## Flask یک میکروفریمورک وب برای زبان پایتون است. flask بدلیل سادگی و کوچک بودن در دسته بندی میکروفریمورک ها قرار میگیرد اما این دلیلی بر ضعیف بودن flask نمیباشد. flask تنها هسته اصلی یک web framework را در اختیار شما قرار میدهد و برای انجام کارهای بیشتر باید از پکیج ها استفاده کنید. از flask میتوان در هر نوع پروژه ای, کوچک یا بزرگ استفاده کرد. flask توسط Armin Ronacher در سال 2010 به عنوان دروغ آوریل ساخته شد اما خیلی سریع به یک پروژه محبوب در سراسر جهان تبدیل شد.  میکروفریمورک بودن flask معایب و مزایای زیادی دارد. به عنوان عیب, به دلیل ساده بودن هسته اصلی فلسک مجبور خواهید بود کد بیشتری بنویسید اما با استفاده از همین سادگی میتوانید هر نوع معماری را پیاده سازی کنید که مزیت آن محاسبه میشود. برای درک قدرت فلسک کافیه بدانید که برای ساخت microservice پیشنهاد میشود که از فلسک بجای جنگو استفاده شود.

=====================================================================================

**intro :**

در این قسمت با فریمورک flask آشنا خواهید شد. flask به اندازه ای کوچک است که میکرو خوانده شود و بعد از کار کردن با آن بتوانید به سادگی شروع به خواندن و فهمیدن سورس کد کنید. اما کوچک بودن به معنی نیست که نسبت به بقیه فریمورک ها چیزی کم دارد. flask بطور کلی سه وابستگی دارد. routing و debugging و wsgi. برای کار کردن با templateها میتوانید از jinja و برای ارتباط با command-line میتوانید از click استفاده کنید. همه این موارد توسط Armin Ronacher ساخته شده اند.

=====================================================================================

**start :**

در این قسمت کار با فلسک را شروع میکنیم. نقطه شروع هر برنامه فلسکی با کلاس Flask است. با استفاده از آبجکتی که از این کلاس ایجاد میکنید میتوانید مسیرهای پروژه‌تان را مشخص کنید.

اولین فایلی که باید ساخته شود و از کلاس Flask در آن استفاده شود میتواند هر اسمی داشته باشد اما معمولاً اسم آن را app قرار میدهند

همانطور که در داکیومنت خوده فلسک وجود دارد instanceی که از کلاس Flask ساخته می‌شود درواقع wsgi برنامه ی شماست به این معنی که برنامه ی شما از این instance شروع می‌شود

اولین پارامتری که Flask میگیرد \_\_name\_\_ میباشد به این معنی که Flask متوجه می‌شود داخل application شما چه چیز هایی متعلق به برنامه ی شماست

به طور کلی پیشنهاد می‌شود که اگر از یک ماژول (در این دوره app.py ) استفاده میکنید بهتر است که اولین پارامتر کلاس Flask ــnameــ باشد اما اگر از پکیج ها استفاده میکنید بهتر اسم پکیج (دایرکتوری) خود را به کلاس Flask دهید

میتوانیم به هر فانکشن چندین rout متفاوت را اختصاص دهیم

=====================================================================================

**templates :**

در این قسمت با نحوه کار کردن با templates در فلسک آشنا میشوید. فلسک به صورت پیشفرض انتظار دارد فایلهای html را در یک دایرکتوری به نام templates ببیند. برای هدایت کاربر به فایلهای html میتوانید از متد render\_template استفاده کنید.

[https://flask.palletsprojects.com/en/2.0.x/api/#template-rendering](https://flask.palletsprojects.com/en/2.0.x/api/" \l "template-rendering)

برای render کردن template ها از jinja2 استفاده می‌شود به این مفهوم که اگر contextی از طریق render\_template به فایل html فرستاده شد با استفاده از تگ های jinja2 میتوانیم context را مدیریت کنیم ( در‌واقع jinja یک template engine میباشد )

=====================================================================================

**jinja :**

در این قسمت با jinja در فلسک آشنا میشوید. jinja یک template engine است که مسئول مدیریت نحوه نمایش اطلاعات در فایلهای html است. همچنین میتوانید با استفاده از ارثبری در templateها کد کمتری بنویسید.

<https://jinja.palletsprojects.com/en/3.0.x/>

با استفاده از jinja2 میتوانیم ارث بری در template ها را نیز انجام دهیم :

[https://jinja.palletsprojects.com/en/3.0.x/templates/#child-template](https://jinja.palletsprojects.com/en/3.0.x/templates/" \l "child-template)

=====================================================================================

**include :**

تگ include برای وارد کردن کدهایی که در فایل های دیگه html نوشته شده استفاده میشود. برای درک بهتر این موضوع میتوانیم فایل base.html را مشاهده کنید (‌توجه شود که این تگ نیز از jinja2 میباشد )

میتوان گقت که عملکرد include و extends باهم متفاوت است

=====================================================================================

**development server :**

در این قسمت نحوه اجرا شدن سرور فلسک را تغییر خواهیم داد. در روش قبلی با استفاده از دستور flask run اقدام به اجرا کردن سرور فلسک میکردیم اما مشکلی که این روش دارد اینست که پس هر بار تغییر در کد مجبوریم سرور را دوباره اجرا کنیم. اما میتوانیم دستور run را در کد نوشته و مقدار debug را True قرار داده تا سرور به صورت خودکار ریستارت شود.

/<https://flask.palletsprojects.com/en/2.0.x/server>

به طور کلی دو روش وجود دارد command-line و in code :  
in code به این صورت میباشد که به صورت مستقیم فایل app اجرا شود

حال اگر به متد run مقدار debug=True را بدهیم دیگر لازم نیست بعد از هر تغییر server را ریستارت کنیم

توجه شود که بعد از تغییرات بالا دیگر نباید دستور flask run را استفاده کنید بلکه باید از دستور python app.py استفاده کنید

مشکلی که در این روش وجود دارد این است که اگر ارور در کد وجود داشته باشد سرور stop می‌شود و بعد از حل مشکل دوباره باید سرور را ریستارت کنید

در روش command-line دیگر مشکل stop شدن سرور وجود ندارد و اگر مشکلی وجود داشته باشد ارور به شما نمایش داده می‌شود ( اگر FLASK\_DEBUG برابر true باشد همچین اتفاقی می‌افتد اما اروری که برمیگرداند ناخوانا میباشد) اما با false کردن FLASK\_DEBUG سرور بعد از هر تغییر دوباره خود ریستارت می‌شود اما اگر اروری وجود داشته باشد سرور stop می‌شود

=====================================================================================

**models :**

در این قسمت با modelها در فلسک آشنا میشوید و همچنین میبینید که چطور میتوان با استفاده از فلسک با دیتابیس کار کرد. در فلسک برای کار کردن با دیتابیس ها از پکیج flask-sqlalchemy استفاده میشود. برای ایجاد کردن جدول دیتابیس از modelها استفاده میشود. هر model یک کلاس است که نام جدول و انواع ستون های آن را مشخص میکند.

Flask sql alchemy یک رابط است که flask را به دیتابیس متصل میکند به این صورت که دیگر لازم نیست دستورات sql را استفاده کنید بلکه یک orm در اختیار شما قرار میدهد که بسیار کار را آسان‌تر میکند

/<https://flask-sqlalchemy.palletsprojects.com/en/2.x>

با استفاده از document زیر میتوانید یک دید کلی به دست آورید

[https://flask-sqlalchemy.palletsprojects.com/en/2.x/quickstart/#a-minimal-application](https://flask-sqlalchemy.palletsprojects.com/en/2.x/quickstart/" \l "a-minimal-application)

همانطور که در داکیومنت بالا مشاهده می‌شود /// بعد از تعیین مدل دیتابیس باید قرار بگیرد و بعد از /// آدرس دهی به فایل دیتابیس شروع می‌شود

مرحله بعد ساخت یک instance از SQLAlchemy میباشد (به عنوان مثال db)

هر مدل شما باید از db.Model ارث بری کند

=====================================================================================

**query :**

در این قسمت با نحوه ارتباط با دیتابیس در فلسک آشنا خواهید شد.

برای ساخت database از مدلی که نوشته‌اید کافیست که وارد shell پایتون شود ( در مسیری که قرار دارید دستور python را بزنید ) و دستورات زیر را وارد کنید :

from app import db

db.create\_all()

حال اگر بخواهید مقداری را در جدول Todo ذخیره کنید کافیست از روش زیر استفاده کنید( در مدل Todo دو فیلد id و date به صورت default خود مقدار میگیرند چرا که id را primary key قرار دادیم و برای date مقدار default تعیین کردیم):

from app import Todo

t1 = Todo(content = ‘this is my first todo’)

db.session.add(t1)

هنوز مقدار t1 در database ذخیره نشده است چرا که این مقدار در داخل session ( ترمینالی که فعال هست) قرار دارد برای دخیره سازی در داخل database باید دستور زیر را نیز بزنید

db.session.commit()

برای دیدن database میتوانید از viewer online استفاده کنید

https://inloop.github.io/sqlite-viewer/

به طور کلی برای خواندن اطلاعات میتوانیم از دو روش استفاده کنیم :  
روش اول تمام object ها را برمیگرداند ( استفاده از متد all ) روش دوم استفاده از متد های filter\_by و filter و order\_by و limit و get میباشد :

[https://flask-sqlalchemy.palletsprojects.com/en/2.x/queries/#querying-records](https://flask-sqlalchemy.palletsprojects.com/en/2.x/queries/" \l "querying-records)

todos = Todo.query.all()

first = Todo.query.filter\_by(id=1)

توجه شود که query به عنوان object manager عمل میکند

=====================================================================================

**get :**

برای اینکه بتوانیم داخل template ها لینک بسازیم میتوانیم از url\_for استفاده کنیم

[https://flask.palletsprojects.com/en/2.0.x/api/#flask.url\_for](https://flask.palletsprojects.com/en/2.0.x/api/" \l "flask.url_for)

به این صورت که اولین مقداری که دریافت میکند اسم فانکشن داخل app.py میباشد و مقادیر بعدی مقادیری است که باید به link مورد نظر ارسال شود

<h3><a href="{{ url\_for('detail', todo\_id=todo.id) }}">{{ todo.content }}</a></h3>

@app.route('/<int:todo\_id>')  
def detail(todo\_id):  
 todo = Todo.query.get(todo\_id)  
 return render\_template('detail.html', todo=todo)