask.ext.mongoengine import MongoEngine

app = Flask(__name__)
app.config.from_pyfile('the-config.cfg')
db = MongoEngine(app)
```

مدیریت فرم ها در فلسک :

برای مدیریت فرم ها می‌توانیم از Flask-WTF به همراه فلسک استفاده کنیم.
با استفاده از این افزونه فلسک قابلیت های بسیار امنیتی و کنترلی رو فرم ها به فلسک اضافه می‌شود و به سادگی می‌توانیم فرم های ایجاد شده با WTF را در view های فلسک پردازش کنیم و به نمایش در بیاوریم.
یک فرم ساده در WTF فلسک:

```
from flask_wtf import Form
from wtforms import StringField
from wtforms.validators import DataRequired

class MyForm(Form):
    name = StringField('name', validators=[DataRequired()])
```

Validate کردن فرم:

```
@app.route('/submit', methods=('GET', 'POST'))
def submit():
    form = MyForm()
    if form.validate_on_submit():
        return redirect('/success')
    return render_template('submit.html', form=form)
```

انجام بسیاری از کارها و همچنین اعتبار سنجی داده های ورودی کاربر در این سیستم بسیار ساده است و حتی کد های سمت مرورگر کاربر نیز به طور خودکار به view مورد نظر افزوده می‌شوند همچنین امکاناتی مانند استفاده بسیار ساده از Recaptcha وجود دارد که به سادگی می‌توان علاوه بر استفاده از captcha اعتبار سنجی آن نیز به صورت خودکار صورت گیرد.

```
from flask_wtf import Form, RecaptchaField
from wtforms import TextField

class SignupForm(Form):
    username = TextField('Username')
    recaptcha = RecaptchaField()
```

همچنین برای آپلود کردن فایل به صورت امن:

```
from werkzeug import secure_filename
from flask_wtf.file import FileField

class PhotoForm(Form):
    photo = FileField('Your photo')

@app.route('/upload/', methods=('GET', 'POST'))
def upload():
    form = PhotoForm()
    if form.validate_on_submit():
        filename = secure_filename(form.photo.data.filename)
        form.photo.data.save('uploads/' + filename)
    else:
        filename = None
    return render_template('upload.html', form=form, filename=filename)
```

همچنین در بخش HTML هم تنها کافی است این تغییرات داده شود

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

Design Patterns:

همان طور که در بخش اول اشاره شد برنامه فلسک می‌تواند به طور Single file application و تنها در یک فایل عرضه شود اما به این معنا نیست که ساختار در آن وجود ندارد.
فلسک به طور کامل از MVC و *MV پشتیبانی می‌کند و به سادگی می‌توان تمامی ساختار هایی که مورد نظر برنامه نویس باشد را در آن پیاده سازی کرد در این فریم ورک هیچ ساختاری به برنامه نویس تحمیل نمی‌شود.
چند نمونه از ساختار هایی که برای استفاده پیشنهاد می‌شوند در ادامه معرفی شده اند.

project

–--apps

----------app1

----------models.py

----------views.py

----------forms.py

----------apis.py

–----statics

–----templates

app

-----models

-----views

-----forms

-----statics

-----templates

افزونه های Flask:

فلسک افزونه های بسیاری دارد و هر روز به این تعداد افزوده می‌شود. این افزونه های کارهای تکراری و معمول در برنامه نویس برای وب مانند انجام عملیات Authentication و Login و logout کاربر رهکارهایی برای ترجمه و بومی سازی نرم‌افزار به زبان های مختلف، ارسال و دریافت ایمیل و... را انجام می‌دهند. به آسانی می‌توان با افزودن این افزونه ها به برنامه فلسک از آن ها استفاده کرد.

لیست کاملی از این افزونه ها در زیر در دسترس است.

http://flask.pocoo.org/extensions/

فلسک به عنوان یک Framework وب بسیار سبک و کارها مطرح است و با توجه به چابک و سریع بودن و قابلیت بالا در Prototyping خصوصا برای تهیه وب‌سرویس ها و API بسیار مناسب است.
با اندکی تغییرات می‌توان برنامه های فلسک را به آماده انجام کارهای بسیاری کرد و نیاز نیست کد هایی که وجود دارد دوباره نوشته شود فلسک در ساختار خود نیز همین گونه است و به طور مثال از WSGI tool موجود که بسیار غنی است یعنی Werkzeug استفاده می‌کند.
استفاده از فلسک ساده و سریع است و به خوبی در حد اندازه های برنامه واقعی وب (Web Scale) عمل می‌کند.

Introduction to Flask

Mohammad Reza Kamalifard

The 5th free and opensourse software conference

September 2014

ZANJAN, IRAN