The Key To Grasp Both Flask And Django in One Session

Python Mauritius UserGroup (pymug)

More info: mscc.mu/python-mauritius-usergroup-pymug/

Where Are We? 🏡

Why	Where
codes	github.com/pymug
share events	twitter.com/pymugdotcom
ping professionals	linkedin.com/company/pymug
all info	pymug.com
discuss	facebook.com/groups/318161658897893
tell friends by like	facebook.com/pymug

Updates

Our mailing list:

https://mail.python.org/mailman3/lists/pymug.python.org/

Abdur-Rahmaan Janhangeer
(Just a Python programmer)

twitter: @osdotsystem
github: github.com/abdur-rahmaanj

www.pythonmembers.club

The Key To Flask and Django

Flask and Django are web frameworks.

Flask is a bare-bones framework

Django has the basics needed for a nice programming experience

⚠️ The explanations aim for understanding and do not necessarily cover all corner cases

Flask example taken: Shopyo (https://github.com/Abdur-rahmaanJ/shopyo). Shopyo internals explained here:

- https://dev.to/abdurrahmaanj/shopyo-enhance-your-flask-by-exploring-an-advanced-flask-app-40j3 or
- https://www.pythonmembers.club/2020/02/25/shopyo-enhance-your-flask-by-exploring-an-advanced-flask-app/

Django example taken: Django Girls blog app created in the tutorial (https://github.com/Abdur-rahmaanJ/DjangoGirlsBlog) with tuto at

- https://tutorial.djangogirls.org/en/

Added both for convenience at

https://github.com/pymug/ARJ_KeyToDjangoAndFlask_Feb2020

Flask Batteries That You Can Use to Mimick Django

- 🔋 flask_sqlalchemy: ORM
- 🔋 flask_marshmallow: REST Api
- 🔋 marshmallow_sqlalchemy
- 🔋 flask_wtf: CSRF protection and forms
- 🔋 flask_script: to have a nice manage.py
- 🔋 flask_migrate: migrations
- 🔋 flask_login: auth & permission levels
- 🔋 flask_httpauth: raw api auth

That should pretty much help have the same base as Django

The same structure as Django

You can use the exact same structure of Django in Flask using the **Divisional pattern** illustrated below.

📁 module / app

- 📄 models.py
- 📄 views.py
- 📄 forms.py
- 📁 templates

The MVT pattern

Both Flask and Django follow the MVT pattern

Model - View - Template

🎁 Bonus

💡 There are also some who hold the view that Django is MVC, where the controller is the Django app itself

💡 Flask was not inspired from Django but it's templating language was

💡 Django introduced routes like Flask, which is the now the preferred option. Mozilla's Django tuto (https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Generic_views) tells

```
re_path(r'^book/(?P<pk>\d+)$', views.BookDetailView.as_view(),
```

> Regular expressions are an incredibly powerful pattern mapping tool. ***They are, frankly, quite unintuitive and can be intimidating for beginners.***

Models

Models define your SQL tables without writing SQL code

Flask Model

```
class Products(db.Model):
    __tablename__ = 'products'
    barcode = db.Column(db.String(100), primary_key=True)
    price = db.Column(db.Float)
    vat_price = db.Column(db.Float)
    selling_price = db.Column(db.Float)
    manufacturer = db.Column(db.String(100),
                    db.ForeignKey('manufacturers.name'))
```

Django Model

```
class Post(models.Model):
    author = models.ForeignKey(settings.AUTH_USER_MODEL,
        on_delete=models.CASCADE)
    title = models.CharField(max_length=200)
    text = models.TextField()
    created_date = models.DateTimeField(default=timezone.now)
    published_date = models.DateTimeField(blank=True,
        null=True)
```

Views & URLS

Flask adds both the view function and the url in the same place. Django defines views in a file and the user specifies the url and the corresponding function in another place.

Views tell what to do in case the client requests a url

Flask Views & URLS

```
@people_blueprint.route("/")  # @app.route('/')
def people_main():
    context = base_context()

    context['people'] = People.query.all()
    return render_template('people/index.html', **context)
```

roughly same as

```
@people_blueprint.route("/")  # @app.route('/')
def people_main():
    context = base_context()

    return render_template('people/index.html',
        people=People.query.all())
```

Django Views & URLS

views.py

```
def post_new(request):
    if request.method == "POST":
        form = PostForm(request.POST)
        if form.is_valid():
            post = form.save(commit=False)
            post.author = request.user
            post.published_date = timezone.now()
            post.save()
            return redirect('post_detail', pk=post.pk)
    else:
        form = PostForm()
    return render(request, 'blog/post_edit.html',
        {'form': form})
```

in the app's url.py

```
urlpatterns = [
        ...
        path('post/new', views.post_new, name='post_new'),
        ...
]
```

Apps or the namespacing of urls

Apps in Django or modules / blueprints in Flask is just a way

- To isolate related codes in one place
- To namespace urls

Flask Blueprint

```
# in the people view
people_blueprint = Blueprint('people', __name__,
        url_prefix='/people')

# usage
#@people_blueprint.route("/")  # url for /people/
#def people_main():
```

in app.py

```
from views.people.people import people_blueprint

app.register_blueprint(people_blueprint)
```

Registering Django App

in settings.py

```
INSTALLED_APPS = [
    ...
    'blog'
]
```

in main urls.py

```
urlpatterns = [
    ...
    path('', include('blog.urls'))
]
```

Templates

Templates are html files to be rendered and presented to the user

Flask Template

```
{% extends "base/main_base.html" %}

{% set active_page ='settings' %}

{% block pagehead %}
<title>Settings</title>
<style>

</style>
{% endblock %}

{% block content %}
<table>

</table>

{% endblock %}
```

Django Template

```
{% extends 'blog/base.html' %}

{% block content %}
    <div class="post">
        {% if post.published_date %}
            <div class="date">
                {{ post.published_date }}
            </div>
        {% endif %}
        <a class="btn btn-default" href="
        {% url 'post_edit' pk=post.pk %}">
        <span class="glyphicon glyphicon-pencil"></span></a>
        <h2>{{ post.title }}</h2>
        <p>{{ post.text|linebreaksbr }}</p>
    </div>
{% endblock %}
```

Configuration Management

FLask typically has config from object or from config file and django has settings.py

Flask config class

```python
class Config:
    """Parent configuration class."""
    DEBUG = False
    SQLALCHEMY_DATABASE_URI = 'sqlite:///test.db'
    SQLALCHEMY_TRACK_MODIFICATIONS = False
    SECRET_KEY = 'qow32ijjdkc756osk5dmck'  # Need a generator
    APP_NAME = 'Demo'
    SECTION_NAME = 'Manufacturer'
    SECTION_ITEMS = 'Products'
    HOMEPAGE_URL = '/manufac/'


class DevelopmentConfig(Config):
    """Configurations for development"""
    ENV = 'development'
    DEBUG = True


app_config = {
    'development': DevelopmentConfig,
    'production': Config,
}
```

then

```
def create_app(config_name):
    app = Flask(__name__)
    app.config.from_object(app_config[config_name])
    ...
```

with usage

```
app = create_app('development')
```

Django

settings.py

```
import os

# Build paths inside the project like this: os.path.join(BASE_D
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__fi


# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/3.0/howto/deployment/ch

# SECURITY WARNING: keep the secret key used in production secr
SECRET_KEY = '3_o8t6*91-xp5md=6#5&$&^7+28+n5+$2rlotvx3!!-62lmz$
```

Quick note on routes

```
post/<int:pk>/
```

can both be used on Flask and Django

Forms

Flask and Django have form models to automatically generate and validate forms

Flask

```python
class LoginForm(FlaskForm):
    username = StringField('Username',
        validators=[DataRequired()])
    password = PasswordField('Password',
        validators=[DataRequired()])
    remember_me = BooleanField('Remember Me')
    submit = SubmitField('Sign In')
```

Django

```
class PostForm(forms.ModelForm):

    class Meta:
        model = Post
        fields = ('title', 'text',)
```

Ending words

Thank you for making it till the end. Flask and Django have many similar concepts which makes it easy to navigate between those two.

I'm personally a Flask fan but appreciate the Django project as they home grew their own ORM etc which is an impressive feat.

The main advantage of Flask is flexibility and freedom which frustrated Django devs turn to.

Django makes it easy to get started by providing basic utilities for web dev which allow websites to be more complete from the start. This in turn leads to greater community and business focus as evidenced by DjangoCon.

🎁 Bonus

💡 Django by far is more popular than Flask
💡 Django has far more jobs than Flask
💡 The basic Django admin is used by RealPython as their everyday login and post creation gateway
💡 If you want Flask, you can do it. Runninginproduction.com episode 10 podcast tells about Flask app with 65, 000+ requests per second with thousands of schools and millions of students
💡 Patreon is a Flask app

> Flask without a great base to start with is a pain and Django without understanding the vision behind makes it difficult to get started with