1901

Flask

ارائه مباحث ویژه 2

نصب و شروع کار با فریمورک فلسک به همراه ذکر مثال از نحوه ی ساخت

دانشجو: دریا ظریف نیا

استاد راهنما: جناب اقای دکتر طالب

دانشکده غیرانتفاعی رفاه

ساخت محیط مجازی(Virtual Environment) :

پروژه‌ای که می‌خواهید در آن محیط مجازی بسازید را ایجاد کنید.

1) به منوی File بروید و گزینه Settings را انتخاب کنید (برای کاربران ویندوز) یا PyCharm در منوی بالا (برای کاربران مک) و سپس Preferences را انتخاب کنید.
2) از منوی سمت چپ، به قسمت] Project: نام پروژه شما] بروید و سپس Project Interpreter را انتخاب کنید.
3) در بالای صفحه، کنار محل نمایش تفسیرگر (Interpreter) فعلی، روی چرخ‌دنده کوچک کلیک کنید و ...Add را انتخاب کنید.
4) در پنجره باز شده، گزینه‌ی Virtualenv Environment را از سمت چپ انتخاب کنید.
5) حالا گزینه‌های زیر را تنظیم کنید:
   - Location: مسیری را که می‌خواهید محیط مجازی در آن ساخته شود را انتخاب یا تایپ کنید.
   - Base interpreter: تفسیرگر پایتونی را که می‌خواهید استفاده کنید، انتخاب کنید. این معمولاً باید نشان‌دهنده نصب پایتون شما باشد.
   - می‌توانید تیک گزینه‌های Inherit global site-packages و Make available to all projects را بر اساس نیاز خود فعال یا غیرفعال کنید.
6) بعد از تنظیم گزینه‌ها، روی OK کلیک کنید تا محیط مجازی ساخته شود.
7) PyCharm شروع به ساخت محیط مجازی می‌کند و پس از اتمام، می‌توانید از محیط مجازی جدید استفاده کنید.

❖ با ساخت محیط مجازی، تمام کتابخانه‌ها و بسته‌هایی که نصب می‌کنید، فقط در آن محیط مجازی قابل دسترسی خواهند بود و تأثیری بر سایر پروژه‌ها یا تنظیمات سیستمی نخواهند داشت.

گزینه‌های Inherit global site-packages و Make available to all projects در PyCharm هنگام ساخت محیط مجازی (Virtual Environment) به شما این امکانات را می‌دهند:

1. Inherit global site-packages: اگر این گزینه را فعال کنید، محیط مجازی جدید دسترسی به بسته‌های (packages) نصب شده در سطح سیستم (global site-packages directory) خواهد داشت. به این ترتیب، شما نیازی نیست همه کتابخانه‌هایی که قبلاً به صورت جهانی نصب کرده‌اید را دوباره در محیط مجازی نصب کنید، بلکه مستقیماً قادر به استفاده از آن‌ها خواهید بود.

2. Make available to all projects: با فعال کردن این گزینه، محیط مجازی ایجاد شده در PyCharm برای تمام پروژه‌های فعلی و آینده‌ی شما در PyCharm قابل دسترسی خواهد بود. به عبارت دیگر، شما می‌توانید یک محیط مجازی را بین چندین پروژه به اشتراک بگذارید بدون اینکه نیاز باشد برای هر کدام به صورت جداگانه محیطی را پیکربندی کنید.

استفاده از این گزینه‌ها به نوع پروژه و نیازهای خاص شما بستگی دارد. برای مثال، اگر می‌خواهید محیطی کاملاً جدا از نصب‌های سیستمی داشته باشید، ممکن است بخواهید گزینه Inherit global site-packages را غیرفعال کنید. اگر محیطی را می‌خواهید که بین پروژه‌ها قابل استفاده باشد، می‌توانید گزینه Make available to all projects را فعال کنید.

فلسک چیست ؟

Flask یک سرویس سمت سرور نیست

Flask یک وب سرور نیست

Flask زبان برنامه نویسی نیست

Flask بهترین ابزار برای وارد شدن به زبان پایتون نیست

در واقع Flask یک فریم ورک وب سبک می‌باشد که با زبان پایتون طراحی و نوشته شده است.

به عبارت دقیق‌تر فلسک یک میکرو فریم ورک است چون بسیاری از ابزارها و کتابخانه‌های رایج سایر فریم ورک‌ها را ندارد.

مثلا فلسک به طور پیش فرض نمی‌تواند با دیتابیس کار کند با فرم‌ها را اعتبارسنجی کند و برای این کار باید سراغ کتابخانه‌ها و دیتابیس‌های موجود بروید.

Flaskدر کجاها استفاده می‌شود؟

Flask کتابخانه ای کوچک یا به اصطلاح جمع و جور است.

- Uber که بزرگترین سرویس درخواست تاکسی اینترنتی
- سامسونگ یکی از بزرگترین تولیدکننده‌های قطعات دیجیتال در دنیا

به عنوان مثالی از برنامه‌های کاربردی که از فریمورک فلسک پشتیبانی می‌کنند می‌توان به «پینترست (Pinterest)و «لینکدین (LinkedIn) «اشاره کرد.

- Red Hat
- Netflix
- Reddit
- Pinterest
- Linkedin
- Mozila
- Hotjar
- Nginx

چرا باید از Flask استفاده کنیم؟

فریم ورک‌ها مجموعه ای از کدهای آماده هستند که کار را برای برنامه نویس آسان می‌کنند.

در واقع فلسک به شما کمک می‌کند بدون درگیر شدن با مسائل پیچیده سطح پایین، تمرکز خود را روی توسعه سرویس خود بگذارید.

این نکته را یادآور شویم که هیچکدام از فریم ورک‌ها از دیگری بالاتر نیست. بعضی از نقاط قوت فلسک که برنامه نویسان را به استفاده از آن ترغیب می‌کنند:

- یادگیری Flask بسیار آسان است.
- این فریم ورک کاملا انعطاف پذیر است.
- یک جامعه قوی پشت زبان پایتون و فریم ورک فلسک قرار دارد.

ساخت یک برنامه تحت وب ساده باFlask

```
from flask import Flask
app = Flask(__name__)
@app.route('/')
def home():
  return "<h1> Hello World </h1>"
if __name__ =="__main__":
  app.run(debug=True,port=8080)
```

این کد مربوط به یک برنامه وب ساده است که با استفاده از فریمورک Flask در زبان برنامه‌نویسی Python نوشته شده است. به تفکیک هر خط توضیح داده می‌شود:

[باید Flask Object را وارد کنید. با این کار تابعی به‌وجود می‌آورید که می‌تواند پاسخ HTTP را برگرداند.]

1. from flask import Flask:

این خط کلاس Flask را از بسته flask وارد می‌کند. کلاس Flask پایه و اساس ایجاد برنامه‌های وب در فریمورک Flask است.

2. app = Flask(__name__) :

به پارامتر name توجه کنید. این پارامتر برای قراردادن یک نام در اپلیکیشن شما استفاده می‌شود.

یک نمونه از کلاس Flask ساخته می‌شود. name یک متغیر ویژه در Python است که به نام ماژول جاری اشاره دارد. این کار به Flask کمک می‌کند تا بداند کجا برای منابع، قالب‌ها و فایل‌های ثابت بگردد.

3. @app.route ('/') :

این @app.route نیز به‌عنوان دکوراتور شناخته می‌شود. @app.route می‌تواند تابع معمولی در پایتون را به تابع نمای فلاسک تبدیل کند. این فرایند می‌تواند مقدار بازگشتی تابع را به پاسخ HTTP تبدیل کند. بدین‌ترتیب، ازطریق گیرنده HTTP شبیه به مرورگر وب نشان داده می‌شود. همچنین، با قراردادن مقدار '/' به @app.route() نشان می‌دهید که این تابع به درخواست‌های وب برای URL/ که URL اصلی است، پاسخ می‌دهد.

این یک Decorator است که مسیر URL ریشه یا Home (/) را به تابعی که بلافاصله بعد از آن می‌آید، اختصاص می‌دهد. این به این معنی است که وقتی کاربر به این URL مراجعه کند، تابع مرتبط اجرا می‌شود.

4. def home () :

تعریف تابع home() است که وقتی کاربر به مسیر URL ریشه (/) مراجعه کند، فراخوانی می‌شود.

5. return "<h1>Hello World</h1 "> :

این خط مقدار را به کاربر برمی‌گرداند. در اینجا، یک رشته HTML است که به مرورگر ارسال می‌شود و به کاربر نمایش داده می‌شود که شامل متن "Hello World" درون یک تگ عنوان (h1) است.

6. if name =="__main__" :

این خط بررسی می‌کند که آیا اسکریپت به عنوان اسکریپت اصلی اجرا می‌شود. این امر معمولاً برای جلوگیری از اجرا شدن بخش‌های خاصی از کد هنگامی که فایل به عنوان ماژول وارد می‌شود، مورد استفاده قرار می‌گیرد.

7. app.run(debug=True, port=8080) :

این خط سرور توسعه‌ی Flask را با فعال‌سازی حالت دیباگ (debug) روی پورت 8080 اجرا می‌کند. فعال بودن حالت دیباگ به توسعه‌دهنده کمک می‌کند تا اشکال‌زدایی را آسان‌تر انجام دهد، زیرا تغییرات بدون نیاز به راه‌اندازی مجدد سرور اعمال می‌شوند و خطاها به صورت دقیق‌تر نمایش داده می‌شوند.

پیغام‌های نمایش داده شده در هنگام اجرای برنامه Flask، اطلاعات مفیدی در مورد وضعیت سرور و تنظیمات آن ارائه می‌دهند:

- Serving Flask app 'main' :

این خط اعلام می‌کند که برنامه Flask با نام 'main' در حال سرویس‌دهی است. نام 'main' از نام فایل اجرایی (main.py) گرفته شده است.

- Debug mode: on:

این خط نشان‌دهنده فعال بودن حالت دیباگ است. حالت دیباگ به توسعه‌دهندگان کمک می‌کند تا خطاها را بهتر تشخیص دهند و برنامه را به صورت دینامیک بدون نیاز به راه‌اندازی مجدد سرور پس از هر تغییر، توسعه دهند.

- WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.:

این پیام یک هشدار است که بیان می‌کند سرور در حال اجرا یک سرور توسعه است و باید از آن در محیط‌های تولیدی استفاده نشود. برای محیط‌های تولید، استفاده از یک سرور WSGI توصیه می‌شود.

- Running on http://127.0.0.1:8080:

این خط آدرس و پورتی که سرور Flask در حال اجرا بر روی آن است را نشان می‌دهد. در این مورد، آدرس 127.0.0.1 (که همان localhost است) با پورت 8080 است.

- Press CTRL+C to quit:

این خط به کاربر توصیه می‌کند که برای توقف سرور، کلیدهای CTRL+C را فشار دهد.

- Restarting with stat:

این خط اعلام می‌کند که سرور Flask در حال راه‌اندازی مجدد با استفاده از ویژگی 'stat' است، که بخشی از فرایند حالت دیباگ است که تغییرات فایل‌ها را زیر نظر می‌گیرد.

- Debugger is active!:

این خط اعلام می‌کند که اشکال‌زدای (debugger) فعال است، که به توسعه‌دهندگان امکان می‌دهد تا بهینه‌سازی کنند و خطاها را در کد خود رفع کنند.

- Debugger PIN: 231-273-482:

این خط یک شماره شناسایی شخصی (PIN) را برای اشکال‌زدای محافظت‌شده نشان می‌دهد. این PIN برای دسترسی به محیط اشکال‌زدای در مرورگر لازم است و امنیت اشکال‌زدای را تضمین می‌کند.

این پیغام‌ها، که به آن‌ها لاگ‌های سرور گفته می‌شود، فعالیت‌هایی هستند که روی سرور وب شما رخ داده‌اند. هر خط اطلاعات مربوط به یک درخواست HTTP که به سرور ارسال شده را نشان می‌دهد:

1. 127.0.0.1 - - [31/May/2024 22:15:53] "GET / HTTP/1.1" 200 -:

این خط نشان می‌دهد که درخواستی از نوع GET به مسیر ریشه (/) ارسال شده است و سرور با کد وضعیت HTTP 200 پاسخ داده است، که به معنای «موفقیت‌آمیز» است.

2. 127.0.0.1 - - [31/May/2024 22:16:13] "GET /favicon.ico HTTP/1.1" 404 -:

این خط نشان می‌دهد که یک درخواست GET برای فایل favicon.ico/ ارسال شده است و سرور با کد وضعیت HTTP 404 پاسخ داده است، که به معنای «یافت نشد» است. این اغلب اتفاق می‌افتد زیرا مرورگرها به صورت خودکار سعی می‌کنند آیکونی را برای سایت دریافت کنند و اگر فایل مربوطه در سرور وجود نداشته باشد، با خطای 404 مواجه می‌شوند.

3. خطوط دیگری که با "GET / HTTP/1.1" 200 ختم می‌شوند نشان‌دهنده درخواست‌های موفق دیگری به مسیر ریشه هستند، جایی که سرور به درستی پاسخ داده و صفحه مورد نظر (در این مورد "Hello World") را نمایش داده است.

این لاگ‌ها می‌توانند برای تحلیل ترافیک وب‌سایت، شناسایی خطاها و مشکلات احتمالی، و اطمینان از عملکرد صحیح سرور مورد استفاده قرار گیرند.

آشنایی با if name == main در پایتون

در پایتون امکان این وجود دارد که بتوانیم یه function رو جوری برنامه ریزی کنیم که فقط در صورتی که به صورت مستقیم صدا زده شد اجرا شود.

زمانی که شما یک کد پایتونی رو اجرا میکنید، پایتون میاد چندتا special variable رو ایجاد میکنه و یه سری اطلاعات رو داخل اونها ذخیره میکنه. یکی از این متغیرهای خاص __name__ هستش.

به طور کلی __name__ دو تا مقدار میگیره:

1. اگه شما یک ماژول رو به صورت مستقیم run کنید مقدار __main__ رو میگیره.

2. و اگه جای دیگه import کرده باشیدش اسم اون ماژول رو میگیره.

https://7learn.com/blog/what-is-flask

آموزش templates در فلسک

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/3.0.x/quickstart/#a-minimal-application

https://flask.palletsprojects.com/en/3.0.x/quickstart/#rendering-templates

آموزش jinja در فلسک

jinjaیک template engine است که مسئول مدیریت نحوه نمایش اطلاعات در فایلهای html است. همچنین میبینید که چطور میتوانید با استفاده از ارثبری درtemplate ها کد کمتری بنویسید.

https://jinja.palletsprojects.com/en/3.1.x/

https://jinja.palletsprojects.com/en/3.1.x/templates/

https://jinja.palletsprojects.com/en/3.1.x/templates/#list-of-builtin-filters

آموزش تگ include در فلسک

با استفاده از bootstrap اقدام به استایل دهی به برنامه‌مون میکنیم و همچنین با تگ include در فلسک آشنا میشوید. تگ include برای وارد کردن کدهایی که در فایل های دیگه نوشته شده استفاده میشود.

/https://getbootstrap.com

وارد سایت شده لینک بوت مربوط به css را بعد تگ title در فایل base جایگذاری کنید.

در فایل base بلاک content رو میزاریم داخل div : پس زمینه خاکستری میده.

```
<div class = "bg-light p-5"> </div>
```

```
<div class="bg-secondary text-white p-5"> </div>
```

حالا وارد لینک زیر شده و بخش Navvar را اضافه میکنیم به بخش body قبل از div :

/https://getbootstrap.com/docs/5.3/components/navbar

نوبار را ساده سازی میکنیم و رنگ لایت هم میتونیم بهش بدیم :

```
<nav class="navbar navbar-expand-lg bg-secondary">
      <ul class="navbar-nav">
          <a class="nav-link active text-white" aria-current="page"
href="#">Home</a>
      </ul>

</nav>
```

حالا وارد بخش jinja سپس بخش template designer documentation و بخش include را پیدا کنید.

فایل html دیگری باید برای navbar ایجاد میکنیم تا کد ها تمیز تر شوند سپس کدهای نوبار را داخلش وارد کرده و در فایل base از تگ include استفاده میکنیم در ابتدای body .

```
{% include 'navbar.html' %}
```

دقت کنید include را با extends اشتباه نگیرید ! این دو کاملا با یکدیگر متفاوت اند.

آموزش models در فلسک

در این بخش با model ها 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 ازmodel ها استفاده میشود. هر model یک کلاس است که نام جدول و انواع ستون های آن را مشخص میکند.

/https://flask-sqlalchemy.palletsprojects.com/en/3.1.x

سپس به بخش Quick Star بروید.

کتابخانه های مربوطه را اضافه میکنیم سپس دستور زیر را بعد از نمونه ی ایجاد شده از فلسک مینویسیم:

```
app.config["SQLALCHEMY_DATABASE_URI"] = " sqlite:///flask.db"
```

سپس یک نمونه از کلاس db ایجاد میکنیم. با استفاده از این نمونه میتوانید با دیتابیس کار کنید مثلا مواردی را حذف کنید یا ستون های آن را ایجاد کنید یا فیلد هایمان را آپدیت کنیم .

```
db = SQLAlchemy(app)
```

حالا میخواهیم table هایی را که داخل دیتا بیس وجود دارند را ایجاد کنیم. کلاس table باید از db.model ارث بری کنند. در برنامه نویسی به این table ها میگیم مدل. مدل کلاسی هست که base ، tableهای ما است.

در اینجا ما مدل Todo را داریم که یک سری ویژگی میگیرد. حال با کد زیر برای کلاس Todo مدل ایجاد میکنیم.

```
class Todo(db.Model):
    id = db.Column(db.Integer, primary_key=True)
    content = db.Column(db.Text, unique=True, nullable=False)
    date = db.Column(db.DateTime, default=datetime.now)
```

```
def __repr__(self):
    return f'Todo({self.id} - {self.content} - {self.date})'
```

الان یک دیتا بیس ایجاد کردیم که اسمش flask است و داخل کلاس یک table داریم بنام todo.

ارتباط با دیتابیس در فلسک

https://flask-sqlalchemy.palletsprojects.com/en/3.1.x/quickstart/#create-the-tables

حال میخواهیم دیتابیس را ایجاد کنیم و مقادیر را داخلش وارد کنیم برای اینکار در کامند، مفسر پایتون را صدا میزنیم.

<< flask shell

< python

<<<from app import db

```
>>> db.create_all()
```

یا

```
with app.app_context():
    db.create_all()
```

```
>>> from app import Todo

>>> t1 = Todo(content='this is my first todo')

>>> db.session.add(t1)

>>> db.session.commit()
```

مراحل بالا را میتوانید در یک مثال دیگر در لینک زیر مشاهده کنید:
https://virgool.io/@alisharify7/%DA%A9%D8%A7%D8%B1-%D8%A8%D8%A7-flask-sqlalchemy-%D8%AF%D8%B1-flask-v2p78pxmoixz

برای مشاهده دیتابیس ساخته شده از یک ابزار آنلاین استفاده میکنیم:

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

برای خواندن دیتا از دیتا بیس دو راه وجود دارد:

1) All
2) Filter.by

Query آبجکت منیجر ما است. مدیری میزاره روی table ها و مدا هامون کاری انجام بدیم.

```
>>>  todos = Todo.query.all()

>>> todos
```

نمایش اطلاعات به دلیل استفاده از متد repr است.

نمایش اطلاعات دیتابیس در فلسک

در این بخش میخواهیم اطلاعاتی که در دیتابیس داریم رو داخل template به کاربر نشون میدیم.

داخل متد home میریم و دستور خواندن All را مینویسیم.

نحوه ی ارسالش به html به صورت key value است پس در بخش return اسم کلید را میدیم todos و value را هم میدم . todos

```
@app.route("/home")
def home():
   todos = Todo.query.all()
   return render_template('home.html', todos=todos)
```

در فایل home.html داخل بلاک content را با مقدار {{todos}} تغیر میدیم .

خروجی را در وب سایت به صورت یک لیست مشاهده میکنیم .

باید حلقه ای برای نمایش جداگانه ی مقادیر داخل لیست ایجاد کنیم.

https://jinja.palletsprojects.com/en/3.1.x/templates/#list-of-control-structures

```
{% for todo in todos %}

    <h3> {{ todo.content }} </h3>
    <small> {{ todo.date }} </small>

{% endfor %}
```

```
{% block content %}
    <div class="card">
        {% for todo in todos %}

            <h3> {{ todo.content }} </h3>
            <small> {{ todo.date }} </small>

        {% endfor %}
    </div>
{% endblock %}
```

```
{% block content %}
    <div class="card">
        {% for todo in todos %}
            <div class="card-body">
                <h3>{{ todo.content }}</h3>
                <small>{{ todo.date}}</small>
            </div>
        {% endfor %}
    </div>
{% endblock %}
```

برای تغیر نمایش تاریخ از متد strftime استفاده میکنیم.

```
<small>{{ todo.date.strftime('%Y-%m-%d') }}</small>
```

آموزش متد get در فلسک

در این بخش با متد get در فلسک آشنا میشوید. با استفاده از متد get میتوانید یک آبجکت بخصوص را از دیتابیس دریافت کنید.

/https://flask-sqlalchemy.palletsprojects.com/en/3.1.x/queries

برای اینکه content هایمان را تبدیل به لینک یا دکمه کنیم باید کد زیر را به app.py اضافه کنیم:

```
@app.route("/<todo_id>")
def detail(todo_id):
    todo = Todo.query.get(todo_id)
    return render_template('detail.html', todo=todo)
```

در فایل home.html باید تگ a داخل تگ h3 برای ایجاد لینک اضافه کنیم:

```
<h3><a href=""> {{ todo.content }} </a></h3>
```

تابع url_for در Flask یک تابع بسیار کاربردی است که برای ساختن URLها بر اساس نام‌های تابع‌های View استفاده می‌شود. این رویکرد به شما اجازه می‌دهد تا به جای نوشتن URLهای سخت‌کد شده در کد خود، URLهایی دینامیک ایجاد کنید که با تغییر مسیرها در فایل‌های مسیریابی، خود به خود به‌روز می‌شوند.

- ❖ مزایای استفاده از url_for:

1. نگه‌داری آسان‌تر کد: تغییر URLها در یک مکان واحد (توابع view) و به‌روزرسانی خودکار آن‌ها در کل برنامه.
2. جلوگیری از خطاهای احتمالی: کاهش احتمال خطاهای تایپی یا مسیریابی نادرست در هنگام نوشتن URL به صورت دستی.
3. امنیت بهتر: جلوگیری از حملات مختلف مسیریابی که ممکن است در صورت استفاده از URLهای سخت‌کد شده رخ دهد.

در مثال زیر url_for نام تابع View مرتبط با URL را به عنوان آرگومان می‌گیرد (' detail ' در این مثال) و URL مرتبط با آن تابع را برمی‌گرداند. این کار به شما اجازه می‌دهد تا URL را در template خود به صورت دینامیک ایجاد کنید، بدون اینکه نیاز باشد URL را به صورت سخت‌کد شده وارد کنید.

در اینجا todo-id به عنوان بخشی از مسیر url قرار میگیرد.

```
<h3><a href="{{ url_for('detail', todo_id=todo.id) }}">{{
todo.content }}</a></h3>
```

روی لینک در وب سایت کلیک کنیم ← id را ارسال میکند به app.py و اطلاعات را از دیتابیس میخونه و به ما برمیگرداند.

حال باید فایل detail.html را ایجاد کنیم.

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

{% block content %}

    <h3>{{ todo.content }}</h3>
    <small>{{ todo.date.strftime('%Y-%m-%d') }}</small>

{% endblock %}
```

میشه همینکار را برای homeانجام داد:

```
<a class="nav-link active text-white" aria-current="page" href=" {{
url_for('home') }}">Home</a>
```

مرور کلی :

وقتی رو ی لینک کلیک میکنیم در واقع این اتفاق می افتد که id را ارسال میکند به app.py و اطلاعات را از داخل دیتا بیس میخواند و به ما برمیگرداند .

یه rout داریم که یک id را میگیرد و میفرستد داخل متد detail و این متد از داخل متد Todo با استفاده از get ، todo-id یعنی

اطلاعات را میخواند میاره و بعدش که خوند مارا میفرسته به detail.html .

بعد اگر بخوایم به توابع مون پارامتر ارسال کنیم میریم داخل home.html از url-for استفاده میکنیم و url-for اسم متد (detail)

را میگیرد و بعد پارامتری را که میخوایم بهش میگیم.