

آموزش پروژه محور

جنگو

محمد رضا ملا حسینی اردکانی

مهرماه ۱۴۰۱

ساختار MVT در جنگو

ساختار MVT در جنگو

جنگو، یک فریم‌ورک پایتون برای ایجاد برنامه‌های کاربردی وب، بر اساس معماری **Model-View-Template (MVT)** است. **MVT** یک الگوی طراحی نرم‌افزار برای توسعه یک برنامه وب است. MVT از سه موجودیت زیر تشکیل شده است:

Model .۱

View .۳

Template .۳

Model .۱

مدل، یک شی است که ساختار داده‌ها را در برنامه جنگو تعریف می‌کند.

✓

مدل، مسئول حفظ کل داده‌های برنامه است.

✓

مدل قرار است به عنوان رابط داده‌های برنامه عمل کند.

✓

مدل، ساختار داده منطقی برنامه است که در نهایت توسط یک پایگاه داده (به طور کلی پایگاه داده‌های رابطه‌ای مانند MySQL، Myserver، Postgres) نشان داده می‌شود.

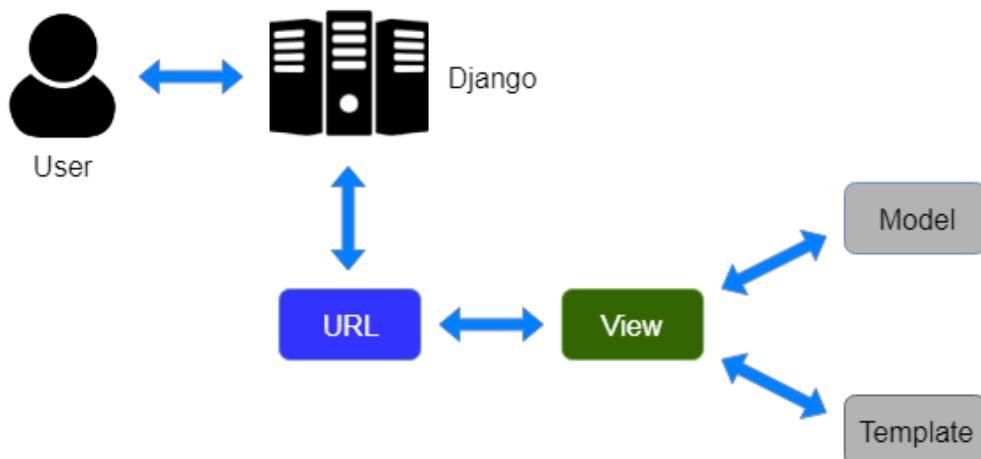
Template .۲

✓

تمپلیت، یک لایه ارائه است که بخش رابط کاربری را به طور کامل مدیریت می‌کند.

View .۳

ویو، یک تابع کنترلگر است که درخواست‌های HTTP را می‌پذیرد، داده‌های لازم برای انجام درخواست را با استفاده از Models بازیابی می‌کند، آنها را پردازش و با استفاده از Templates آنها را در رابط کاربری در قالب پاسخ HTTP ارائه می‌کند.



همانطور که در تصویر بالا می‌بینید، زمانی که یک کاربر درخواستی را ارسال می‌کند، مراحل زیر طی می‌شود:

۱) کاربر یک درخواست از طریق URL به برنامه جنگویی ارسال می‌کند.

۲) جنگو به دنبال یک URL معتبر در خود می‌گردد.

۳) اگر URL به view خاصی متصل باشد آنگاه آن view صدای زده خواهد شد.

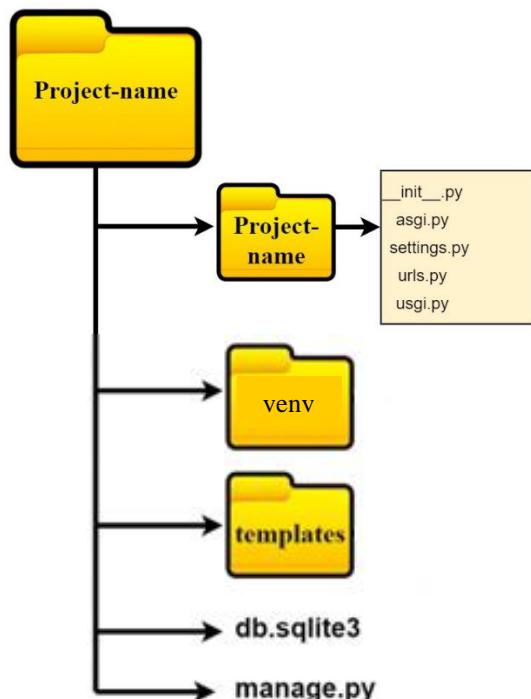
۴) در صورت نیاز view به model متصل شده و اطلاعات مناسبی را دریافت خواهد کرد.

۵) در آخر view یک بخش از templates، همراه با اطلاعات دریافتی از model را به کاربر نمایش می‌دهد.

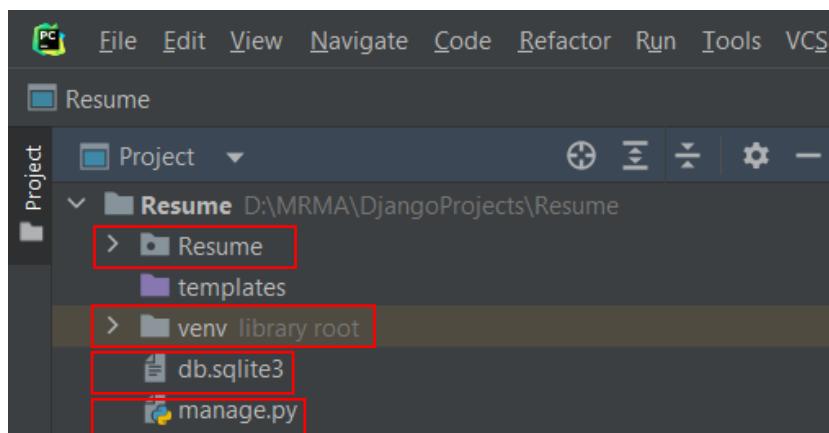
ساختار دایرکتوری‌ها و فایل‌ها در جنگو

ساختار دایرکتوری‌ها و فایل‌ها در جنگو

جنگو از ساختار دایرکتوری برای مرتب کردن بخش‌های مختلف برنامه وب استفاده می‌کند. ایجاد یک پروژه و سازماندهی مناسب آن به DRY (Don't Repeat Yourself) و Clean (تمیز نگه داشتن) پروژه کمک می‌کند. هنگامی که شما داخل پای‌چارم یک پروژه جنگویی ایجاد می‌کنید، خود جنگو یک دایرکتوری ریشه همانم با نام پروژه ایجاد می‌کند. دایرکتوری دیگر همان محیط مجازی یا venv است. همچنین دو فایل db.sqlite3 و manage.py را در دایرکتوری پروژه مشاهده می‌کنید:



برای پروژه مثال ما:



manage.py❶

این فایل به عنوان یک ابزار خط فرمان برای پروژه‌های شما عمل می‌کند. شما از این فایل برای اشکال زدایی، استقرار و اجرای برنامه‌های وب خود استفاده خواهید کرد.

چندین دستور مهم با manage.py قابل اجرا است:

الف. runserver

این دستور برای راهاندازی سرور آزمایشی ارائه شده توسط فریمورک جنگو برای اجرای برنامه وب شماست.

ب. makemigrations

از این دستور برای تعریف تغییرات در پایگاه داده استفاده می‌شود (هنوز تغییرات در پایگاه اعمال نمی‌شود).

ج. migrate

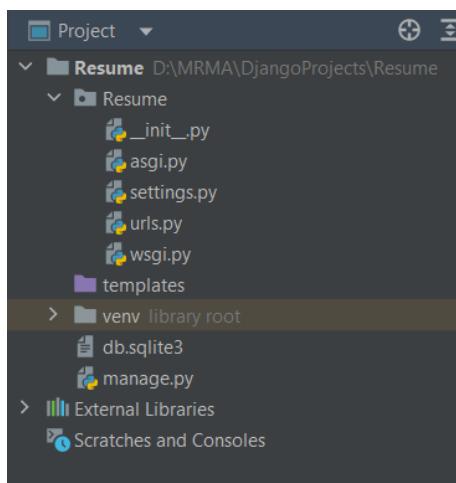
این مرحله بعد از دستور makemigrations است. از این دستور برای ایجاد تغییرات در ماثول‌های پایگاه داده استفاده می‌شود.

db.sqlite3 ②

این فایل یک فایل پایگاه داده‌ای است که در آن تمام داده‌هایی که تولید می‌کنید ذخیره می‌شود. اگر از SQLite استفاده می‌کنید، پایگاه داده یک فایل در رایانه شما خواهد بود. این فایل در دایرکتوری پروژه شما ذخیره می‌شود (BASE_DIR / 'db.sqlite3').

۳ دایرکتوری ریشه جنگو

هنگام ایجاد یک پروژه جدید، یک دایرکتوری ریشه همنام با نام پروژه ایجاد می‌شود. این دایرکتوری دربردارنده فایل‌های پیکربندی پروژه است:



۱. __init__.py

این فایل، یک فایل خالی است. عملکرد این این است که به مفسر پایتون بگوید که دایرکتوری ریشه یک پکیج است، وجود این فایل (_____.py) در این دایرکتوری، آن را به یک پروژه پایتون تبدیل می‌کند.

۲. settings.py

این فایل شامل پیکربندی پروژه جنگو است. setting.py مهمترین فایل است:

✓ از این فایل برای افزودن همه App‌ها و برنامه‌های میان افزار استفاده می‌شود.

✓ این فایل حاوی تنظیمات اصلی پروژه جنگو است.

✓ این فایل حاوی چندین نام متغیر است که اگر مقدار آنها را تغییر دهید، برنامه شما مطابق با آن کار خواهد کرد.

✓ این فایل حاوی sqlite3 به عنوان پایگاه داده پیش فرض است. شما می‌توانید این پایگاه داده را به PostgreSQL، Mysql یا MongoDB با توجه به برنامه وب که ایجاد می‌کنید، تغییر دهید.

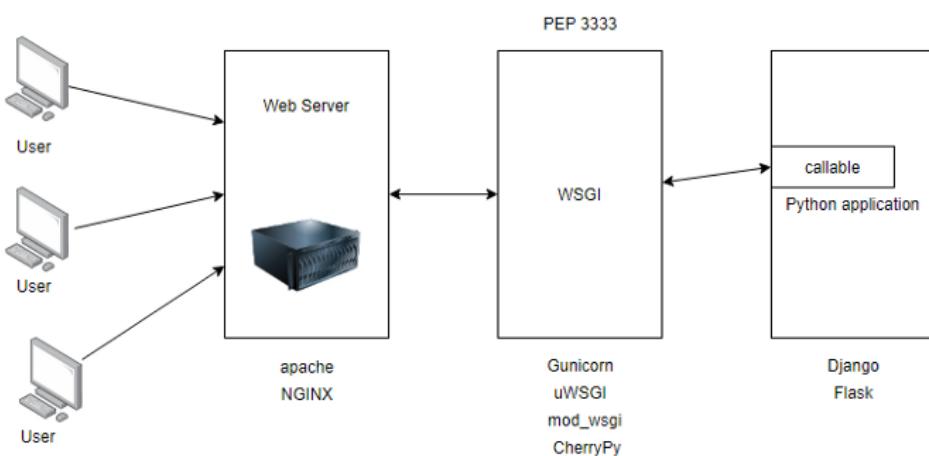
✓ این فایل دربردارنده برخی App‌ها و میان‌افزارهای از پیش نصب شده‌ای است که برای ارائه عملکردهای اصلی نیازند.

urls.py .۳

URL یک منبع‌باب جهانی است و از آن برای ارائه آدرس منابعی (تصاویر، صفحات وب، وب‌سایت‌ها و غیره) که در اینترنت وجود دارند، استفاده می‌شود. به عبارت ساده‌تر، این فایل به جنگو می‌گوید که اگر کاربری یک URL مشخصی ارائه می‌دهد، او را به کجا هدایت کند.

wsgi.py .۴

زمانی که وب‌سایت را کامل توسعه دادید، وظیفه بعدی میزبانی برنامه تحت وب شماست. WSGI یا Web Server Gateway Interface یک وب‌서ور مختص زبان پایتون است که ممکن می‌کند تا بتوانیم کدهای پایتونی را روی سرور اجرا کنیم. نحوه تعامل سرورها با App‌ها را شرح می‌دهد. برای انجام تعامل بین وب‌سرور و برنامه پایتون به یک واسطه WSGI نیاز است. استاندارد برای انجام ارتباط بین وب‌سرور و برنامه پایتون، WSGI است.



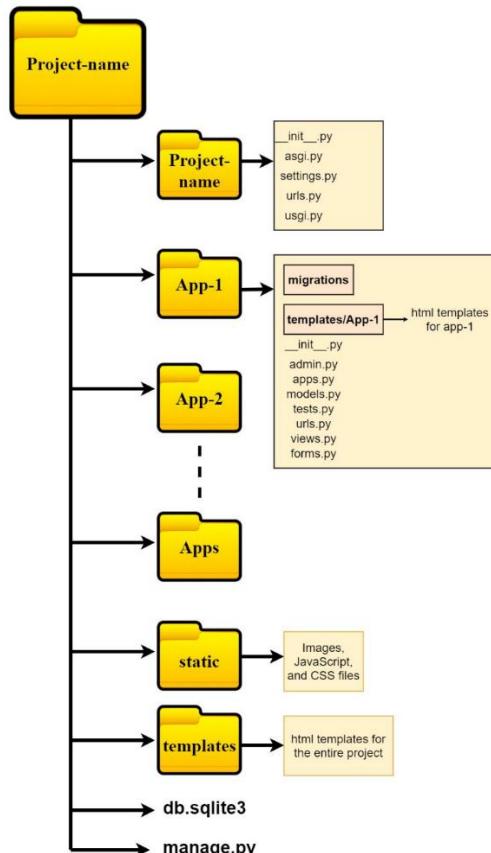
[//what-is-wsgi\۷۲ https://www.mongard.ir/articles/](https://www.mongard.ir/articles/): توضیح بیشتر

دایرکتوری venv ④

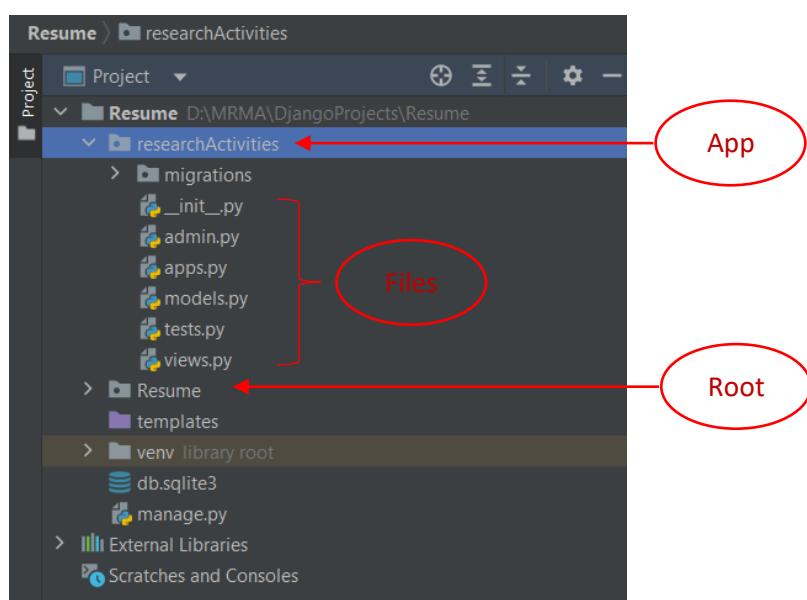
دایرکتوری venv شامل تمام فایل‌های اجرایی لازم برای استفاده از پکیج‌هایی است که یک پروژه پایتون به آن نیاز دارد.

جنا^گو های App

جنگو از مفهوم پروژه‌ها و App‌ها برای مدیریت کدها استفاده می‌کند و آنها را در قالبی خوانا ارائه می‌دهد. یک پروژه جنگو شامل یک یا چند App در داخل آن است. به عنوان مثال، یک سایت تجارت الکترونیک جنگو در دنیای واقعی یک App برای احراز هویت کاربر، یک App دیگر برای پرداخت‌ها، و یک App سوم برای جزئیات فهرست کالاها خواهد داشت: هر کدام بر روی یک عملکرد واحد تمرکز خواهند کرد.



پس از ایجاد یک App، فایل‌های فرعی زیر در آن App وجود خواهند داشت:



۱. init_.py_

این فایل همان عملکرد فایل `init_.py` در ساختار پروژه جنگو را ارائه می‌دهد. این یک فایل خالی است و نیازی به تغییر ندارد. این فقط نشان می‌دهد که دایرکتوری App یک پکیج است.

۲. admin.py

فایل `admin.py` برای نمایش مدل‌های برنامه داخل پنل مدیریت جنگو استفاده می‌شود. همچنین می‌توانید پنل مدیریت خود را سفارشی کنید.

۳. apps.py

فایلی است برای کمک به کاربر برای پیکربندی App‌ها. کاربران می‌توانند ویژگی‌های App خود را با استفاده از فایل `apps.py` پیکربندی کنند. با این حال، پیکربندی ویژگی‌ها کار نادری است که کاربر باید انجام دهد، زیرا اغلب اوقات پیکربندی پیش‌فرض برای کار با آن کافی است.

۴. models.py

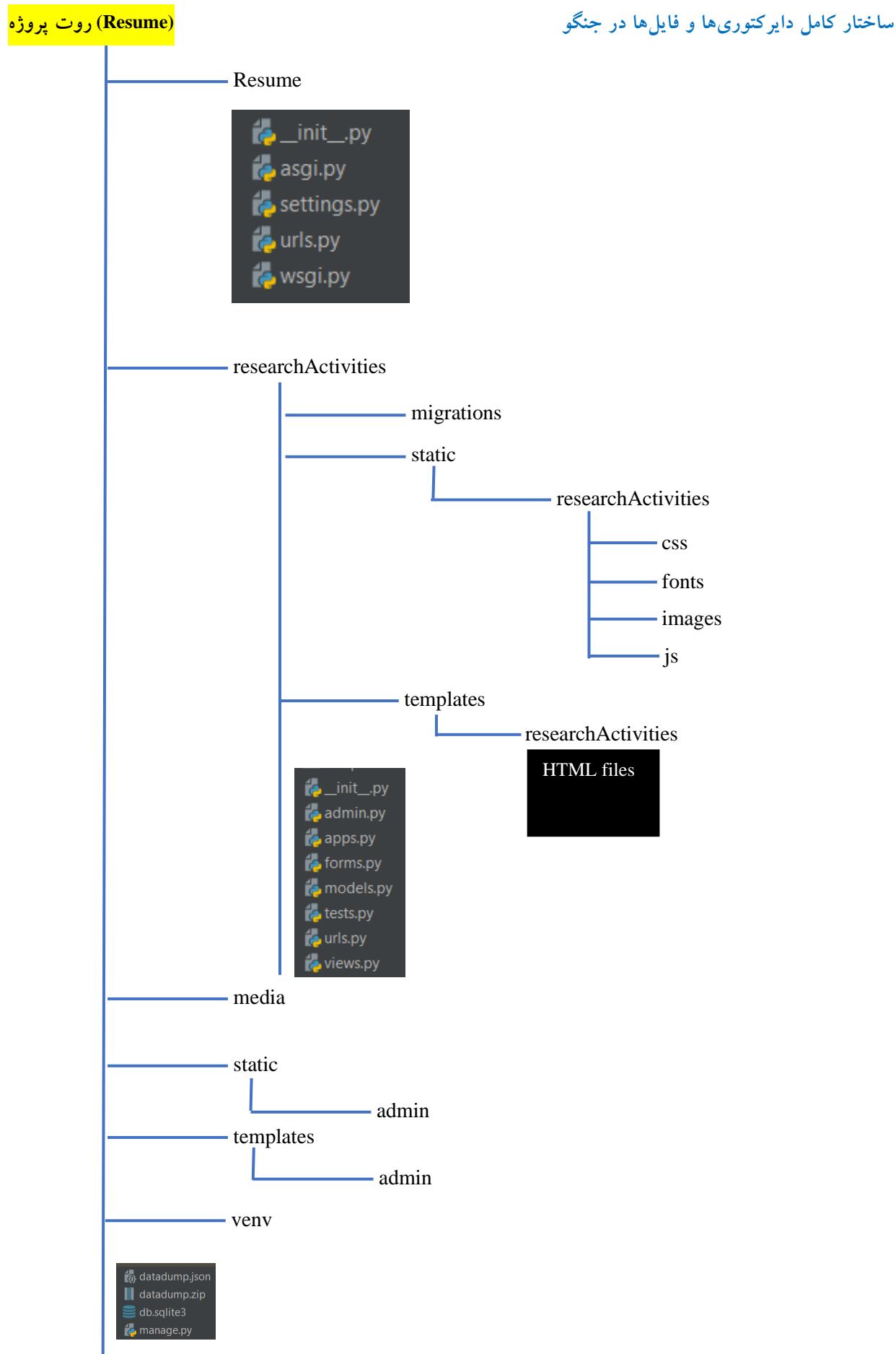
- ✓ مدل‌های برنامه‌های کاربردی وب را در قالب کلاس‌ها نشان می‌دهد.
- ✓ مدل‌ها ساختار پایگاه داده را تعریف می‌کنند.
- ✓ مدل‌ها طراحی واقعی مجموعه داده‌ها، روابط بین آنها و ویژگی‌های آنها را توصیف می‌کنند.

۵. views.py

در فریم‌ورک جنگو، `view` یا کلاس‌های پایتون هستند که یک درخواست وب را دریافت می‌کنند و یک پاسخ وب را برمی‌گردانند. پاسخ می‌تواند یک پاسخ ساده HTTP، یک پاسخ تولید کننده یک تمپلیت یا یک پاسخ از نوع `redirect` باشد که کاربر را به صفحه دیگری هدایت می‌کند. ویوها معمولاً در فایلی به نام `views.py` قرار می‌گیرند که در دایرکتوری App شما قرار دارد.

۶. urls.py

مانند `urls.py` در ساختار پروژه جنگو کار می‌کند. هدف اصلی، پیوند دادن درخواست URL کاربر به صفحات مربوطه‌ای است که به آنها اشاره می‌کند.



ORM

(Object Relational Mapping)

چیست؟ ORM

یک لایه مترجم بین زبان برنامه‌نویسی و پایگاه داده رابطه‌ای است که این دو به هم تبدیل می‌کند.



یک لایه شی گرا بین پایگاه‌های داده رابطه‌ای و زبان‌های برنامه‌نویسی شی گرا فراهم می‌کند. با کمک این لایه دیگر نیازی به نوشتن پرس‌وجوها با زبان SQL نیست. شما می‌توانید پرس‌وجوها را با استفاده از پارادایم شی گرایی زبان برنامه‌نویسی دلخواه خود بنویسید.

در شیوه‌های قدیمی یا سنتی برای دسترسی به اطلاعات DBMS پرس‌وجوها باید به شکل SQL نوشته می‌شدند ولی زمانی که از ORM استفاده می‌کنید نیازی نیست حتی دستورات SQL را بلد باشید کافی است بتوانید از ORM استفاده کنید تا داده‌ها را از دیتابیس بازیابی یا دستکاری کنید.

ابزارهای ORM بین توسعه‌دهندگان OOP محبوب هستند زیرا میزان دانش SQL مورد نیاز برای اتصال پایگاه داده به یک برنامه کاربردی را به حداقل می‌رسانند. ORM‌ها همچنین به طور خودکار کد SQL را تولید می‌کنند و به شما امکان می‌دهند بر روی تولید منطق کاری برنامه تمرکز کنید.

شروع پروژه جدید

(VS Code با)

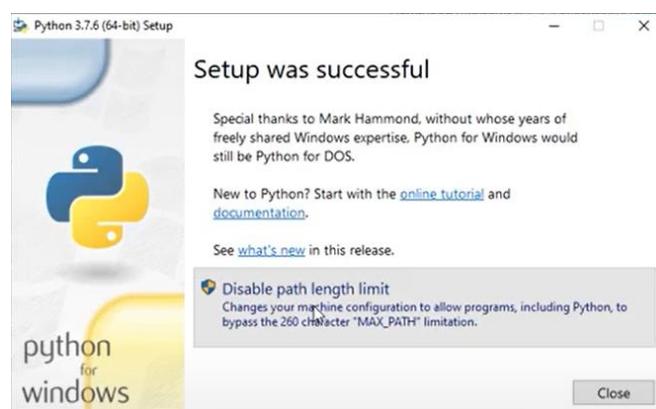
اگر از ویرایشگر VS Code استفاده می‌کنید، ابزارهای زیر را بترتیب نصب کنید:

نصب پایتون

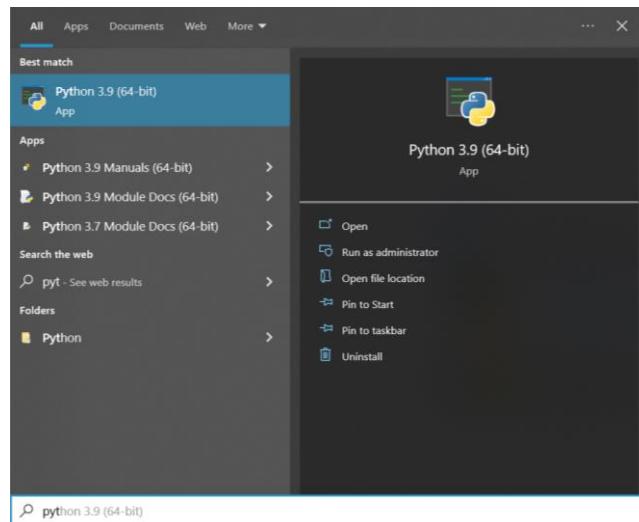
۱. دانلود پایتون از سایت: <https://www.python.org>
۲. نصب پایتون
۳. در مراحل نصب، تیک Add python to PATH را حتماً بزنید (اگر این تیک را نزنید بعداً نمی‌توانید در cmd پایتون را صدا بزنید یا پکیجی برای آن نصب کنید):



۴. بعد از نصب روی Disable path limit کلیک کنید تا در قسمت system environment محدودیت طول آدرس path را برطرف کند:



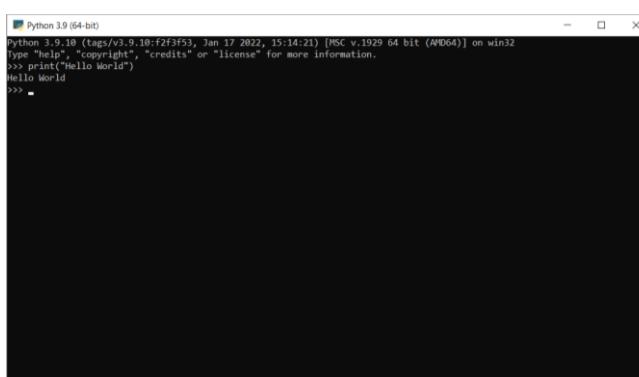
۵. برای تست: اگر سرج را بزنیم:



۶. اگر روی آن کلیک کنیم کنیم ترمینال پایتون باز می شود:



۷. برای تست درستی نصب، دستور print("Hello World") را وارد کنید:



۸. می توان داخل CMD هم با زدن python آنرا تست کرد:

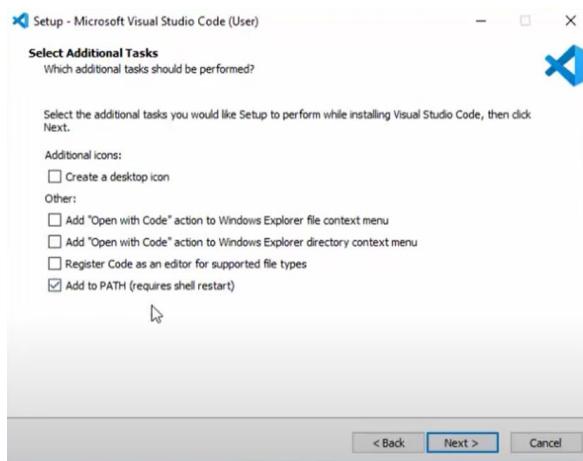


VS Code نصب

۱. دانلود VS Code از سایت: <https://code.visualstudio.com>

۲. نصب VS Code

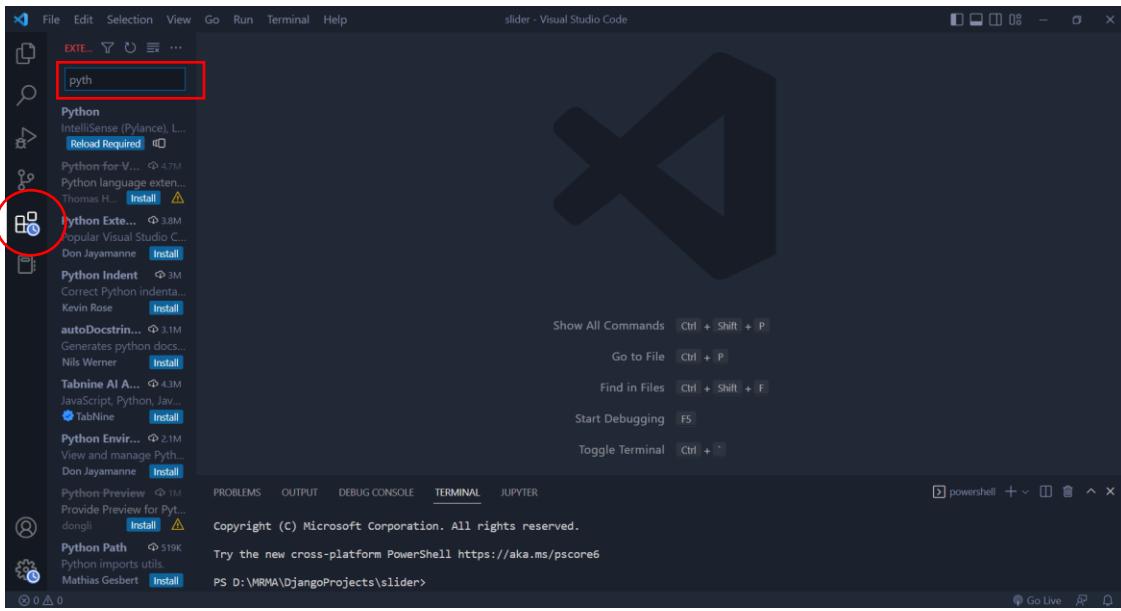
۳. در مرحله زیر، همه تیک‌ها را بزنید:



افزونه‌های موردنیاز در محیط VS Code

بعد از نصب Extension‌های موردنیاز را داخل VS Code اضافه کنیم:

۱. زبانی که قرار است در محیط VS Code با آن کار کنید را اضافه کنید. برای اینکار در بخش Python Extension را سرچ و آنرا نصب کنید:

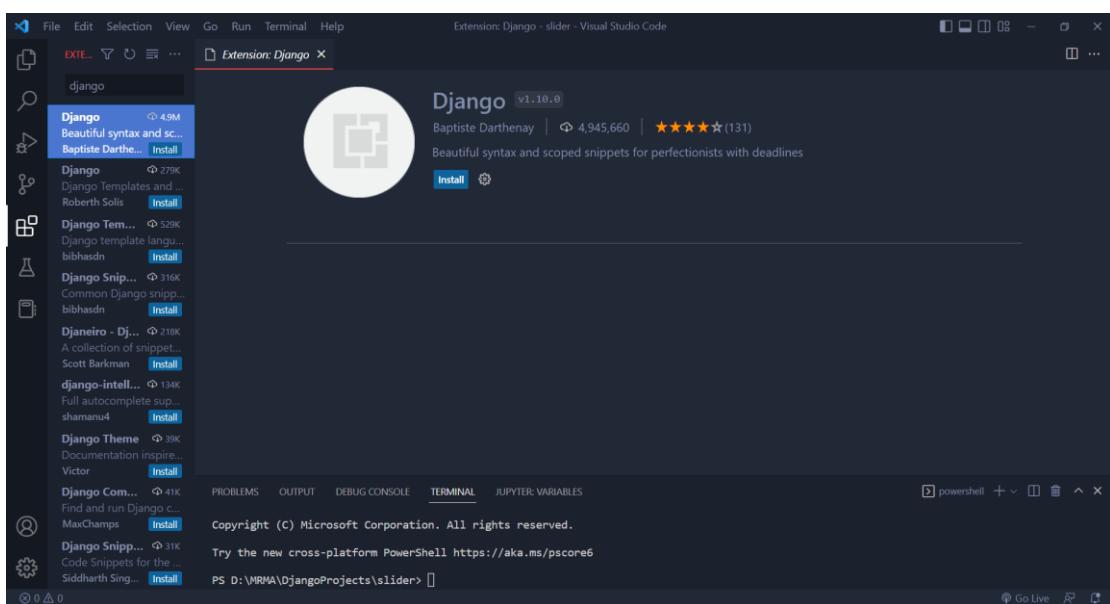


۲. افزونه Auto complete

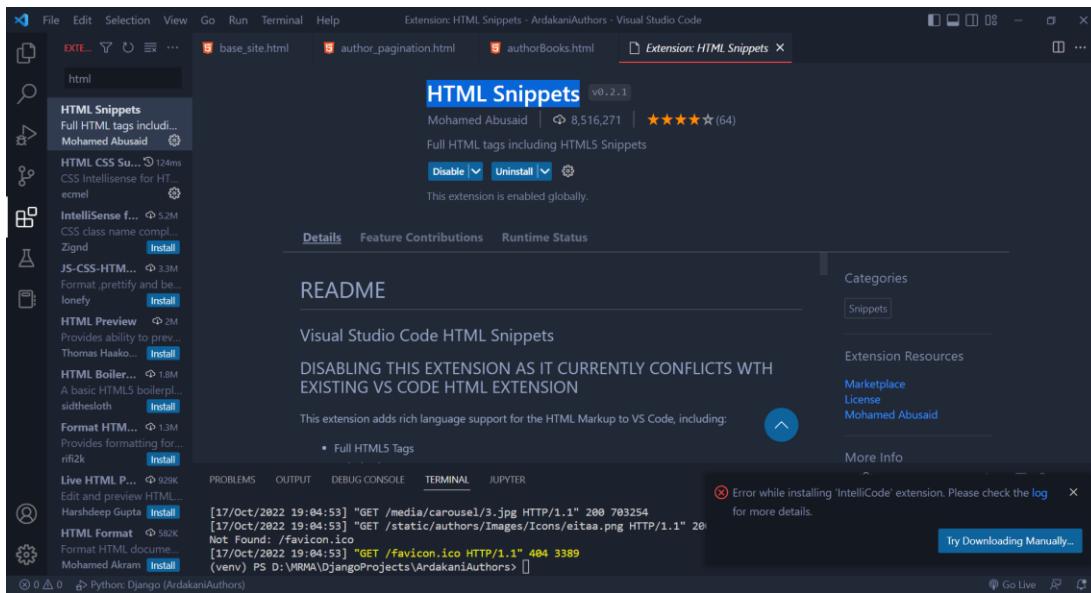
Tabnine AI Autocomplete for Javascript, Python, Typescr
یا

IntelliCode

۳. افزونه django



۴. افزونه HTML Snippets



ایجاد محیط مجازی (اگر از VS Code IDE استفاده می‌کنید)

(۱) نصب پکیج virtualenv

پکیج `virtualenv` ابزاری برای ایجاد محیط‌های ایزووله در پایتون است. از نسخه پایتون ۳.۳، این پکیج در کتابخانه استاندارد پایتون با نام `venv` ادغام شده است. پس اگر از نسخه پایتون ۳.۳ به بالا استفاده می‌کنید، نیازی به نصب چیزی ندارید. اما اگر نسخه پایتون شما پایین‌تر از ۳.۳ است با دستورات زیر می‌توانید این پکیج را نصب کنید:

```
py -m pip install --user virtualenv
```

(۲) ایجاد دایرکتوری پروژه (در مثال ما دایرکتوری `Resume`)

(۳) ایجاد یک محیط مجازی: برای ایجاد یک محیط مجازی در پایتون، در `cmd`، به دایرکتوری پروژه خود بروید (ساده‌ترین روش، رفتن به محل دایرکتوری . از آنجا در قسمت نوار آدرس، `cmd` را تایپ کنید) و کتابخانه `venv` را از مفسر پایتون صدا بزنید:

```
D:\MRMA\Presentation\Ardakan\Django\Resume> py -m venv my_env
```

(۴) فعال کردن محیط مجازی در پایتون

بعد از ایجاد کردن محیط مجازی باید آن را فعال کنید. قبل از شروع نصب پکیج‌ها یا استفاده از پکیج‌ها در محیط مجازی، باید محیط مجازی را فعال کنید. فعال کردن یک محیط مجازی، فایل‌های اجرایی پایتون و `pip` مخصوص محیط مجازی را در `PATH` پوسته شما قرار می‌دهد.

با دستور زیر می‌توانید محیط مجازی را فعال کنید:

```
D:\MRMA\Presentation\Ardakan\Django\Resume>.my_env\Scripts\activate
```

بعد از فعال شدن محیط مجازی، `prompt` شما به شکل زیر خواهد بود:

```
(my_env) D:\MRMA\Presentation\Ardakan\Django\Resume>
```

تا زمانی که محیط مجازی شما فعال باشد، `pip` پکیج‌ها را در آن محیط خاص نصب می‌کند و شما می‌توانید پکیج‌ها را در برنامه پایتون خود وارد کرده و از آنها استفاده کنید.

✓ با دستور pip freeze می توانیم پکیج های نصب شده در محیط را مشاهده کنیم.

غیرفعال کردن محیط مجازی در پایتون

اگر می خواهید پروژه را تغییر دهید یا محیط مجازی خود را ترک کنید، به سادگی اجرا کنید:

(my_env) D:\MRMA\Presentation\Ardakan\Django\Resume>Deactivate

اگر می خواهید دوباره وارد محیط مجازی شوید، کافی است همان دستور العمل های بالا را در مورد فعال سازی یک محیط مجازی دنبال کنید. نیازی به ایجاد مجدد محیط مجازی نیست.

نصب جنگو

(my_env) D:\MRMA\Presentation\Ardakan\Django\Resume> pip install django

شروع اولین پروژه جنگو

۱) اجرای فایل django-admin از پوشه env/scripts

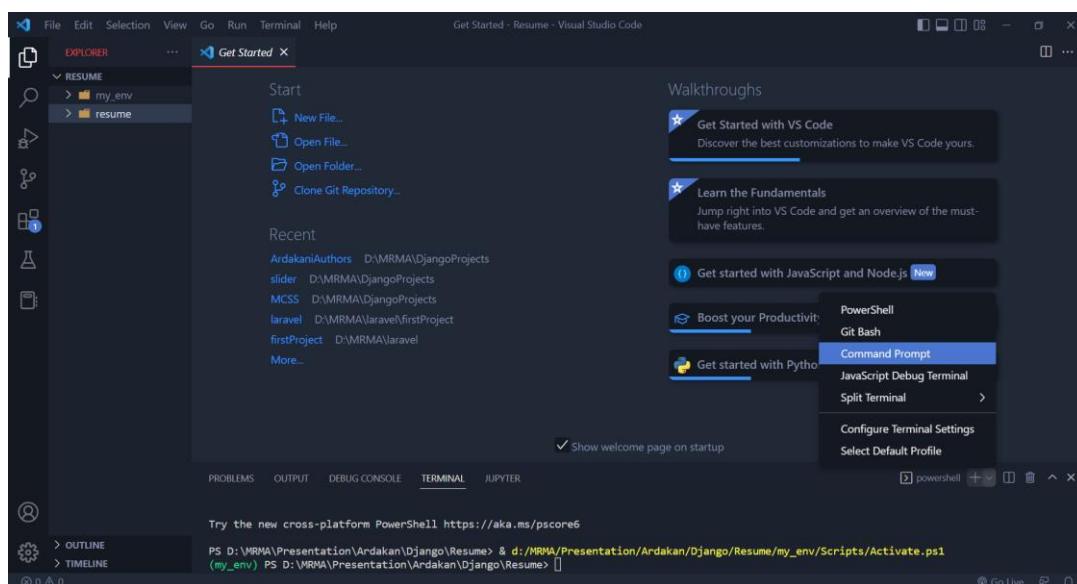
(my_env) D:\MRMA\Presentation\Ardakan\Django\Resume>my_env\scripts\django-admin startproject resume

۲) باز کردن پوشه پروژه داخل VS Code

۳) اگر از منوی Terminal گزینه New Terminal را انتخاب کنیم می بینیم که داخل محیط مجازی نیستیم. برای رفع مشکل:

a. از منوی View گزینه Command Palette را انتخاب کنید و سپس گزینه Python: Select Interpreter را انتخاب و در نهایت پایتون داخل محیط مجازی خود را انتخاب نمایید.

b. وقتی از منوی Terminal گزینه New Terminal را انتخاب کردید همانند تصویر زیر حتما Command Prompt را انتخاب کنید:



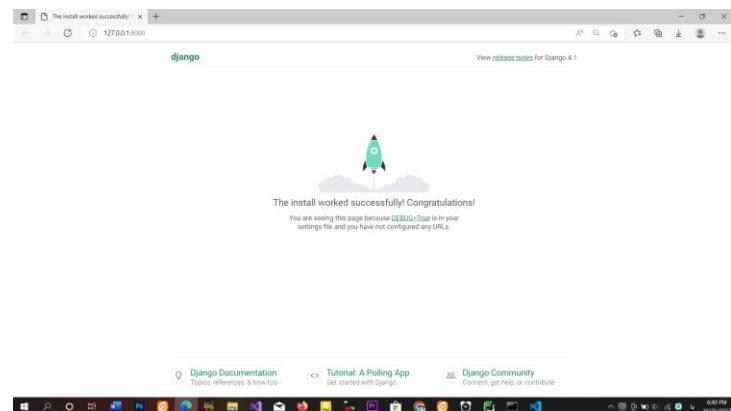
۴) برای مشاهده اولین صفحه وب اپلیکیشن جنگو، فایل manage.py داخل دایرکتوری پروژه را اجرا می کنیم. برای اینکار داخل ترمینال، ابتدا وارد دایرکتوری پروژه شده و دستور زیر را اجرا می کنیم:

(my_env) D:\MRMA\Presentation\Ardakan\Django\Resume>cd resume

آموزش پروژه محور جنگو

(my_env) D:\MRMA\Presentation\Ardakan\Django\Resume>python manage.py runserver

۵) با کلیک روی لینک سرور کوکال، اولین صفحه وب اپلیکیشن با توجه به پیش فرض های جنگو ظاهر می شود:



شروع پروژه جدید

(Pycharm با)

نصب پای چارم

لینک دانلود پای چارم:

<https://p3.download.ir/fa/entry//۹۸۸۶۷>

راهنمای نصب

1- ابتدا نرم افزار را نصب کنید.

2- از درون Crack پوشه ja-netfilter را در درایو C کپی کنید. (آدرس C:\)

3- برای کرک نرم افزار بایستی vmoptions پیکربندی شود. این کار را می توانید به دو صورت انجام دهید:

روش اول (بصورت خودکار): در مسیر install-current-user.vbs فایل ja-netfilter\scripts را اجرا کنید.

روش دوم (بصورت دستی): به محل نصب نرم افزار (مسیر پیش فرض C:\Program Files\JetBrains\PyCharm 2022.2\bin) مراجعه کنید و فایل goland64.exe.vmoptions را بوسیله برنامه Notepad ویندوز باز کنید. خط زیر را در انتهای متن ها (در خط جدید) وارد نموده و تغییرات را save کنید:

-javaagent:C:\ja-netfilter\ja-netfilter.jar=jetbrains

4- نرم افزار را اجرا کنید. پنجره License باز می شود.

4-1 گزینه Active PyCharm را و سپس Activation Code را انتخاب کنید.

4-2 از پوشه Crack فایل Activation Code.txt را باز کنید. متن درون آن را در گادر وارد نموده و بر روی Activate کلیک کنید.

5- نرم افزار فعال شده و بدون هیچ محدودیتی قابل استفاده است.

نکات:

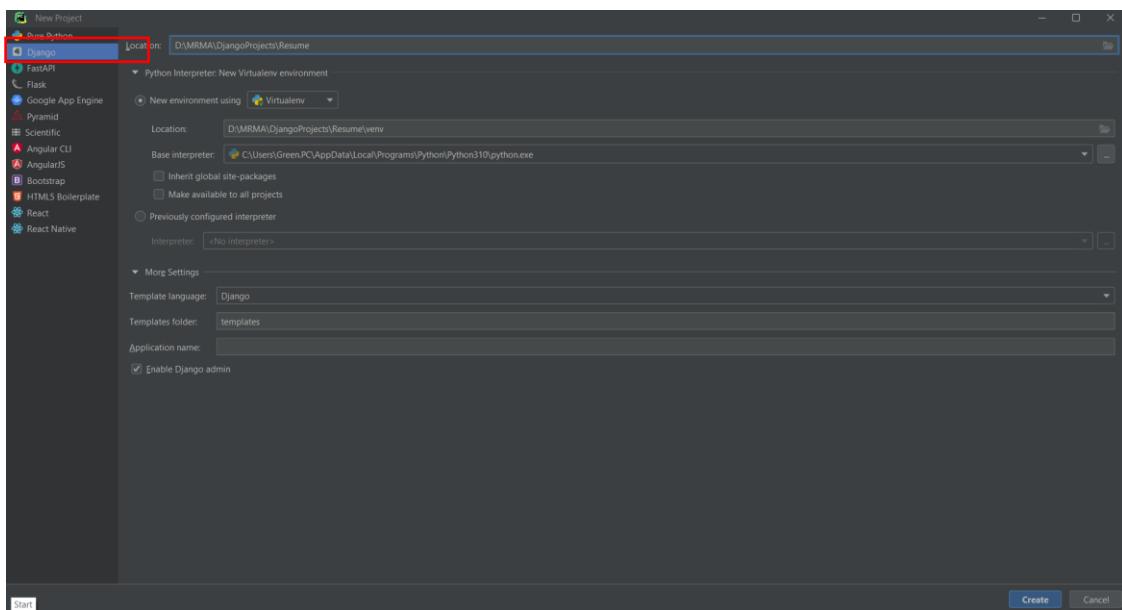
- در هر زمان فقط از یک javaagent می توان استفاده نمود (برای اطلاعات بیشتر فایل Read Me.txt را مطالعه کنید).

- نرم افزار به روش بالا نصب شده است و به درستی اجرا می شود.

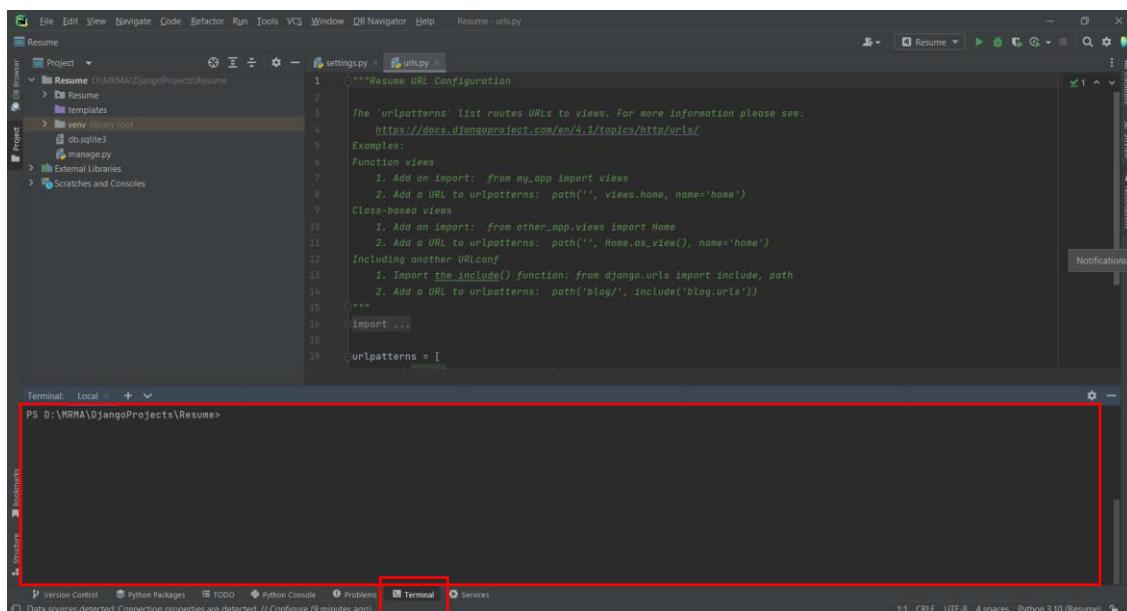
-ممکن است مجبور شوید هر بار برای اجرای نرم افزار مرحله 4 را تکرار کنید.

ایجاد پروژه با پایی چارم

(۱) بعد از انتخاب New Project، تنظیمات زیر را اعمال کنید: (دقت شود گزینه دوم sidebar یعنی django انتخاب شده باشد):



(۲) بعد از ایجاد پروژه، وارد ترمینال پایی چارم شوید:



(۳) در خط فرمان ترمینال، با دستور زیر، محیط مجازی را فعال نمایید:

PS D:\MRMA\ DjangoProjects\Resume> venv\Scripts\activate

با فعال شدن ترمینال، ابتدای خط فرمان (venv) ظاهر می شود:

(venv) D:\MRMA\ DjangoProjects\Resume>

نکته ۱: اگر (venv) ابتدای خط فرمان ترمینال نبود، با مراحل زیر ترمینال پیش فرض را از Power shell به CMD تغییر دهید:

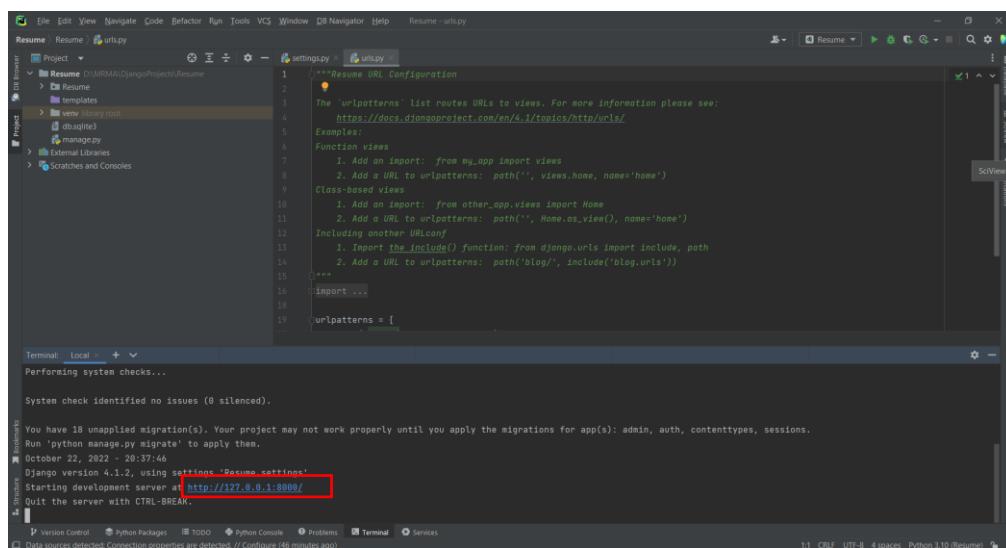
- 1) Open pycharm --> Go to Settings --> Tools --> Terminal
- 2) Change the Shell Path to C:\Windows\system32\cmd.exe from PS
- 3) Check the Activate virtualenv checkbox
- 4) Hit apply and open new terminal

نکته ۲: داخل ترمینال، با دستور pip freeze می‌توان دید که جنگو و برخی نیازمندی‌های دیگر در venv نصب شده است.

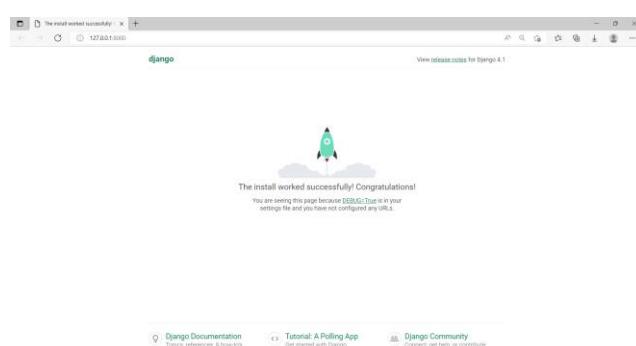
۴) هنگامی که یک پروژه جنگو جدید را راهاندازی کردید، می‌توانید کار با سرور توسعه داخلی جنگو را شروع کنید. سرور توسعه جنگو، یک سرور توسعه لوکال فوق العاده را ارائه می‌دهد که توسعه‌دهنگان می‌توانند از آن برای پیش‌نمایش به روزرسانی کد و آزمایش ویژگی‌های وب‌سایت جنگویی استفاده کنند. دستور زیر را در ترمینال اجرا کنید (دقت شود داخل دایرکتوری پروژه باشد):

(venv) D:\MRMA\ DjangoProjects\Resume>python .\manage.py runserver

دستور runserver یک دستور فرعی داخلی از فایل manage.py جنگو است که یک سرور توسعه را برای این پروژه خاص جنگو راهاندازی می‌کند.



۵) با کلیک روی لینک سرور کوکال، اولین صفحه وب‌اپلیکیشن با توجه به پیش‌فرض‌های جنگو ظاهر می‌شود:



ایجاد اولین App و مدل در جنگو

۱. اجرای دستور زیر در ترمینال:

```
python manage.py startapp researchActivities
```

نکته: نام‌گذاری App‌ها

- با حروف کوچک
- بدون _
- بصورت جمع

۲. افزودن اپ ایجاد شده به مجموعه اپ‌های جنگو

در فایل settings.py، اپ را به INSTALLED_APPS اضافه کنید:

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'researchActivities',
]
```

ایجاد مدل‌ها

۱) در دایرکتوری اپ ایجاد شده، داخل فایل models.py مدل‌ها را می‌سازیم. بازای هر مدل یک کلاس ایجاد می‌کنیم:

نکته: نام‌گذاری مدل‌ها:

- با حروف کوچک، حرف اول بزرگ
- بصورت مفرد

همه کلاس‌های تعریف شده از کلاس models.Model ارث‌بری می‌کنند.

```
class Paper(models.Model):
    CHOICES = [('0', 'داخلی ژورنال'),
               ('1', 'خارجی ژورنال'),
               ('2', 'داخلی کنفرانس'),
               ('3', 'خارجی کنفرانس')]
    paperType = models.CharField(max_length=1, choices=CHOICES)
    title = models.CharField(max_length=500)
    doi = models.CharField(max_length=1000)
    author1 = models.CharField(max_length=500)
    author2 = models.CharField(max_length=500)
    author3 = models.CharField(max_length=500)
    author4 = models.CharField(max_length=500)
    abstract = models.TextField()
    journal = models.CharField(max_length=500)
    issn = models.CharField(max_length=500)
    volume = models.CharField(max_length=500)
    publication_date = models.DateField()
    rank = models.CharField(max_length=500)

    def __str__(self):
        return self.title
```

تابع استرینگ کلاس (`__str__(self)`)، می‌شود تا ویژگی مدنظر از کلاس برگردانده شود. این متدهای نمایش رشته‌ای شی را برمی‌گرداند. متدهای `__str__(self)` هر زمان که `(str()` را روی یک شی فراخوانی کنید فراخوانی می‌شود. جنگو در تعدادی از مکان‌ها از `str(obj)` استفاده می‌کند. مهمتر از همه، نمایش یک شی در سایت مدیریت جنگو و نیز به عنوان مقدار درج شده در تمپلیت هنگام نمایش یک شی. بنابراین، همیشه باید یک نمایش خوب و قابل خواندن را برای انسان از مدل از طریق `__str__(self)` برگردانید. غالباً این دستور، فیلد عنوان یا نام را از مدل بازمی‌گرداند.

نکته: اگر بخواهیم بیش از یک فیلد توسط تابع استرینگ کلاس برگردانده شود:

```
class Person(models.Model):
    first_name = models.CharField(max_length=50)
    last_name = models.CharField(max_length=50)

    def __str__(self):
        return "first_name:{} last_name:{}".format(first_name, last_name)
        #return '%s %s' % (self.first_name, self.last_name)
```

(۲) بترتیب با دستورات زیر فایل حاوی تغییرات داخل مدل‌ها ایجاد و به پایگاه داده اعمال می‌شوند:

- 1) `python manage.py makemigrations`
- 2) `python manage.py migrate`

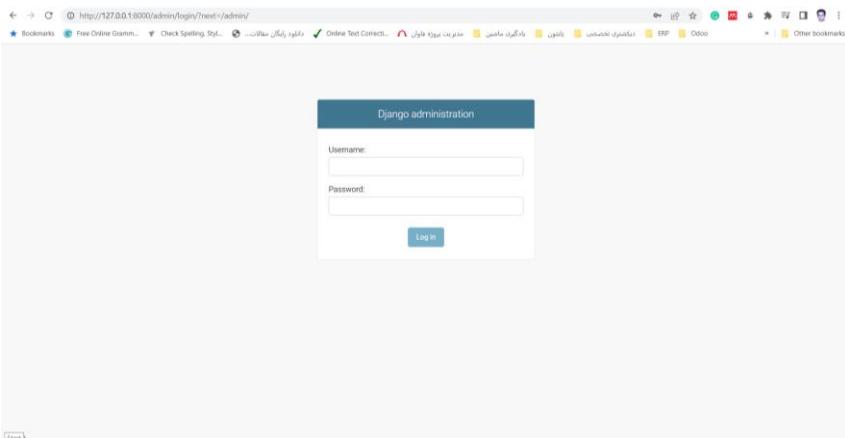
فایل‌های مهاجرتی، حاوی تغییراتی را ایجاد می‌کند که باید در پایگاه داده اعمال شوند، اما در واقع هیچ چیزی را در پایگاه داده شما تغییر نمی‌دهد.

تغییرات واقعی را در پایگاه داده شما بر اساس فایل‌های مهاجرت اعمال می‌کند. ✓

بخش ادمین پروژه

با تایپ دستور زیر در مرورگر می‌توانید وارد بخش ادمین شوید:

<http://127.0.0.1:8000/admin/>



- درخواست یوزرنیم و پسورد می‌دهد. باید ادمینی را تعریف کنید. داخل ترمینال دستور زیر را وارد کنید:

python manage.py createsuperuser

✓ یوزرنیم، آدرس ایمیل و پسورد را وارد کنید. حال می‌توانید با سوپریوزر ایجاد شده وارد بخش ادمین شوید.

• برای فارسی کردن محیط ادمین:

✓ در فایل settings.py در بخش LANGUAGE_CODE بجای fa-ir، en-us عبارت UTC بجای TIME_ZONE

عبارت Asia/Tehran را قرار دهید.

• برای افزودن مدل‌های خودتان به پنل ادمین:

✓ در فایل admin.py داخل دایرکتوری آپ ایجاد شده، مدل‌ها را ریجستر کنید تا در پنل ادمین قابل مشاهده باشند:

▪ ابتدا مدل‌ها را به فایل admin.py ایمپورت کنید:

```
from researchActivities.models import Paper
```

▪ با دستور زیر مدل‌ها را ریجستر کنید:

```
admin.site.register(Paper)
```

▪ اگر مجدد سرور را ران بگیرید مدل ما داخل پنل ادمین ظاهر می‌شود.

• برای فارسی کردن مدل‌ها و فیلدات آن داخل پنل ادمین:

▪ داخل فایل app.py برای آپ، verbose name تعريف کنید:

```
class ResearchactivitiesConfig(AppConfig):
    default_auto_field = 'django.db.models.BigAutoField'
    name = 'researchActivities'
    verbose_name = 'من پژوهشی های فعالیت'
```

▪ داخل models.py برای مدل‌ها و فیلدات مدل، verbose name تعريف کنید:

```

class Paper(models.Model):
    class Meta:
        verbose_name = 'مقاله'
        verbose_name_plural = 'مقالات'
    CHOICES = [('0', 'داخلی ژورنال'),
               ('1', 'خارجی ژورنال'),
               ('2', 'خارجی کنفرانس'),
               ('3', 'خارجی کنفرانس')]
    paperType = models.CharField(max_length=1, choices=CHOICES, verbose_name='نوع')
    title = models.CharField(max_length=500, verbose_name='مقاله عنوان')
    doi = models.CharField(max_length=1000)
    author1 = models.CharField(max_length=500, verbose_name='اول نویسنده')
    author2 = models.CharField(max_length=500, verbose_name='دوم نویسنده')
    author3 = models.CharField(max_length=500, verbose_name='سوم نویسنده')
    author4 = models.CharField(max_length=500, verbose_name='چهارم نویسنده')
    abstract = models.TextField(verbose_name='مقاله چکیده')
    journal = models.CharField(max_length=500, verbose_name='ژورنال نام')
    issn = models.CharField(max_length=500)
    volume = models.CharField(max_length=500, verbose_name='ژورنال شمارگان')
    publication_date = models.DateField(verbose_name='انتشار تاریخ')
    rank = models.CharField(max_length=500, verbose_name='ژورنال رتبه')

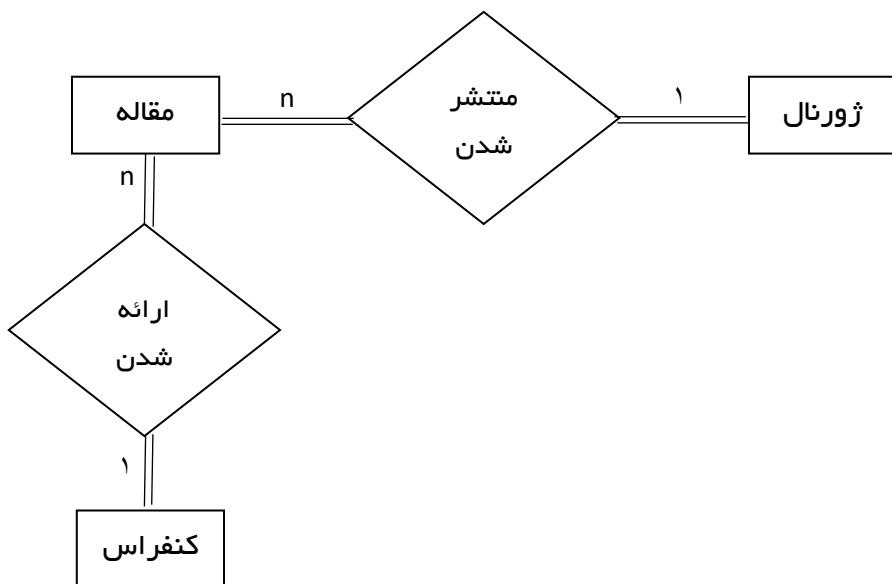
    def __str__(self):
        return self.title

```

ارتباط یک به چند در مدل‌ها

ارتباط یک به چند در مدل‌ها

فرض کنید می‌خواهید تک مدل ایجاد شده قبلی (Paper) را بصورت زیر ویرایش کنید:



بعد از ایجاد دو مدل به نام‌های Journal و Conference، فیلد‌های از نوع کلید خارجی (Foreign key) را به مدل Paper اضافه کنید:

```

class Paper(models.Model):
    class Meta:
        verbose_name = 'مقاله'
        verbose_name_plural = 'مقالات'
        title = models.CharField(max_length=500, verbose_name='عنوان')
        doi = models.CharField(max_length=1000)
        author1 = models.CharField(max_length=500, verbose_name='اول نویسنده')
        author2 = models.CharField(max_length=500, verbose_name='دوم نویسنده')
        author3 = models.CharField(max_length=500, verbose_name='سوم نویسنده')
        author4 = models.CharField(max_length=500, verbose_name='چهارم نویسنده')
        abstract = models.TextField(verbose_name='چکیده')
        publication_date = models.DateField(verbose_name='تاریخ انتشار')
        conference = models.ForeignKey(Conference, on_delete=models.PROTECT, null=True,
                                       verbose_name="کنفرانس")
        journal = models.ForeignKey(Journal, on_delete=models.PROTECT, null=True,
                                    verbose_name="ژورنال")

    def __str__(self):
        return self.title
  
```

نکه: بعد از اعمال هر تغییر در مدل‌ها، دو دستور makemigrations و migrate به ترتیب اجرا شوند

نکه: بعد از ایجاد هر مدل، آنها را در فایل admin.py登録 کنید تا در پنل مدیریت دیده شود:

```

from researchActivities.models import Paper, Conference, Journal
admin.site.register(Paper)
admin.site.register(Conference)
admin.site.register(Journal)
  
```

نکته: چنانچه به هر دلیلی فایل 0001_initial.py را از فolder migrations حذف کردید حتماً دیتابیس را یعنی فایل db.sqlite3 را هم حذف کنید تا جنگو تغییرات بعدی را که migrate می‌کنید متوجه شود.

نمایش فیلدهای مدنظر یک مدل در پنل ادمین

در فایل admin.py دستورات زیر را بنویسید:

```
from researchActivities.models import Paper, Conference, Journal

class PaperAdmin(admin.ModelAdmin):
    list_display = ("title", "doi", "author1", "author2", "publication_date",
    "conference", "journal", "paper_image")

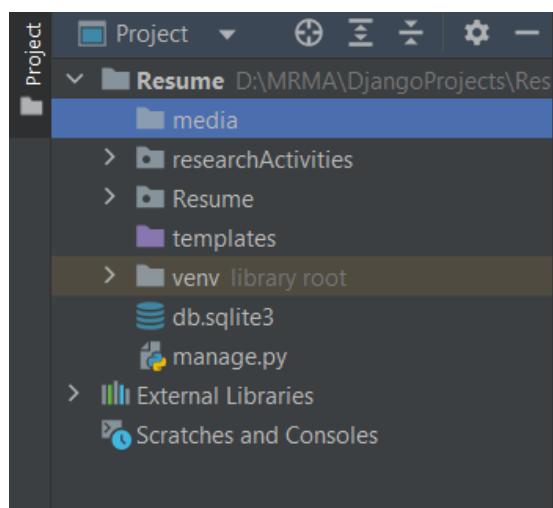
class ConferenceAdmin(admin.ModelAdmin):
    list_display = ("confTopic", "confType", "title", "event_date", "event_place")

class JournalAdmin(admin.ModelAdmin):
    list_display = ("journalType", "title", "issn", "volume", "publisher", "rank")

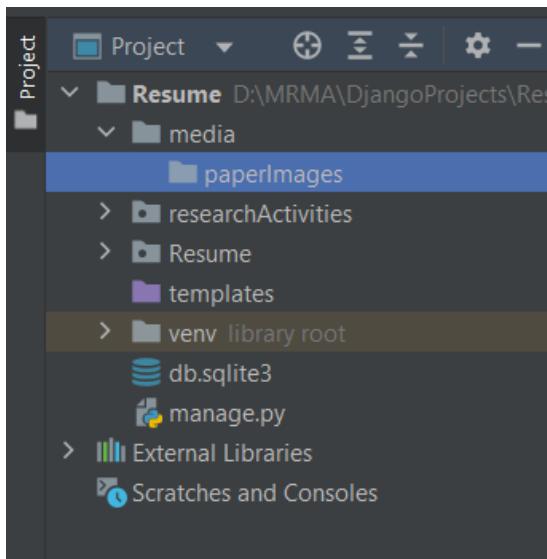
admin.site.register(Paper, PaperAdmin)
admin.site.register(Conference, ConferenceAdmin)
admin.site.register(Journal, JournalAdmin)
```

کار با ویژگی تصاویر در پنل ادمین

۱. ایجاد دایرکتوری با نام media هم‌سطح با App manage.py یا App (در روت پروژه)



۲. ایجاد دایرکتوری مدنظر برای ذخیره مدياهای داخل دایرکتوری media



۳. تعریف مسیری برای میدیاها داخل فایل settings.py که جنگو بداند کجا قرار است میدیاها آپلود/خوانده شوند:

The screenshot shows the PyCharm code editor with the 'settings.py' file open. The code is as follows:

```

111
112 USE_I18N = True
113
114 USE_TZ = True
115
116
117 # Static files (CSS, JavaScript, Images)
118 # https://docs.djangoproject.com/en/4.1/howto/static-files/
119
120 STATIC_URL = 'static/'
121 MEDIA_URL = '/media/' ← به روت پروژه (BASE_DIR) دایرکتوری media اضافه کن
122 MEDIA_ROOT = os.path.join(BASE_DIR, 'media')
123
124 # Default primary key field type
125 # https://docs.djangoproject.com/en/4.1/ref/settings/#default-auto-field
126
127 DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'

```

A red box highlights the line 'MEDIA_ROOT = os.path.join(BASE_DIR, 'media')'. A red arrow points from the text 'به روت پروژه (BASE_DIR) دایرکتوری media اضافه کن' to this line.

نکته: در ابتدای فایل settings.py باید `os.path` ایمپورت شده باشد:

The screenshot shows the PyCharm code editor with the 'settings.py' file open. The code is as follows:

```

1 """
2 Django settings for Resume project.
3
4 Generated by 'django-admin startproject' using Django 4.1.2.
5
6 For more information on this file, see
7 https://docs.djangoproject.com/en/4.1/topics/settings/
8
9 For the full list of settings and their values, see
10 https://docs.djangoproject.com/en/4.1/ref/settings/
11
12 import os.path
13 from pathlib import Path
14
15 # Build paths inside the project like this: BASE_DIR / 'subdir'.
16 BASE_DIR = Path(__file__).resolve().parent.parent
17

```

A red box highlights the line 'import os.path'.

۴. هنگام اجرا کردن پروژه، نصب pillow را درخواست می‌کند.

```

Generated by 'django-admin startproject' using Django 4.1.2.

For more information on this file, see
https://docs.djangoproject.com/en/4.1/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/4.1/ref/settings/


import os.path
from pathlib import Path

# Build paths inside the project like this: BASE_DIR / 'subdir'.
BASE_DIR = Path(__file__).parent.parent

Terminal: C:\Windows\runserver + 
self.check(display_num_errors=True)
File "D:\MRA\ DjangoProjects\Resume\venv\lib\site-packages\django\core\management\base.py", line 546, in check
    raise SystemCheckError(msg)
django.core.management.base.SystemCheckError: System check identified some issues:

ERRORS:
researchActivities.Paper.paper_image: (fields.E210) Cannot use ImageField because Pillow is not installed.
    HINT: Get Pillow at https://pypi.org/project/Pillow/ or run command "python -m pip install Pillow".
System check identified 1 issue (0 silenced).

```

۵. با کلیدهای **ctrl+c** اجرا را قطع و سپس دستور `python -m pip install Pillow` را برای نصب از پنجره بالا کپی و در ترمینال اجرا کنید.

۶. ویژگی «تصویر مقاله» را به مدل Paper اضافه کنید:

```

class Paper(models.Model):
    class Meta:
        verbose_name = 'مقاله'
        verbose_name_plural = 'مقالات'
    title = models.CharField(max_length=500, verbose_name='مقاله عنوان')
    doi = models.CharField(max_length=1000)
    author1 = models.CharField(max_length=500, verbose_name='اول نویسنده')
    author2 = models.CharField(max_length=500, verbose_name='دوم نویسنده', blank=True)
    author3 = models.CharField(max_length=500, verbose_name='سوم نویسنده', blank=True)
    author4 = models.CharField(max_length=500, verbose_name='چهارم نویسنده', blank=True)
    abstract = models.TextField(verbose_name='مقاله چکیده')
    publication_date = models.DateField(verbose_name='انتشار تاریخ')
    paper_image = models.ImageField(upload_to="paperImages/", verbose_name='مقاله تصویر')
    conference = models.ForeignKey(Conference, on_delete=models.PROTECT, null=True,
                                   verbose_name="کنفرانس", blank=True)
    journal = models.ForeignKey(Journal, on_delete=models.PROTECT, null=True, verbose_name="زنگنه", blank=True)

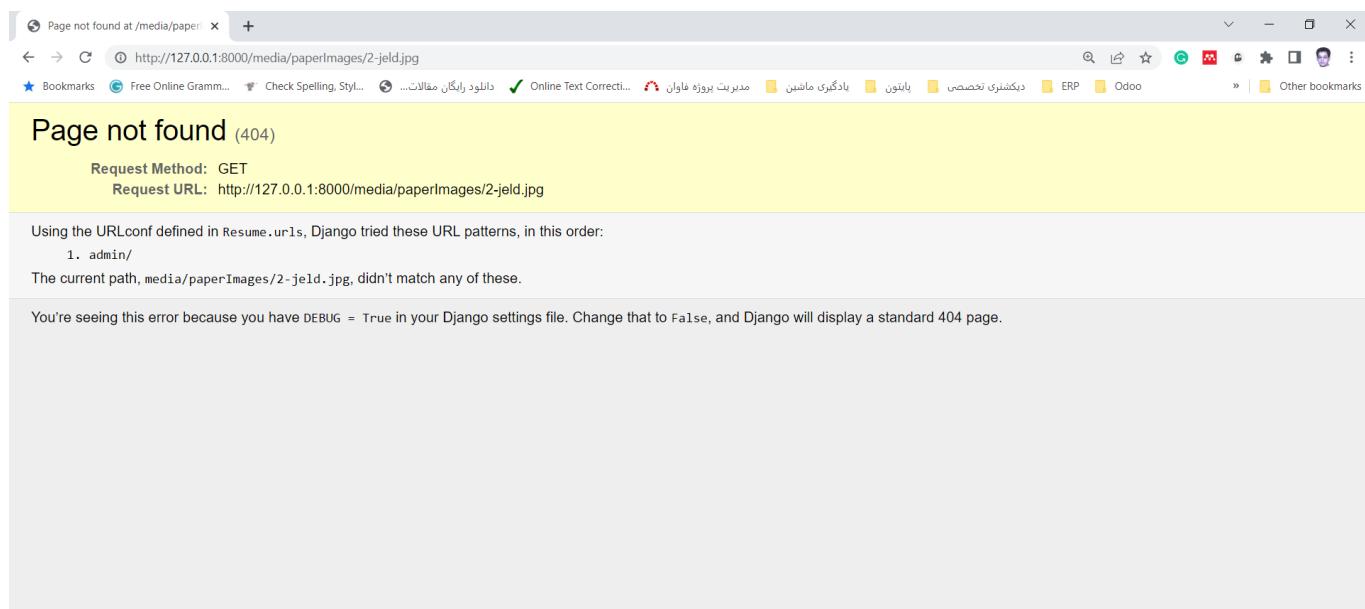
    def __str__(self):
        return self.title

```

۷. دستورات `makemigrations` و `migrate` را اجرا کنید.

۸. برنامه را اجرا کنید می‌بینید که ویژگی تصویر هم به مقاله اضافه شده است.

۹. اگر در پنل ادمین روی لینک تصویر کپی کنید خطایی را می‌بینید:



علت خطا: در فایل settings.py ، چون در حال توسعه وب سایت هستید پس debug=true است. کار سرو کردن فایل‌های مديا بر عهده سرور است نه بر عهده جنگو (جنگو فایل‌های استاتیک را سرو می‌کند و در نتیجه وقتی در حالت دیگر هم هستید اینگونه فایل‌ها سرو می‌شوند). الان در حال توسعه هستید و می‌خواهید در حالت Debug=true بمانید پس چکار کنید که این خطا ظاهر نشود:

وارد فایل urls.py شوید باید به جنگو بگویید وقتی در حالت debug=true هستید فایل‌های داخل فایل سیستم را برای شما سرو کند:

```
if settings.DEBUG:
    urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT)
```

نکته: اگر کلماتی قرمزرنگ شدند با نگه داشتن موس روی آنها، ایمپورت‌های لازم را انجام دهید.

مشاهده پایگاه داده سیستم و جداول آن

در پنجره ابزار پایگاه داده (View | Tool Windows | Database) می‌توانید با پایگاه‌های داده کار کنید. می‌توانید ساختارهای داده را در پایگاه داده خود مشاهده و اصلاح کنید و سایر وظایف مرتبط را انجام دهید. برای مشاهده جداول، روی جدول مدنظر دوبار کلیک کنید.

ویو در جنگ

ویو در جنگ

در الگوی MVT

۱. مدل همان کلاسها و متناظر آنها جداولی است که قبل ایجاد کردید.

۲. ویو، درخواست‌های کاربر را دریافت و پردازش می‌کند.

۳. و خروجی را به تmpliyت ارسال می‌نماید تا کاربر نتیجه را مشاهده کند.

چنانچه بخواهید درخواست‌های کاربرها را که بصورت URL در مرورگر وارد می‌کنند پاسخ دهید و کاربر را بسمت صفحه مورد نظرش هدایت نمایید باید:

- ابتدا مسیرهایی را که کاربر بصورت URL در مرورگر وارد می‌کند را مشخص کنید
- سپس برای آنها ویو نوشه
- و در نهایت تmpliyت بنویسید.

ایجاد ویو

۱. می‌خواهید وقتی کاربر یوآرال researchActivities/papers/list را در مرورگر وارد کرد لیست مقالات برای کاربر نمایش داده شود.

در فایل urls.py مسیر زیر را تعریف کنید:

```
urlpatterns = [
    path('admin/', admin.site.urls),
    path('researchActivities/papers/list', paperListView),
]
```

نام ویویی که قرار است یوآرال مقابلش را پردازش نماید

نکه: برای رفع خطاهایی که زیرخط قرمزرنگ دارند ایمپورت‌های مربوط را با نگه داشتن موس روی آنها، اعمال کنید.

۲. ویو مدنظر را داخل views.py ایجاد کنید:

```
def paperListView(request):
    return HttpResponse('من مقالات لیست')
```

نام ویو دقیقاً همان نامی است که در فایل urls.py مشخص کرده بودید.

در واقع در ویوی فوق گفته می‌شود وقتی درخواستی (آرگومان request) رسید، پاسخی (Response) را برگردان.

پاسخها در ویو از نوع HttpResponse هستند. در واقع زمانی از HttpResponse استفاده می‌شود که بخواهیم خروجی‌ای را غیر از مسیر تmpliyت برای کاربر

برگردانیم استفاده می‌شود.

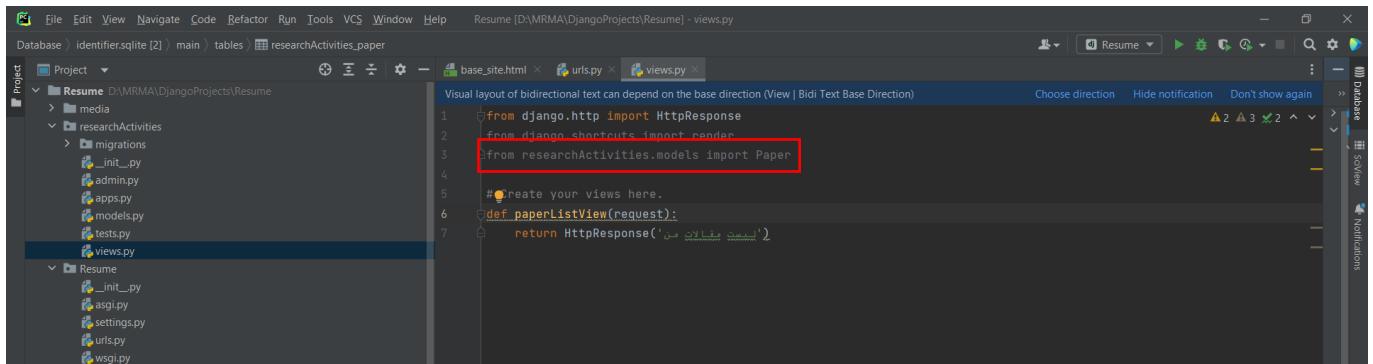
۳. برنامه را اجرا و یوآرال زیر را وارد کنید:

<http://127.0.0.1:8000/researchActivities/papers/list>

۴. برنامه بدرستی اجرا می‌شود:



۵. برای مشاهده لیست مقالات، ابتدا داخل ویو، مدل مدنظر (Paper) را ایمپورت کنید:

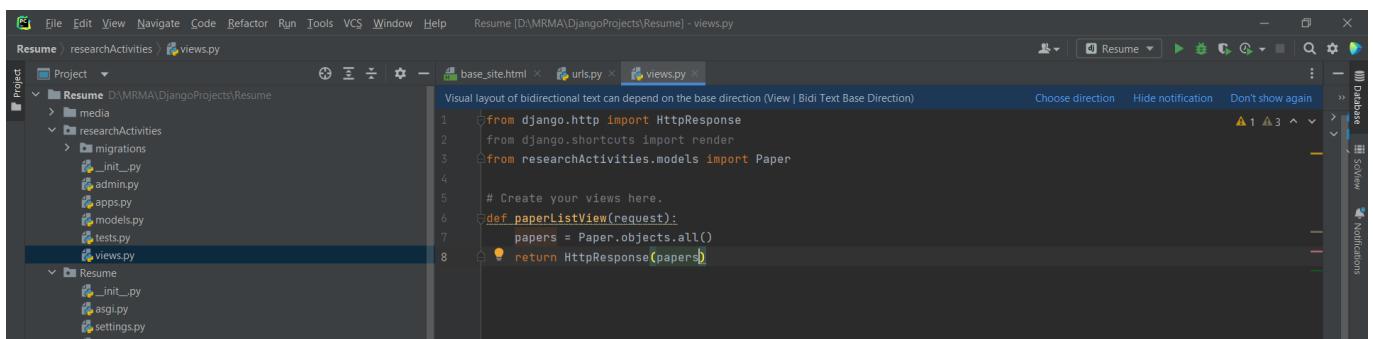


```

File Edit View Navigate Code Refactor Run Tools VCS Window Help
Resume [D:\MRMA\ DjangoProjects\Resume] - views.py
Database > identifier.sqlite [2] > main > tables > researchActivities_paper
Project > Resume > media > migrations > init_.py > admin.py > apps.py > models.py > tests.py > views.py
base_site.html > urls.py > views.py
Visual layout of bidirectional text can depend on the base direction (View | Bidi Text Base Direction)
Choose direction Hide notification Don't show again
1 from django.http import HttpResponseRedirect
2 from django.shortcuts import render
3 from researchActivities.models import Paper
4
5 # Create your views here.
6 def paperListView(request):
7     return HttpResponseRedirect('list of papers')

```

۶. ویوی paperListView را بصورت زیر کامل کنید:



```

File Edit View Navigate Code Refactor Run Tools VCS Window Help
Resume [D:\MRMA\ DjangoProjects\Resume] - views.py
Project > Resume > media > migrations > init_.py > admin.py > apps.py > models.py > tests.py > views.py
base_site.html > urls.py > views.py
Visual layout of bidirectional text can depend on the base direction (View | Bidi Text Base Direction)
Choose direction Hide notification Don't show again
1 from django.http import HttpResponseRedirect
2 from django.shortcuts import render
3 from researchActivities.models import Paper
4
5 # Create your views here.
6 def paperListView(request):
7     papers = Paper.objects.all()
8     return HttpResponseRedirect(papers)

```

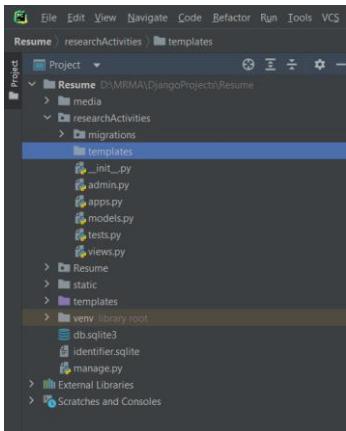
۷. برنامه را اجرا کنید. لیست مقالات ظاهر می شوند:



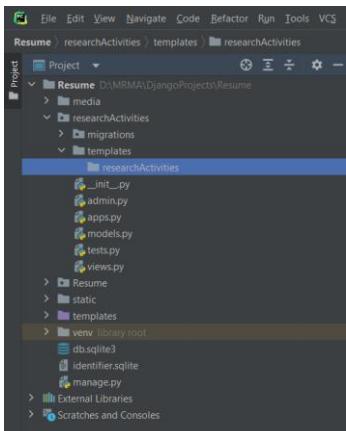
تمپلیت در جنگو

ایجاد تمپلیت

۱. ابتدا در دایرکتوری آپ مدنظر، ساپ دایرکتوری با نام `templates` ایجاد کنید:

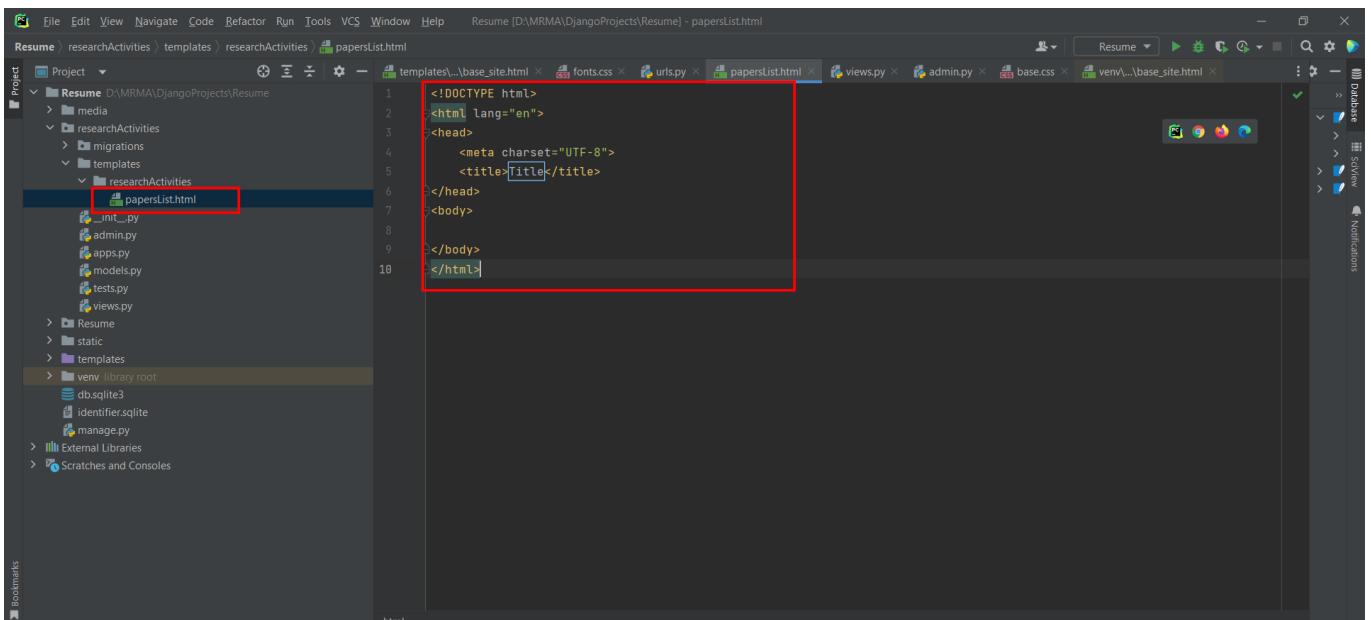


۲. داخل دایرکتوری `templates`، یک دایرکتوری همنام با آپ ایجاد کنید:

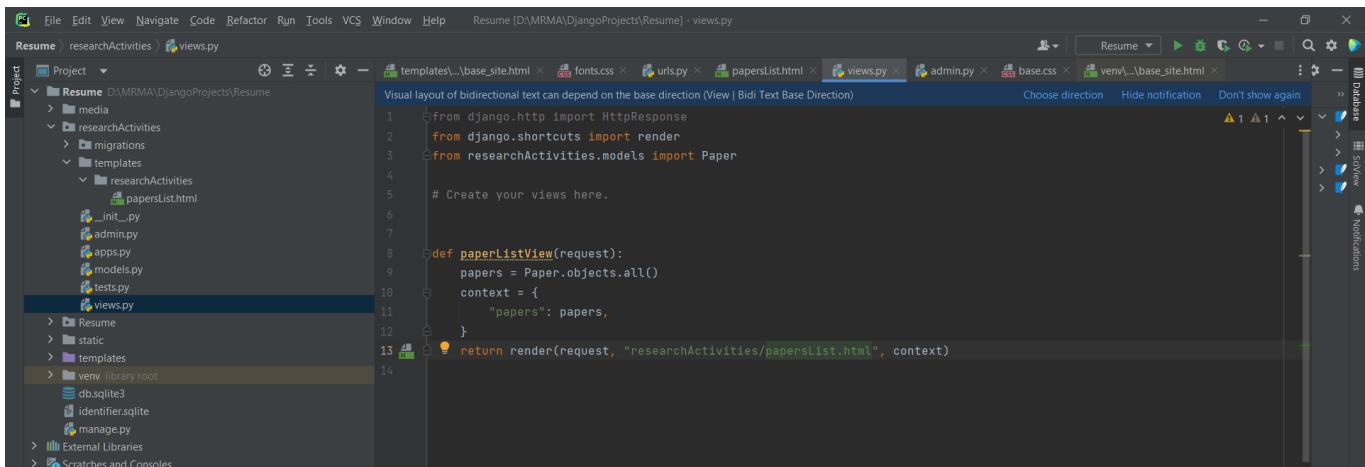


۳. برای ایجاد تمپلیت، داخل ساپ دایرکتوری `researchActivities`، یک فایل `html` با نامی مناسب با عملکردش ایجاد می‌کنیم

:(`papersList.html`)



۴. به ویوی paperListView برمی‌گردیم تا به ویو گفته شود تمپلیت فوق یعنی papersList.html را بصورت زیر تغییر می‌دهیم:



```

File Edit View Navigate Code Refactor Run Tools VCS Window Help Resume [D:\MRMA\ DjangoProjects\Resume] - views.py
Resume > researchActivities > Views.py
Project > Resume D:\MRMA\ DjangoProjects\Resume
  > media
  > researchActivities
    > migrations
    > templates
      > researchActivities
        > papersList.html
        > __init__.py
        > admin.py
        > apps.py
        > models.py
        > tests.py
        > views.py
  > Resume
  > static
  > templates
  > venv library root
    > db.sqlite3
    > identifiers.sqlite
    > manage.py
> External Libraries
> Scratches and Consoles

Visual layout of bidirectional text can depend on the base direction (View | Bidi Text Base Direction)
1 from django.http import HttpResponseRedirect
2 from django.shortcuts import render
3 from researchActivities.models import Paper
4
5 # Create your views here.
6
7
8 def paperListView(request):
9     papers = Paper.objects.all()
10    context = {
11        "papers": papers,
12    }
13    return render(request, "researchActivities/papersList.html", context)
14

```

سه آرگومان دارد:

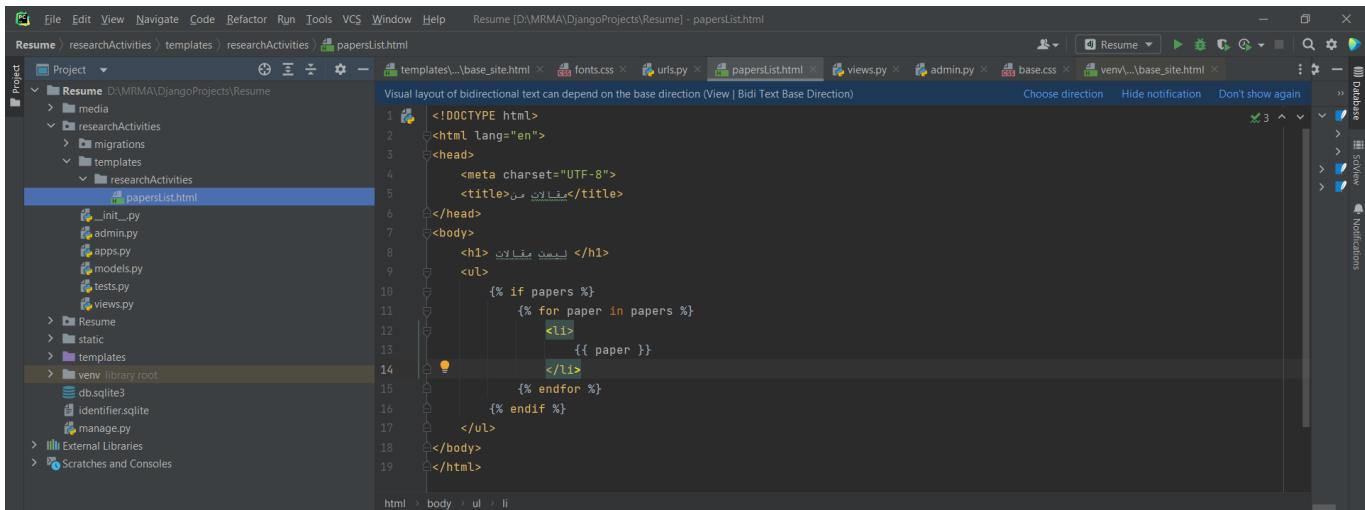
render (request, template, context)

- همان request که آرگومان ویو بود

- نام تمپلیت مدنظر بصورت استرینگ Template

- محتوایی که قرار است به کاربر نشان داده شود. context بصورت یک دیکشنری با زوج کلید-مقدار تولید می‌شود.

۵. به تمپلیت papersList.html برمی‌گردیم و آنرا بدلخواه طراحی می‌کنیم:



```

File Edit View Navigate Code Refactor Run Tools VCS Window Help Resume [D:\MRMA\ DjangoProjects\Resume] - papersList.html
Resume > researchActivities > templates > researchActivities > papersList.html
Project > Resume D:\MRMA\ DjangoProjects\Resume
  > media
  > researchActivities
    > migrations
    > templates
      > researchActivities
        > __init__.py
        > admin.py
        > apps.py
        > models.py
        > tests.py
        > views.py
  > Resume
  > static
  > templates
  > venv library root
    > db.sqlite3
    > identifiers.sqlite
    > manage.py
> External Libraries
> Scratches and Consoles

Visual layout of bidirectional text can depend on the base direction (View | Bidi Text Base Direction)
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="UTF-8">
5     <title>بیوگرافی</title>
6   </head>
7   <body>
8     <h1>لیست مقالات</h1>
9     <ul>
10    {% if papers %}
11      {% for paper in papers %}
12        <li>
13          {{ paper }}
14        </li>
15      {% endfor %}
16    {% endif %}
17    </ul>
18  </body>
19</html>

```

نکته: برای نوشتن دستورات داخل تمپلیت html، از {{ }} و برای نمایش مقادیر از {{ }} استفاده می‌شود.

۶. پروژه را اجرا می‌کنیم خروجی زیر را خواهیم داشت:



لیست مقالات

- جالش‌های امنیتی جدید در حوزه بانکداری الکترونیک
- نشر فناوری اطلاعات در جلوگیری از انتشار کرونا وی

۷. در ادامه نمایش را راست به چپ (داخل تمپلیت، rtl را اضافه می کنیم)، و تعداد مقالات را هم نمایش می دهیم (داخل ویو و تمپلیت تغییرات زیر را می دهیم):

```

Visual layout of bidirectional text can depend on the base direction (View | Bidirectional Text Base Direction)
1  <!DOCTYPE html>
2  <html lang="en" dir="rtl">
3    <head>
4      <meta charset="UTF-8">
5      <title>من مقالات من</title>
6    </head>
7    <body>
8      <h1>لیست مقالات من</h1>
9      <ul>
10     {% if papers %}
11       <p>تعداد مقالات من : {{ papersCount }}</p>
12       {% for paper in papers %}
13         <li>
14           {{ paper }}
15         </li>
16       {% endfor %}
17     {% endif %}
18   </ul>
19 </body>
20 </html>

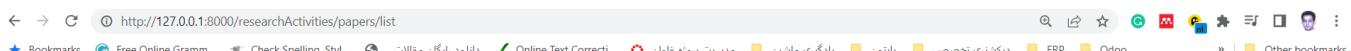
```

```

Visual layout of bidirectional text can depend on the base direction (View | Bidirectional Text Base Direction)
1  from django.http import HttpResponseRedirect
2  from django.shortcuts import render
3  from researchActivities.models import Paper
4
5  # Create your views here.
6
7
8  def paperListView(request):
9      papers = Paper.objects.all()
10
11      context = {
12          "papers": papers,
13          "papersCount": papers.count(),
14      }
15
16      return render(request, "researchActivities/papersList.html", context)

```

۸. برنامه را اجرا می کنیم:

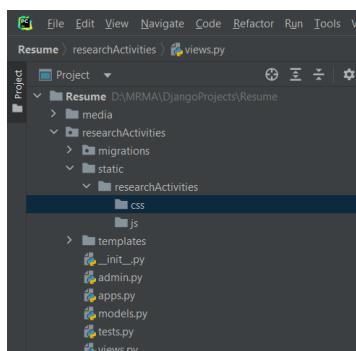


لیست مقالات

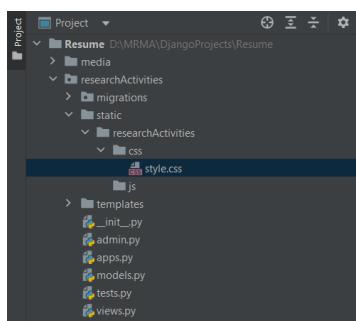
تعداد مقالات من: 2

- چالش‌های امنیتی جدید در حوزه پانکاری الکترونیک
- نقش فناوری اطلاعات در حلولگری از انتشار کرونا

۹. برای استفاده از فایل‌های استاتیک نظیر استایل‌ها و جاواسکریپت‌ها، دایرکتوری و ساب‌دایرکتوری‌های زیر را ایجاد می کنیم:



۱۰- داخل دایرکتوری style.css فایل style.css را ایجاد می کنیم (فعلاً خالی است):



۱۱- استفاده از فایل style.css داخل تمپلیت papersList از طریق تگ link: در خط دوم تمپلیت، با دستور { % load static % } دستور static را معرفی می کنیم.

```

<!DOCTYPE html>
{% load static %}
<html lang="en" dir="rtl">
<head>
    <meta charset="UTF-8">
    <title>جنگو</title>
    <link rel="stylesheet" type="text/css" href="{% static 'researchActivities/css/style.css' %}">
</head>
<body>
    <h1>تمپلیت مقالات</h1>
    <ul>
        {% if papers %}
            <p>تعداد مقالات: {{ papersCount }}</p>
            {% for paper in papers %}
                <li>
                    {{ paper }}
                </li>
            {% endfor %}
        {% endif %}
    </ul>
</body>

```

۱۲. در فایل settings.py، روت استاتیک را هم مشخص می کنیم:

```

TIME_ZONE = 'Asia/Tehran'
USE_I18N = True
USE_TZ = True

# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/4.1/howto/static-files/
STATIC_URL = 'static/'
STATIC_ROOT = os.path.join(BASE_DIR, 'static')

MEDIA_URL = 'media/'
MEDIA_ROOT = os.path.join(BASE_DIR, 'media')

# Default primary key field type
# https://docs.djangoproject.com/en/4.1/ref/settings/#default-auto-field
DEFAULT_AUTO_FIELD = 'django.db.models.BigAutoField'

```

۱۳. حال اگر بعنوان مثال استایل زیر را داخل style.css داشته باشیم:

```

p {
    color: red;
    font-size: 20px;
}

```

و برنامه را اجرا کنیم، خروجی زیر را خواهیم داشت:



لیست مقالات

تعداد مقالات من: 2

- چالش‌های امنیتی جدید در حوزه پانکاری المکترونیک
- نوش قنواری اطلاعات در جلوگیری از انتشار کووید ۱۹

کتابخانه‌های موردنیاز برای طراحی تمپلیت

۱. بوت استرپ: <https://getbootstrap.com/>

دانلود بوت استرپ از این مسیر: <https://getbootstrap.com/docs/>

۲. جی‌کوئری: <https://jquery.com>

دانلود جی‌کوئری فشرده از این مسیر: <https://jquery.com/download>

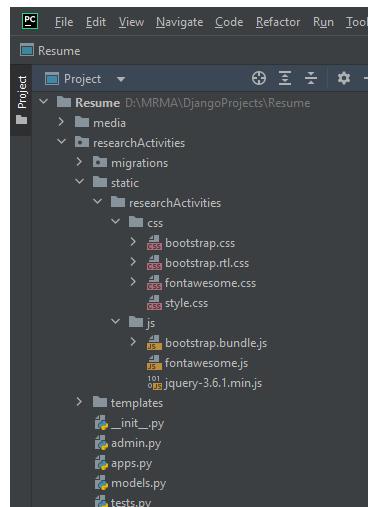
۳. فونت‌اسِم: <https://fontawesome.com>

دانلود فونت اسِم از این مسیر: <https://fontawesome.com/download>

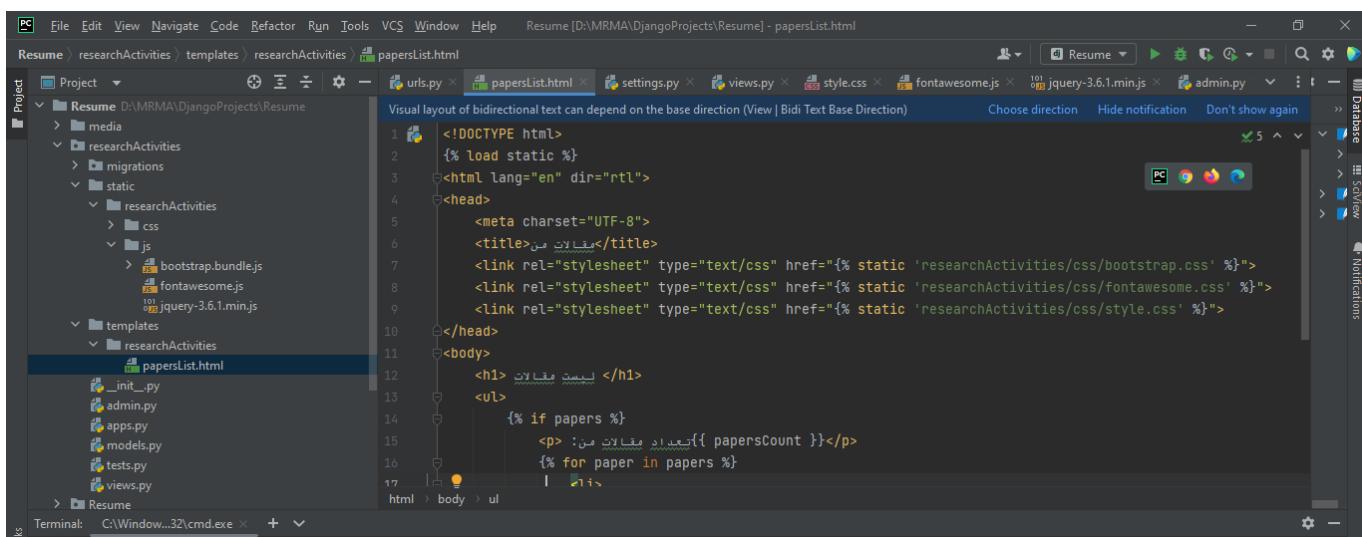
نحوه استفاده از کتابخانه‌های فوق

۱. داخل دایرکتوری css از اپ ایجاد شده (researchActivities | static| researchActivities | css) fontawesome.css را قرار دهید.

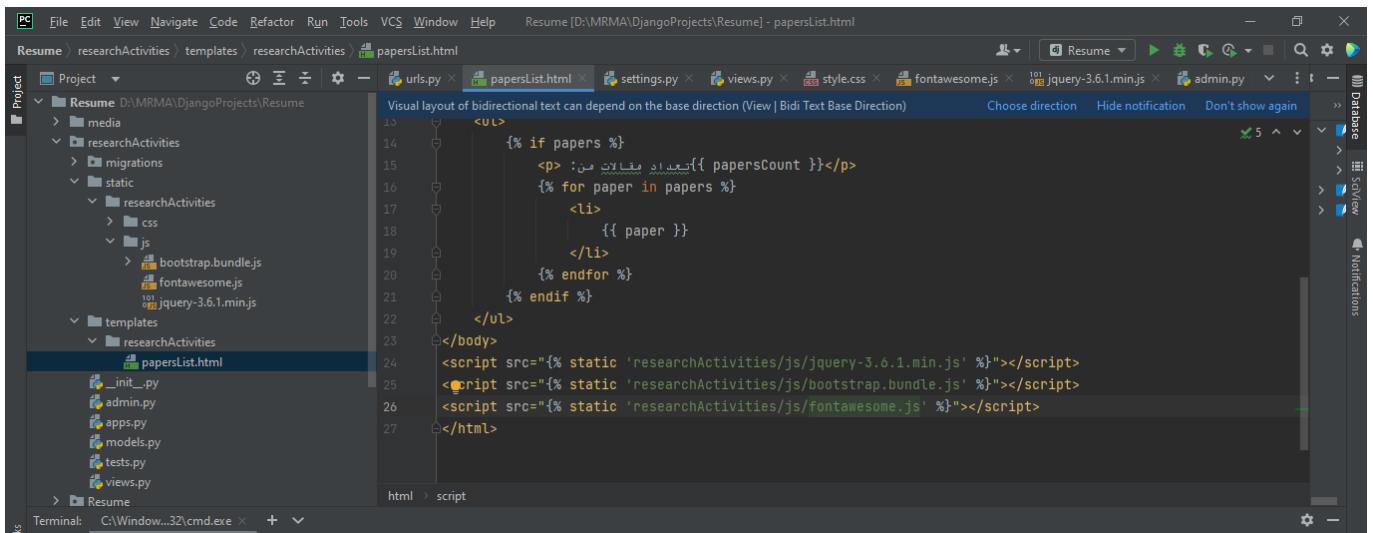
۲. داخل دایرکتوری js از اپ ایجاد شده (researchActivities | static| researchActivities | js) jquery-3.6.1.min.js و fontawesome.js را قرار دهید.



۳. افزودن لینک css‌ها و js‌ها به کد



نکته: لینک ها داخل تگ head و لینک فایل css خودتان، بعد از لینک های دیگر باشد.



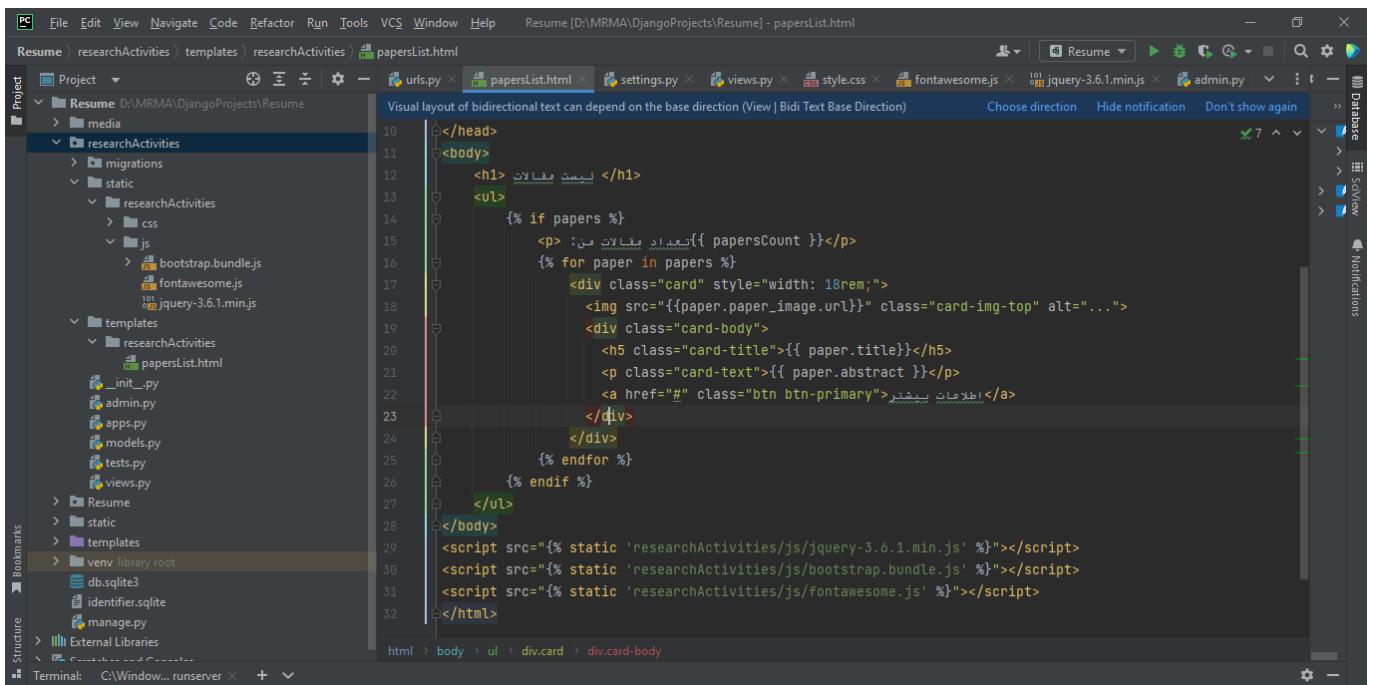
```

<ul>
    {% if papers %}
        <p>تعداد مقالات: {{ papersCount }}</p>
        {% for paper in papers %}
            <li>
                {{ paper }}
            </li>
        {% endif %}
    {% endif %}
</ul>
</body>
<script src="{% static 'researchActivities/js/jquery-3.6.1.min.js' %}"></script>
<script src="{% static 'researchActivities/js/bootstrap.bundle.js' %}"></script>
<script src="{% static 'researchActivities/js/fontawesome.js' %}"></script>
</html>

```

نکته: اسکریپت ها بعد از تگ body و ترتیب اسکریپت ها بهمان ترتیب فوق باشد.

۴. یک تمپلیت نمونه:



```

<head>
<body>
    <h1>لیست مقالات</h1>
    <ul>
        {% if papers %}
            <p>تعداد مقالات: {{ papersCount }}</p>
            {% for paper in papers %}
                <div class="card" style="width: 18rem;">
                    
                    <div class="card-body">
                        <h5 class="card-title">{{ paper.title }}</h5>
                        <p class="card-text">{{ paper.abstract }}</p>
                        <a href="#" class="btn btn-primary">اطلاعات بیشتر</a>
                    </div>
                </div>
            {% endif %}
        {% endif %}
    </ul>
</body>
<script src="{% static 'researchActivities/js/jquery-3.6.1.min.js' %}"></script>
<script src="{% static 'researchActivities/js/bootstrap.bundle.js' %}"></script>
<script src="{% static 'researchActivities/js/fontawesome.js' %}"></script>
</html>

```

سیستم Grid در بوت استرپ

سیستم Grid در بوت استرپ به شما اجازه می‌دهد که بر مبنای ۱۲ ستون، صفحه‌ی خود را ایجاد کنید. اگر شما نمی‌خواهید از ۱۲ ستون به طور مشخص استفاده کنید می‌توانید ستون‌ها را با هم ترکیب کرده و ستون‌های عریض‌تری ایجاد کنید:

span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1	span 1
span 4				span 4				span 4				
span 4		span 8								span 6		
span 6						span 6						
span 12												

سیستم Grid در بوت استرپ **واکنش‌گرا** است، و اندازه ستون‌ها بسته به اندازه‌ی صفحه نمایش به طور خودکار تغییر می‌کند.

کلاس‌های Grid

	xs <576px	sm ≥576px	md ≥768px	lg ≥992px	xl ≥1200px	xxl ≥1400px
Container max-width	None (auto)	540px	720px	960px	1140px	1320px
Class prefix	.col-	.col-sm-	.col-md-	.col-lg-	.col-xl-	.col-xxl-
# of columns	12					
Gutter width	1.5rem (.75rem on left and right)					
Custom gutters	Yes					
Nestable	Yes					
Column ordering	Yes					

کلاس‌های grid در بوت استرپ ۵، سر جمع ۶ کلاس هستند:

.col-. برای دستگاه‌های بسیار کوچک با سایز صفحه‌ی زیر ۵۷۶ پیکسل ✓

.col-sm-. برای دستگاه‌های کوچک با سایز صفحه‌ی مساوی با یا کمتر از ۵۷۶ پیکسل ✓

.col-md-. برای دستگاه‌های متوسط با سایز صفحه‌ی مساوی با یا بیشتر از ۷۶۸ پیکسل ✓

.col-lg-. برای دستگاه‌های بزرگ با سایز صفحه‌ی مساوی با یا بیشتر از ۹۹۲ پیکسل ✓

.col-xl-. برای دستگاه‌های بسیار بزرگ با سایز صفحه‌ی مساوی با یا بیشتر از ۱۲۰۰ پیکسل ✓

.col-xxl-. برای دستگاه‌های بسیار بزرگ با سایز صفحه‌ی مساوی با یا بیشتر از ۱۴۰۰ پیکسل ✓

کلاس‌های بالا می‌توانند با هم ترکیب شده و صفحاتی پویاتر و قابل انعطاف‌تر را ایجاد کنند.

ساختار پایه یک Grid در بوت استرپ

در نمونه مثال زیر می‌توانید یک ساختار Grid در بوت استرپ را مشاهده کنید:

```
<div class="row">
  <div class="col-*-*"></div>
  <div class="col-*-*"></div>
</div>
<div class="row">
  <div class="col-*-*"></div>
  <div class="col-*-*"></div>
  <div class="col-*-*"></div>
</div>
<div class="row">
  ...
</div>
```

در نمونه مثال بالا ابتدا یک row ایجاد می‌کنیم، سپس تعدادی ستون را اضافه می‌کنیم (ستون‌هایی که با کلاس col-* مشخص شده‌اند) دقت داشته باشید که مجموع اعداد در ستون‌هایی که با col-* برای هر row مشخص می‌شوند باید برابر با عدد ۱۲ باشد.

در زیر نمونه مثال‌هایی آورده شده است که می‌توانید با استفاده از آنها سیستم گردید در بوت استرپ را به سادگی فرا بگیرید.

سه ستون با پهنایی برابر

.col-sm-4	.col-sm-4	.col-sm-4
-----------	-----------	-----------

در نمونه مثال زیر می‌توان مشاهده نمود که چطور می‌توان به ساختاری سه ستونی با پهنایی برابر رسید که از تبلت‌ها تا دسکتاپ‌های بزرگ مقیاس‌پذیر است و در تلفن‌های همراه یا صفحه نمایش‌هایی که پهنای آنها از ۵۷۶ پیکسل کمتر باشد ستون‌ها به صورت خودکار در زیر هم قرار می‌گیرند:

```
<div class="row">
  <div class="col-sm-4">.col-sm-4</div>
  <div class="col-sm-4">.col-sm-4</div>
  <div class="col-sm-4">.col-sm-4</div>
</div>
```

دو ستون با پهنایی متفاوت

.col-sm-4	.col-sm-8
-----------	-----------

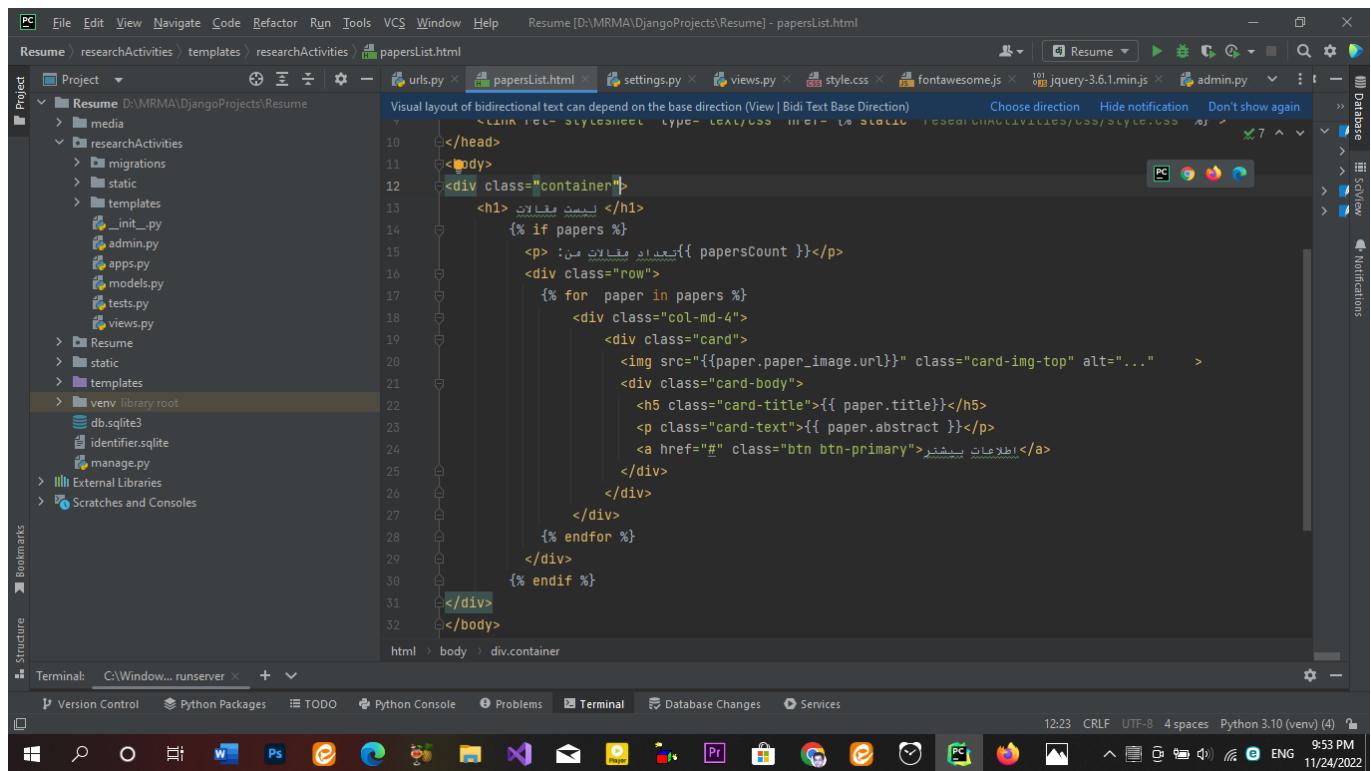
در نمونه مثال زیر می‌توان مشاهد کرد که چطور می‌شود به ساختاری دو ستونی با پهنایی نابرابر رسید که از تبلت‌ها تا دسکتاپ‌های بزرگ مقیاس‌پذیر است:

```
<div class="row">
  <div class="col-sm-4">.col-sm-4</div>
  <div class="col-sm-8">.col-sm-8</div>
</div>
```

تکه کد زیر به چه معناست؟

```
<div class="row">
  <div class="col-xs-12 col-md-4">div1</div>
  <div class="col-xs-12 col-md-4">div2</div>
  <div class="col-xs-12 col-md-4">div3</div>
</div>
```

۵. طراحی بهتر با گرید:

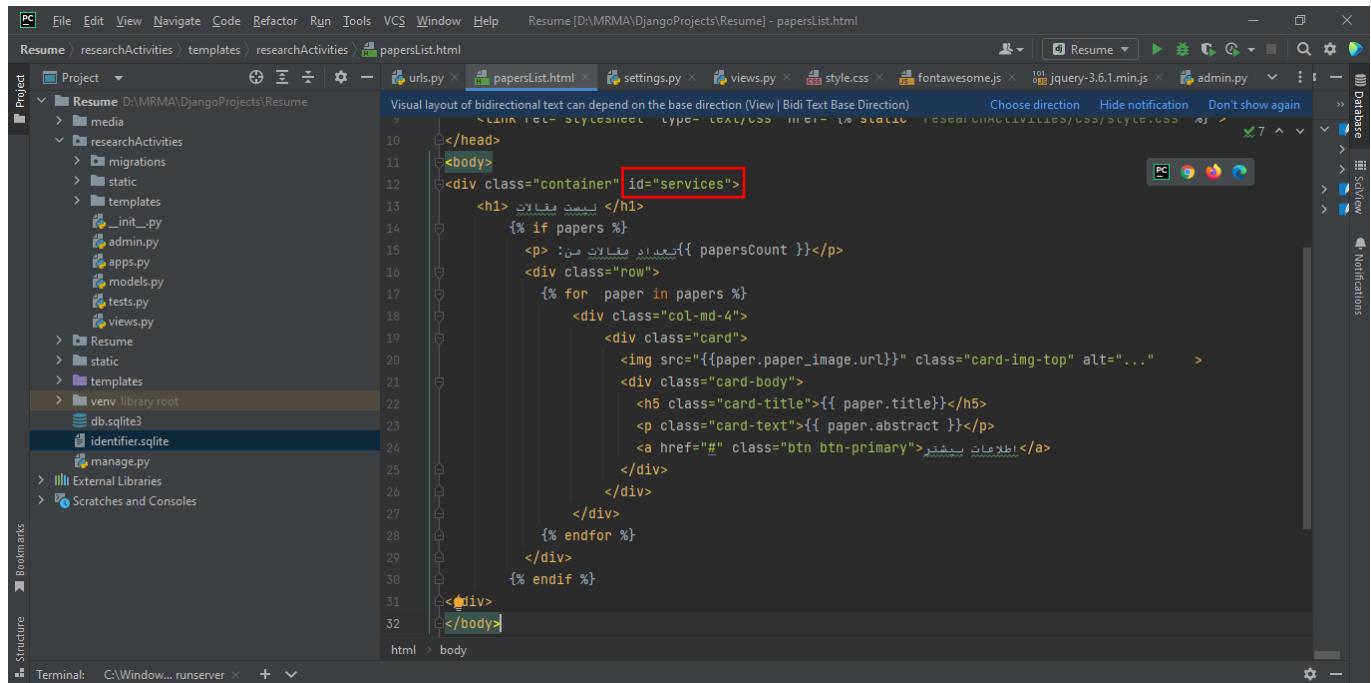


```

<head>
</head>
<body>
<div class="container">
    <h1>لیست مقالات</h1>
    {% if papers %}
        <p>تعداد مقالات من : {{ papersCount }}</p>
        <div class="row">
            {% for paper in papers %}
                <div class="col-md-4">
                    <div class="card">
                        
                        <div class="card-body">
                            <h5 class="card-title">{{ paper.title }}</h5>
                            <p class="card-text">{{ paper.abstract }}</p>
                            <a href="#" class="btn btn-primary">اطلاعات بیشتر</a>
                        </div>
                    </div>
                {% endfor %}
            </div>
        {% endif %}
    </div>
</body>

```

۶. طراحی بهتر با بوت استرپ و استایل دهی:



```

<head>
</head>
<body>
<div class="container" id="services">
    <h1>لیست مقالات</h1>
    {% if papers %}
        <p>تعداد مقالات من : {{ papersCount }}</p>
        <div class="row">
            {% for paper in papers %}
                <div class="col-md-4">
                    <div class="card">
                        
                        <div class="card-body">
                            <h5 class="card-title">{{ paper.title }}</h5>
                            <p class="card-text">{{ paper.abstract }}</p>
                            <a href="#" class="btn btn-primary">اطلاعات بیشتر</a>
                        </div>
                    </div>
                {% endfor %}
            </div>
        {% endif %}
    </div>
</body>

```

```

PC File Edit View Navigate Code Refactor Run Tools VCS Window Help Resume [D:\MRMA\ DjangoProjects\Resume] - style.css
Resume > researchActivities > static > researchActivities > css > style.css
Project urls.py papersList.html settings.py views.py style.css fontawesome.js jquery-3.6.1.min.js admin.py
1 #services img {
2     object-fit: cover;
3     height : 15rem;
4     transition: transform 0.2s;
5 }
6 #services img:hover {
7     transform: scale(1.1);
8 }
9

```

به شما امکان می دهد مقادیر یک پر اپرتی (ویژگی) را به آرامی، در مدت زمان معین تغییر دهید. برای ایجاد یک transition effect باید دو چیز را مشخص کنید:

ویژگی CSS که می خواهد به آن افکت اضافه کند ✓

مدت زمان افکت ✓

در CSS فوق، المان img از دیو با آیدی services دارای transition effect با پر اپرتی transform بمدت 0.2s است. پر اپرتی transform می دهد المانها را جابه جا کنید، بچرخانید یا مقیاس بندی کنید. این transition effect وقتی شروع خواهد شد که پر اپرتی مشخص شده برای آن یعنی transform تغییر مقدار دهد، لذا با قرار گرفتن موس روی img مقدار transform را تغییر می دهیم (transform: scale(1.1)).

پر اپرتی scale به شما امکان می دهد اندازه عناصر را تغییر دهید.

۷. برای اینکه متن بدن کارت، اسکرول داشته باشد:

```

PC File Edit View Navigate Code Refactor Run Tools VCS Window Help Resume [D:\MRMA\ DjangoProjects\Resume] - papersList.html
Resume > researchActivities > templates > researchActivities > papersList.html
Project urls.py settings.py views.py style.css fontawesome.js jquery-3.6.1.min.js admin.py
10 </head>
11 <body>
12 <div class="container" id="services">
13     <h1>لیست مقالات</h1>
14     {% if papers %}
15         <p>تعداد مقالات من: {{ papersCount }}</p>
16         <div class="row">
17             {% for paper in papers %}
18                 <div class="col-md-4">
19                     <div class="card">
20                         
21                         <div class="card-body">
22                             <h5 class="card-title">{{ paper.title }}</h5>
23                             <p class="scroll-box">{{ paper.abstract }}</p>
24                             <a href="#" class="btn btn-primary" data-toggle="modal" data-target="#detailsModal">اطلاعات بیشتر</a>
25                         </div>
26                     </div>
27                 {% endfor %}
28             </div>
29         {% endif %}
30     </div>
31 </div>
32 </body>

```

```
#services img {
    object-fit: cover;
    height : 15rem;
    transition: transform 0.2s;
}

#services img:hover {
    transform: scale(1.1);
}

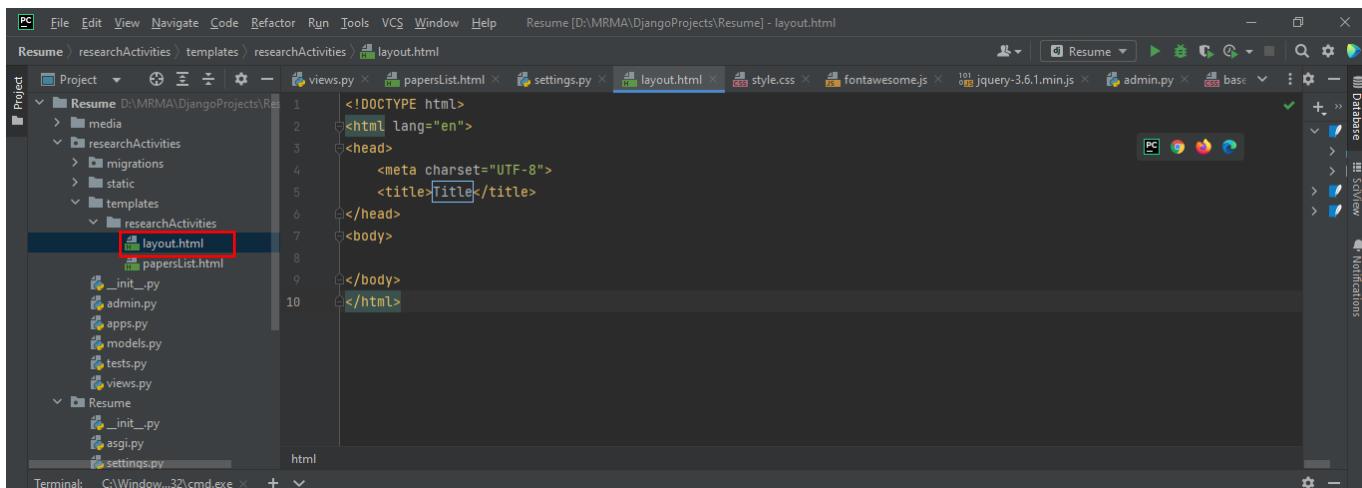
.scroll-box {
    overflow-y: scroll;
    height: 200px;
    padding: 1rem
}
```

لیاوت در جنگو

هنگامی که تمپلیت‌های بیشتری را به پروژه خود اضافه می‌کنید، متوجه خواهد شد که بسیاری از بخش‌های تمپلیت‌های متفاوت مشابه هستند. ساختار کلی همه تمپلیت‌ها نسبتاً ثابت می‌ماند: همه تمپلیت‌ها دارای یک هدر و فوتر مشترک هستند، در حالی که محتوای داخل ساختار ممکن است تغییر کند. با `Template Inheritance`، این امکان وجود دارد که یک بار تمپلیت HTML اسکلت را تعریف کنید و سپس از آن تمپلیت پایه برای سایر تمپلیت‌ها ارت برید.

اگر بخواهید تمپلیت پایه را برای تمام App‌های پروژه جنگو به اشتراک بگذارید، این تمپلیت پایه را در پوشه `templates` ریشه اصلی پروژه قرار دهید. اگر ترجیح می‌دهید برای هر برنامه یک تمپلیت پایه منحصر به فرد داشته باشد، می‌توانید یک تمپلیت پایه را در خود هر App قرار دهید.

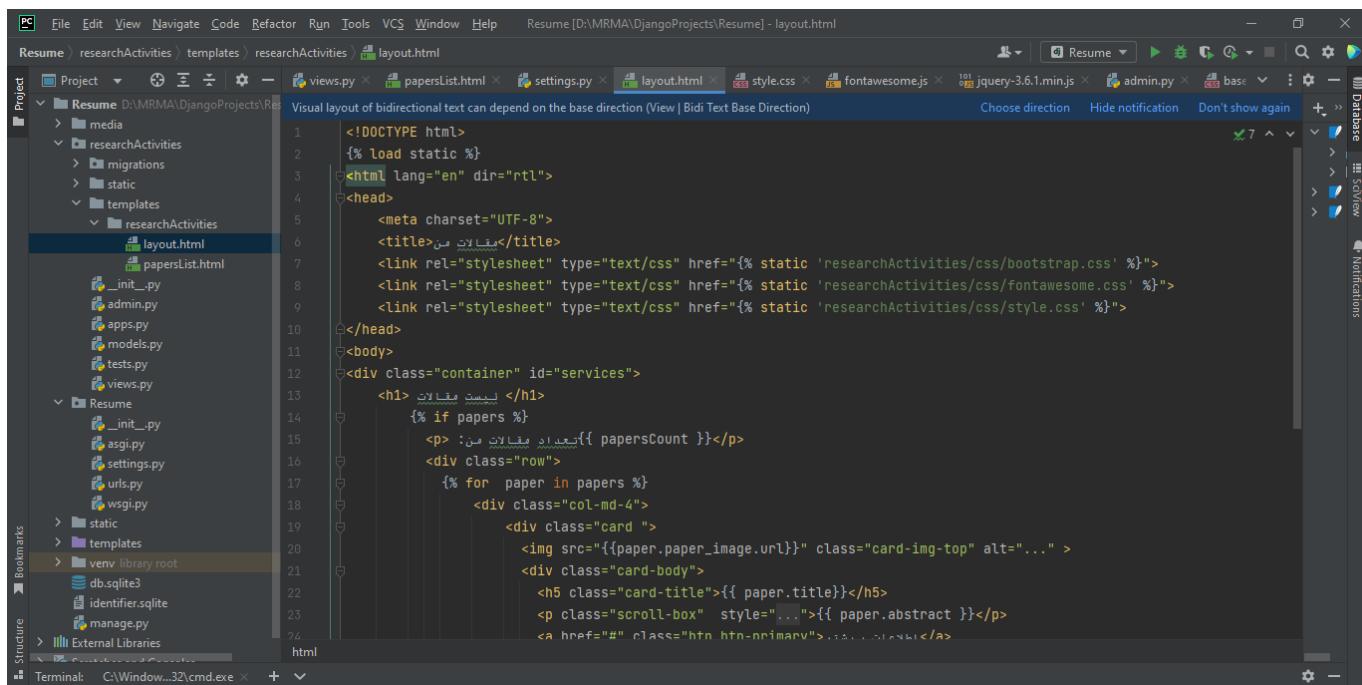
۱. یک فایل html با نام layout.html را داخل ساب دایرکتوری researchActivities از دایرکتوری templates ایجاد می‌کنیم:



The screenshot shows the PyCharm IDE interface. The project structure on the left includes a `Resume` project with `researchActivities`, `migrations`, `static`, and `templates` directories. Inside `templates`, there is a `researchActivities` directory which contains `layout.html`. The `layout.html` file is open in the editor, showing its code:

```
<!DOCTYPE html>
<html lang="en">
    <head>
        <meta charset="UTF-8">
        <title>Title</title>
    </head>
    <body>
    </body>
</html>
```

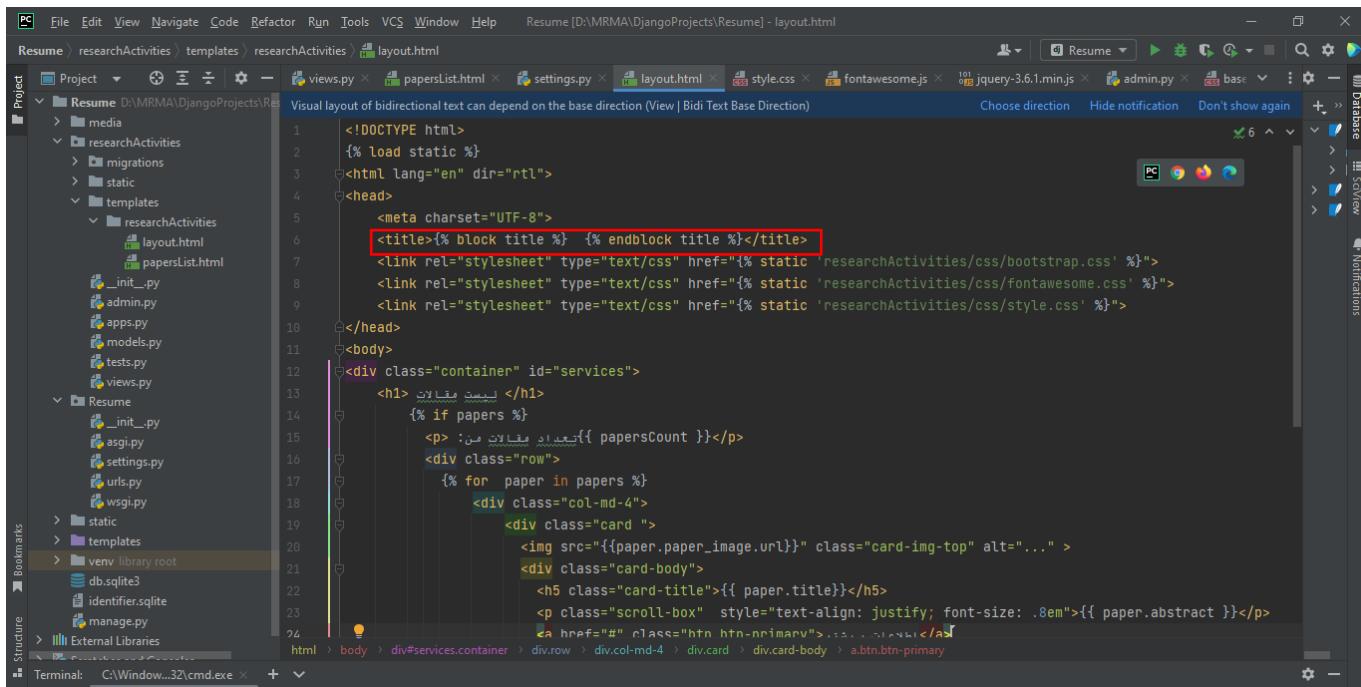
۲. محتوای تمپلیت برنامه را (paperList.html) را کپی و آنرا جایگزین محتوای layout.html می‌کنیم:



The screenshot shows the PyCharm IDE interface again. The project structure is the same as before. The `layout.html` file now contains the content of the `paperList.html` file. A notification at the top right of the editor window says "Visual layout of bidirectional text can depend on the base direction (View | Bidi Text Base Direction)". The `paperList.html` file's code is as follows:

```
<!DOCTYPE html>
{% load static %}
<html lang="en" dir="rtl">
    <head>
        <meta charset="UTF-8">
        <title>مقالات من:</title>
        <link rel="stylesheet" type="text/css" href="{% static 'researchActivities/css/bootstrap.css' %}">
        <link rel="stylesheet" type="text/css" href="{% static 'researchActivities/css/fontawesome.css' %}">
        <link rel="stylesheet" type="text/css" href="{% static 'researchActivities/css/style.css' %}">
    </head>
    <body>
        <div class="container" id="services">
            <h1>لیست مقالات </h1>
            {% if papers %}
                <p>تعداد مقالات من: {{ papersCount }}</p>
                <div class="row">
                    {% for paper in papers %}
                        <div class="col-md-4">
                            <div class="card">
                                
                                <div class="card-body">
                                    <h5 class="card-title">{{ paper.title }}</h5>
                                    <p class="scroll-box" style="..."><{{ paper.abstract }}></p>
                                    <a href="#" class="btn btn-primary">...</a>
                                </div>
                            </div>
                        </div>
                    {% endfor %}
                </div>
            {% endif %}
        </div>
    </body>
</html>
```

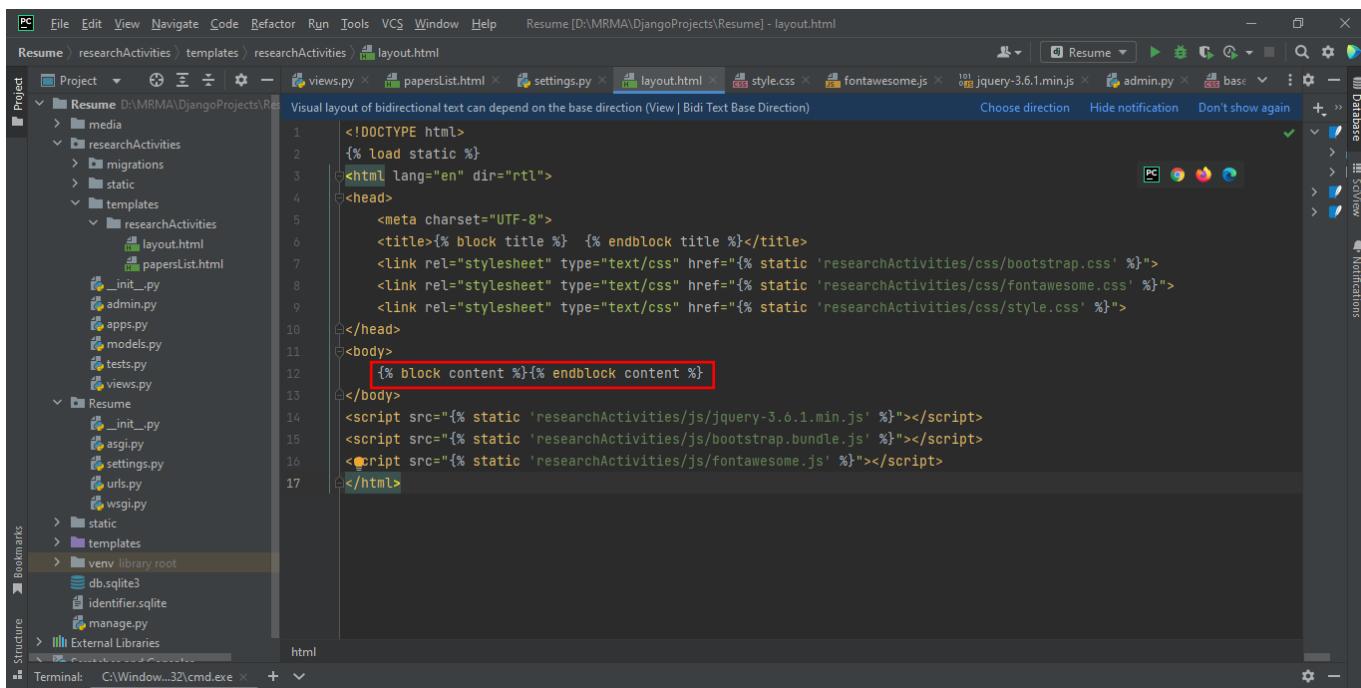
۳. محتوای تگ head برای سایر تمپلیت‌ها باید تکرار شود پس آنرا نگه می‌داریم. فقط مقدار داخل تگ title برای هر تمپلیت منحصر بفرد خواهد بود لذا آنرا با تگ block جایگزین می‌کنیم. تگ بلاک برای تعریف بلاکی استفاده می‌شود که می‌تواند توسط تمپلیت‌های فرزند بازنویسی شود. به عبارت دیگر، هنگامی که یک بلاک را در تمپلیت پایه تعریف می‌کنید، می‌گویید که این ناحیه از طریق یک تمپلیت فرزند پر خواهد شد:



```
<!DOCTYPE html>
{% load static %}
<html lang="en" dir="rtl">
    <head>
        <meta charset="UTF-8">
        <title>% block title % { % endblock title %}</title>
        <link rel="stylesheet" type="text/css" href="{% static 'researchActivities/css/bootstrap.css' %}">
        <link rel="stylesheet" type="text/css" href="{% static 'researchActivities/css/fontawesome.css' %}">
        <link rel="stylesheet" type="text/css" href="{% static 'researchActivities/css/style.css' %}">
    </head>
    <body>
        <div class="container" id="services">
            <h1>لیست مقالات </h1>
            {% if papers %}
                <p>تعداد مقالات من: {{ papersCount }}</p>
                <div class="row">
                    {% for paper in papers %}
                        <div class="col-md-4">
                            <div class="card">
                                
                                <div class="card-body">
                                    <h5 class="card-title">{{ paper.title }}</h5>
                                    <p class="scroll-box" style="text-align: justify; font-size: .8em">{{ paper.abstract }}</p>
                                    <a href="#" class="btn btn-primary" style="color: white; background-color: #007bff; border-color: #007bff">دانلود</a>
                            </div>
                        </div>
                    {% endfor %}
                </div>
            {% endif %}
        </div>
    </body>
</html>
```

نکه: آنچه مابین { % block title % } و { % endblock title %} است که از تمپلیت‌های فرزند ارسال خواهد شد.

۴. یک بلاک دیگر برای «محتوای لی اوت» در نظر می‌گیریم. آنچه داخل تگ body هست را حذف و بجای آن { % block content % } { % endblock content % } را جایگزین می‌کنیم.



```
<!DOCTYPE html>
{% load static %}
<html lang="en" dir="rtl">
    <head>
        <meta charset="UTF-8">
        <title>% block title % { % endblock title %}</title>
        <link rel="stylesheet" type="text/css" href="{% static 'researchActivities/css/bootstrap.css' %}">
        <link rel="stylesheet" type="text/css" href="{% static 'researchActivities/css/fontawesome.css' %}">
        <link rel="stylesheet" type="text/css" href="{% static 'researchActivities/css/style.css' %}">
    </head>
    <body>
        {% block content %}{% endblock content %}
    </body>
</html>
```

۵. استفاده از لی اوت تعریف شده در سایر تمپلیت‌ها. داخل تمپلیت paperList.html، ابتدا قسمت‌های مشترک را حذف می‌کنیم و فقط محتوای خاص تمپلیت را نگه می‌داریم. قسمت‌های مشترک دیگر مورد نیاز نیستند زیرا وقتی این صفحه بارگذاری می‌شود، HTML موجود در تمپلیت پایه را به ارث می‌برند:

![...]({{paper.paper_image.url}}){{ paper.abstract }}</p>
 اطلاعات بیشتر
 </div>
 </div>
 {% endfor %}
 {% endif %}
 </div>
 </div>
</div>
"/>

۶. هنگامی که تمپلیت پایه خود را دارید (layout.html)، می‌توانید آن را در سایر تمپلیت‌های فرزند گسترش دهید. راه انجام این کار با کلمه کلیدی extends است. بنابراین در فایل paperList.html که روی آن کار می‌کنیم، می‌توانیم آن تگ را در بالای فایل اضافه کنیم:

![...]({{paper.paper_image.url}}){{ paper.abstract }}</p>
 اطلاعات بیشتر
 </div>
 </div>
 {% endfor %}
 {% endif %}
 </div>
 </div>
</div>
"/>

۷. مقداردهی بلاک‌های تمپلیت پایه در تمپلیت paperList.html برای اینکار ابتدا مقدار بلاک title را که قرار است از تمپلیت list به تمپلیت پایه تزریق شود را مشخص می‌کنیم:

![...]({{paper.paper_image.url}})

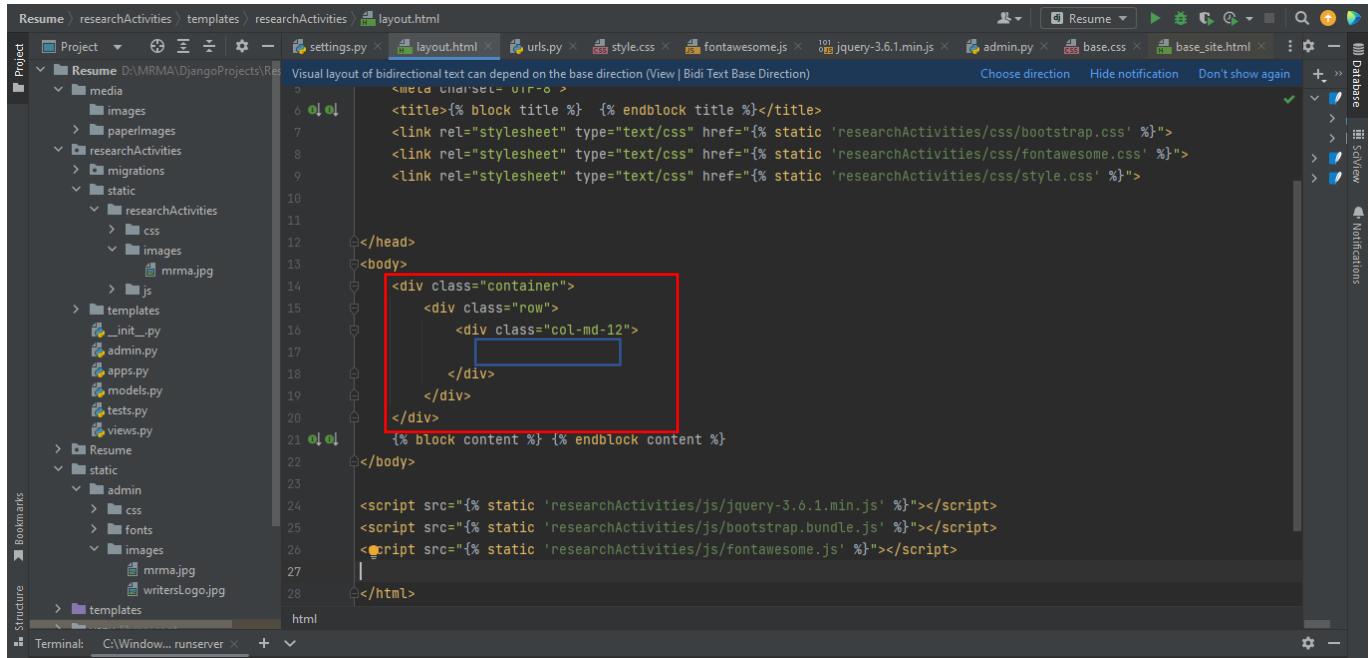
۸. محتوای اصلی تمپلیت paperList.html content را داخل بلاک content قرار می‌دهیم:

![...]({{paper.paper_image.url}})

۹. برنامه را اجرا می کنیم. می بینیم که لی اوت بدرستی شکل گرفته است:

ایجاد یک هدر ریسپانسیو

۱. ایجاد divهای زیر در تگ body از تمپلیت پایه layout.html

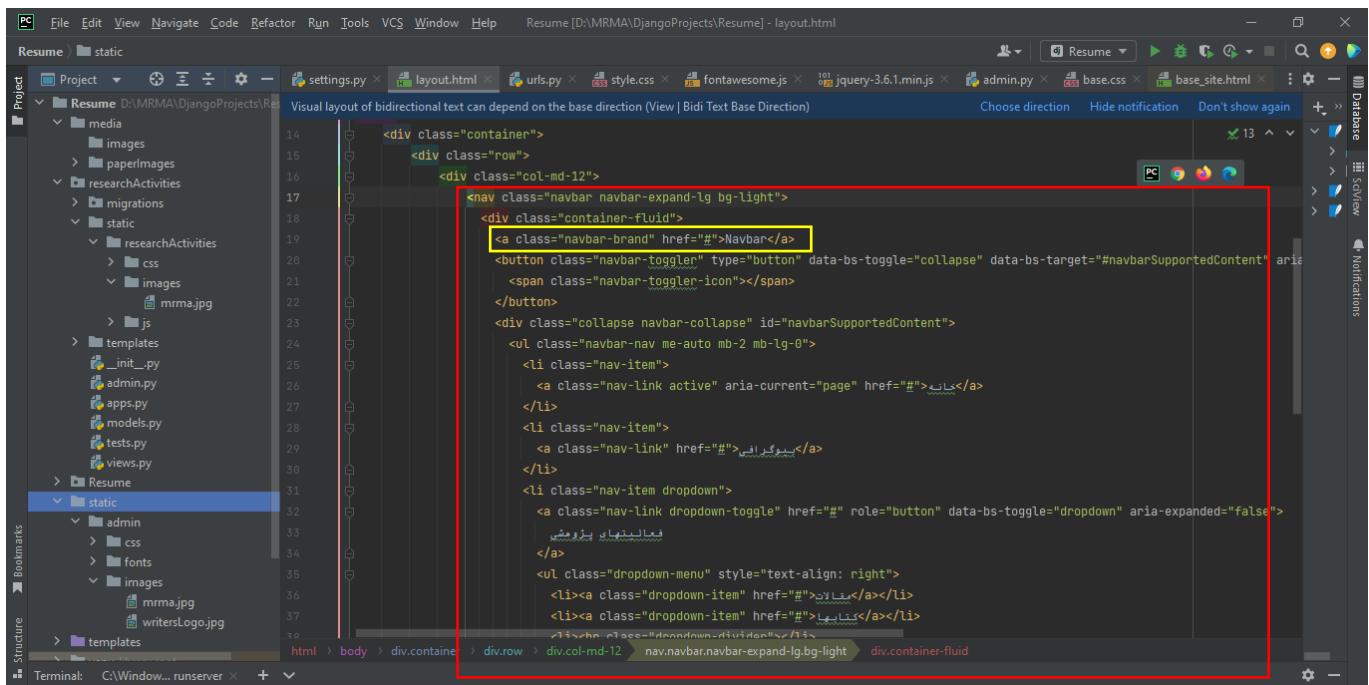


```

<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>{% block title %} {% endblock title %}</title>
        <link rel="stylesheet" type="text/css" href="{% static 'researchActivities/css/bootstrap.css' %}">
        <link rel="stylesheet" type="text/css" href="{% static 'researchActivities/css/fontawesome.css' %}">
        <link rel="stylesheet" type="text/css" href="{% static 'researchActivities/css/style.css' %}">
    </head>
    <body>
        <div class="container">
            <div class="row">
                <div class="col-md-12">
                    <!-- Content -->
                </div>
            </div>
        </div>
        {% block content %} {% endblock content %}
    </body>
</html>

```

۲. کپی کردن تکه کد مربوط به nav bar از مستندات بوت استرپ در آدرس <https://getbootstrap.com/docs/5.2/components/navbar> و قرار دادن آن در تگ div با کلاس col-md-12 (داخل کادر آبی رنگ بالا):

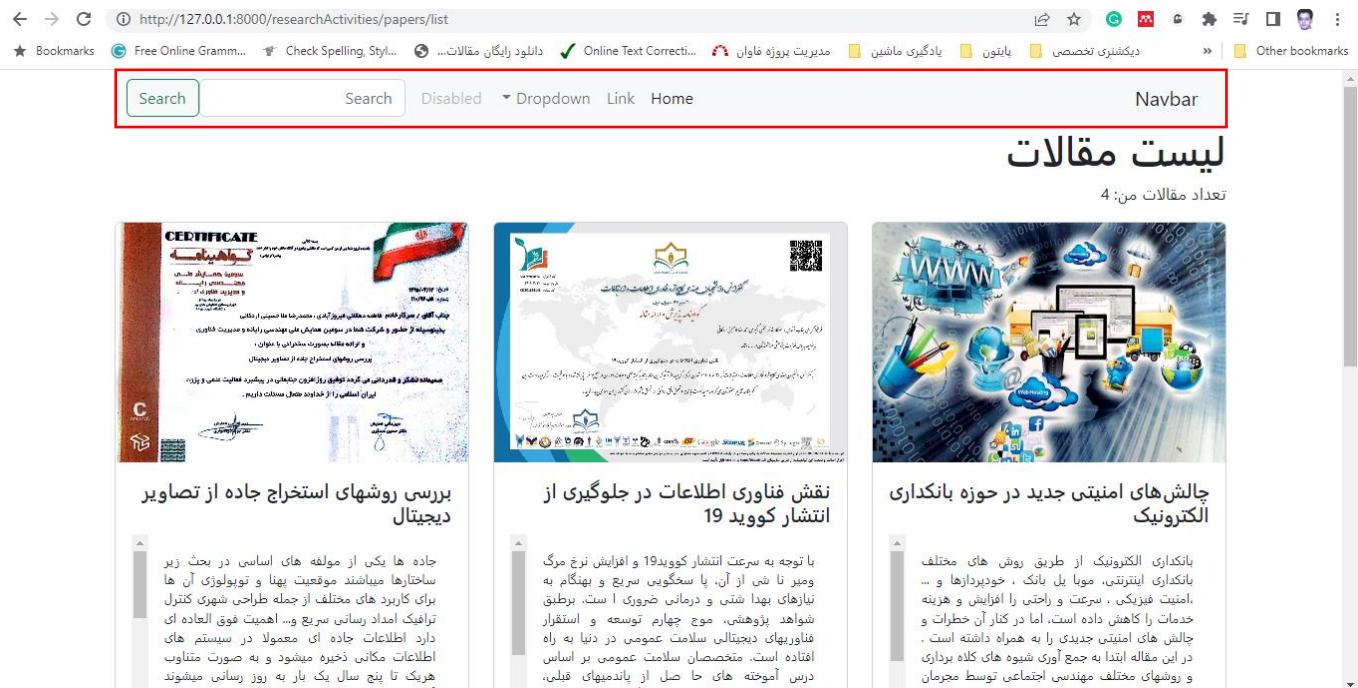


```

<div class="container">
    <div class="row">
        <div class="col-md-12">
            <nav class="navbar navbar-expand-lg bg-light">
                <div class="container-fluid">
                    <a class="navbar-brand" href="#">Navbar</a>
                    <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">
                        <span class="navbar-toggler-icon"></span>
                    </button>
                    <div class="collapse navbar-collapse" id="navbarSupportedContent">
                        <ul class="navbar-nav me-auto mb-2 mb-lg-0">
                            <li class="nav-item">
                                <a class="nav-link active" aria-current="page" href="#">دانشجویی
                            </li>
                            <li class="nav-item">
                                <a class="nav-link" href="#">پژوهشی
                            </li>
                            <li class="nav-item dropdown">
                                <a class="nav-link dropdown-toggle" href="#" role="button" data-bs-toggle="dropdown" aria-expanded="false">
                                    فعالیتهای پژوهشی
                                </a>
                                <ul class="dropdown-menu" style="text-align: right">
                                    <li><a class="dropdown-item" href="#">دانشجوییپژوهشیدانشجویی

```

۳. برنامه را اجرا کنید؛ یک منوی ریسپانسیو:



لیست مقالات

تعداد مقالات من: 4

بررسی روش‌های استخراج جاده از تصاویر دیجیتال

جاده‌ها یکی از مولفه‌های اساسی در بحث زیر ساختارها می‌باشند. موقعيت پهنا و تنویلی آن‌ها برای کاربرد هرگز مغایر نمی‌باشد. معمولاً در سیستم‌های ترافیک امداد رسانی سریع... اهیت فوق العاده‌ای دارد. اطلاعات جاده‌ای معمولاً در سیستم‌های اطلاعات مکانی ذخیره می‌شود و به صورت متنابض هریک تا پنج سال یک بار به روز رسانی می‌شوند.

چالش‌های امنیتی جدید در حوزه بانکداری الکترونیک

بانکداری الکترونیک از طریق روش‌های مختلف بانکداری اینترنتی، موبایل بانک، خودبردارها... امنیت فیزیکی، سرعت و راحتی را افزایش و هزینه خدمات را کاهش داده است، اما در کاران خطوط و چالش‌های امنیتی جدیدی را به همراه داشته است. در این مقاله ابتدا به جمع آوری شیوه‌های کلاه برداری و روش‌های مختلف مهندسی اجتماعی توسط مجرمان



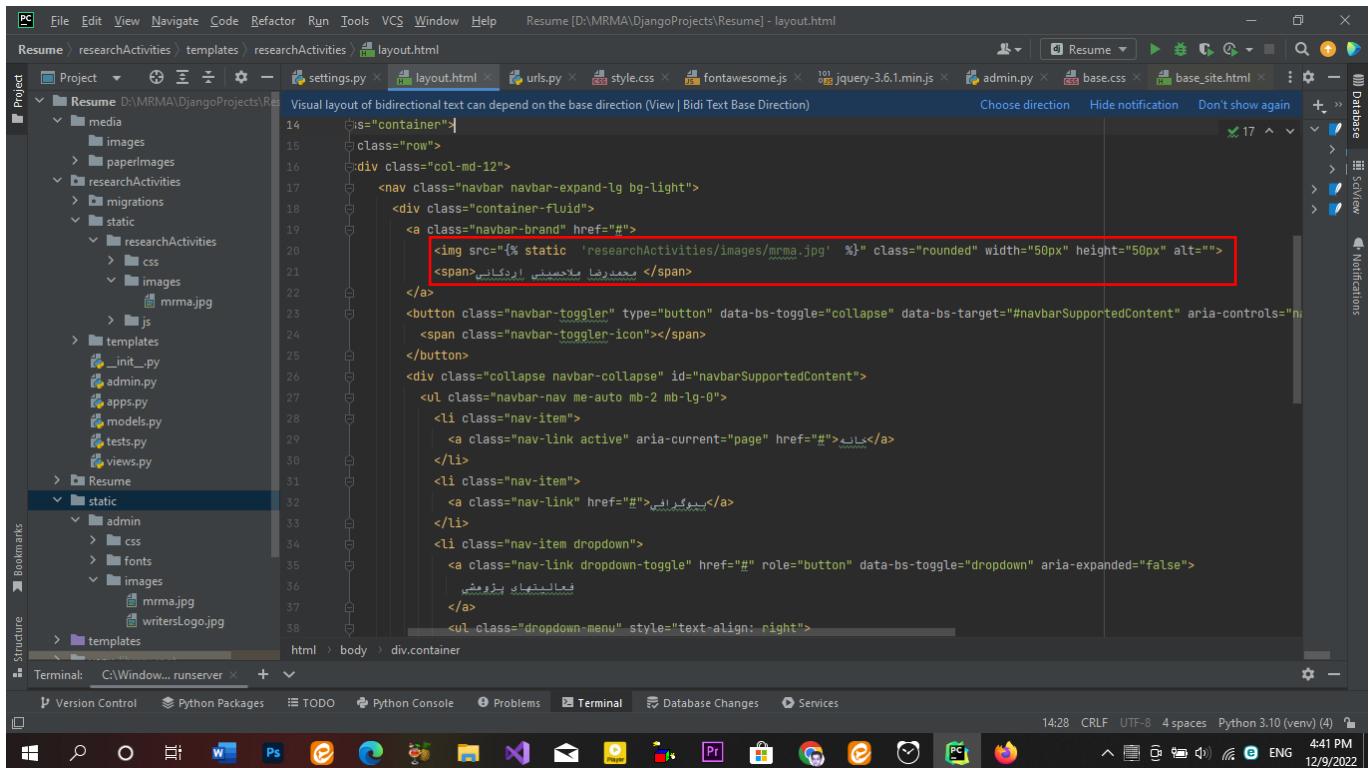
لیست مقالات

تعداد مقالات من: 4

چالش‌های امنیتی جدید در حوزه بانکداری الکترونیک

بانکداری الکترونیک از طریق روش‌های مختلف بانکداری اینترنتی، موبایل بانک، خودبردارها... امنیت فیزیکی، سرعت و راحتی را افزایش و هزینه خدمات را کاهش داده است، اما در کاران خطوط و چالش‌های امنیتی جدیدی را به همراه داشته است. در این مقاله ابتدا به جمع آوری شیوه‌های کلاه برداری و روش‌های مختلف مهندسی اجتماعی توسط مجرمان

۴. بجای Navbar در کادر زردنگ بالا تصویر قرار می‌دهیم:



```

File Edit View Navigate Code Refactor Run Tools VCS Window Help Resume [D:\MRMA\ DjangoProjects\Resume] - layout.html
Resume > researchActivities > templates > researchActivities > layout.html
Project
  - Resume
    - media
      - images
    - static
      - researchActivities
        - css
        - images
          - mrmaj.jpg
        - js
      - templates
        - __init__.py
        - admin.py
        - apps.py
        - models.py
        - tests.py
        - views.py
    - Resume
    - static
      - admin
        - css
        - fonts
        - images
          - mrmaj.jpg
          - writersLogo.jpg
      - templates
  - settings.py
  - urls.py
  - style.css
  - fontawesome.js
  - jquery-3.6.1.min.js
  - admin.py
  - base.css
  - base_site.html
  - layout.html
  - Choose direction Hide notification Don't show again
  - Database
  - Notifications
  - SCView
  - Notifications
  - Terminal: C:\Windows\system32\runserver ...
  - Version Control Python Packages TODO Problems Database Changes Services
  - 14:28 CRLF UTF-8 4 spaces Python 3.10 (venv) (4)
  - 12/9/2022

```

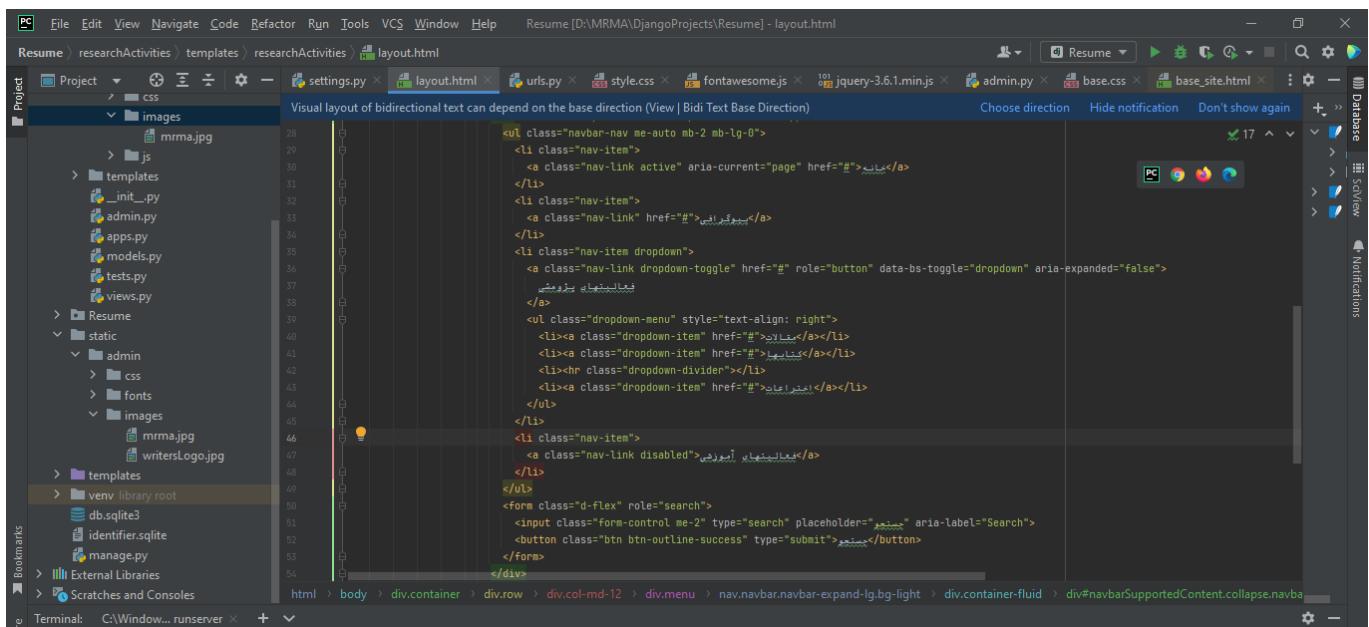
Visual layout of bidirectional text can depend on the base direction (View | Bidi Text Base Direction)

```

14 <div class="container">
15   <div class="row">
16     <div class="col-md-12">
17       <nav class="navbar navbar-expand-lg bg-light">
18         <div class="container-fluid">
19           <a class="navbar-brand" href="#">
20             
21             محمد رضا ملاحی مهندسی اردکانی
22           </a>
23           <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-label="Toggle navigation">
24             <span class="navbar-toggler-icon"></span>
25           </button>
26           <div class="collapse navbar-collapse" id="navbarSupportedContent">
27             <ul class="navbar-nav me-auto mb-2 mb-lg-0">
28               <li class="nav-item">
29                 <a class="nav-link active" aria-current="page" href="#">خانه</a>
30               </li>
31               <li class="nav-item">
32                 <a class="nav-link" href="#">پروفایل افرادی</a>
33               </li>
34               <li class="nav-item dropdown">
35                 <a class="nav-link dropdown-toggle" href="#" role="button" data-bs-toggle="dropdown" aria-expanded="false">
36                   فعالیت‌های پژوهشی
37                 </a>
38                 <ul class="dropdown-menu" style="text-align: right">
39                   <li><a href="#">دانش</a></li>
40                   <li><a href="#">پژوهش</a></li>
41                   <li><a href="#">مقالات علمی</a></li>
42                   <li><a href="#">پروژه‌ها</a></li>
43                   <li><a href="#">آیینه انجمن</a></li>
44                 </ul>
45               </li>
46             </ul>
47             <div class="d-flex" role="search">
48               <input class="form-control me-2" type="search" placeholder="جستجو" aria-label="Search">
49               <button class="btn btn-outline-success" type="submit">جستجو</button>
50             </div>
51           </div>
52         </div>
53       </div>
54     </div>

```

۵. آیتم‌های منو را فارسی می‌کنیم:



```

File Edit View Navigate Code Refactor Run Tools VCS Window Help Resume [D:\MRMA\ DjangoProjects\Resume] - layout.html
Resume > researchActivities > templates > researchActivities > layout.html
Project
  - Resume
    - static
      - admin
        - css
        - fonts
        - images
          - mrmaj.jpg
          - writersLogo.jpg
      - templates
  - settings.py
  - urls.py
  - style.css
  - fontawesome.js
  - jquery-3.6.1.min.js
  - admin.py
  - base.css
  - base_site.html
  - layout.html
  - Choose direction Hide notification Don't show again
  - Database
  - Notifications
  - SCView
  - Notifications
  - Terminal: C:\Windows\system32\runserver ...
  - Version Control Python Packages TODO Problems Database Changes Services
  - 14:28 CRLF UTF-8 4 spaces Python 3.10 (venv) (4)
  - 12/9/2022

```

Visual layout of bidirectional text can depend on the base direction (View | Bidi Text Base Direction)

```

28 <ul class="navbar-nav me-auto mb-2 mb-lg-0">
29   <li class="nav-item">
30     <a class="nav-link active" aria-current="page" href="#">خانه</a>
31   </li>
32   <li class="nav-item">
33     <a class="nav-link" href="#">پروفایل افرادی</a>
34   </li>
35   <li class="nav-item dropdown">
36     <a class="nav-link dropdown-toggle" href="#" role="button" data-bs-toggle="dropdown" aria-expanded="false">
37       فعالیت‌های پژوهشی
38     </a>
39     <ul class="dropdown-menu" style="text-align: right">
40       <li><a href="#">دانش</a></li>
41       <li><a href="#">پژوهش</a></li>
42       <li><a href="#">مقالات علمی</a></li>
43       <li><a href="#">پروژه‌ها</a></li>
44       <li><a href="#">آیینه انجمن</a></li>
45     </ul>
46   </li>
47   <li class="nav-item">
48     <a class="nav-link disabled">محمد رضا ملاحی مهندسی اردکانی</a>
49   </li>
50 </ul>
51 <div class="d-flex" role="search">
52   <input class="form-control me-2" type="search" placeholder="جستجو" aria-label="Search">
53   <button class="btn btn-outline-success" type="submit">جستجو</button>
54 </div>

```

۶. برنامه را اجرا می کنیم:

The screenshot shows a web browser with the following details:

- Address Bar:** http://127.0.0.1:8000/researchActivities/papers/list#
- Bookmarks Bar:** Bookmarks, Free Online Gramm..., Check Spelling, Style..., Online Text Correcti..., مدیریت بروزه فاوان, پادگیری ماشین, پایتون, دیکشنری تخصصی, Other bookmarks.
- Page Content:**
 - Header:** جستجو (Search), خانه, بیوگرافی, فعالیتهای آموزشی, فعالیتهای پژوهشی.
 - Profile:** محمد رضا ملاحسینی اردکانی (Portrait).
 - Section:** لیست مقالات (List of Articles).
 - Count: تعداد مقالات من: 4.
 - Thumbnail preview of four articles.
 - Sidebar:** مقالات (Articles), کتابها (Books), اختراعات (Plagiarism).
 - Certificate Preview:** A certificate titled "کارهای پژوهشی" (Research Activities) from "دانشگاه علوم پزشکی اسلامی ایران" (Iran University of Medical Sciences) dated 1400-01-01, issued to "دکتر مهدی سعیدی" (Dr. Morteza Saeidi) for "بررسی روش‌های استخراج جاده از تصاویر دجیتال" (Digital road extraction methods research).

۷. برای اینکه در صورت اسکرول صفحه به سمت پایین، navbar بصورت ثابت بالای صفحه بماند، کلاس fixed-top را به تگ nav اضافه کنید:

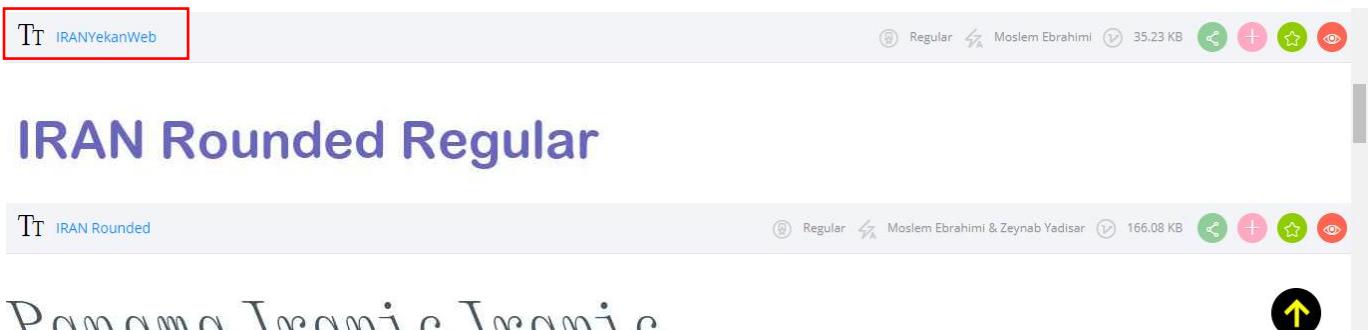
```
<nav class="navbar navbar-expand-lg bg-light fixed-top">
```

استفاده از فونت فارسی

۱. وارد سایت <https://www.onlinewebfonts.com> شوید.

۲. Iran را جستجو کنید.

۳. روی فونت IRANYekanWeb را کپی کنید.



۴. لینک مرتبط با فونت را کپی کنید.

IRANYekanWeb Font Info

PACKAGE : IRANYekanWeb

STYLE : Regular

VERSION : 1.00 October 22, 2016, initial release

TAG'S :

AUTHOR : Moslem Ebrahimi

SIZE : 35.23 KB

Waiting for www.google.com...

@font-face

#1 Add to the head section of web page.

```
<link href="//db.onlinewebfonts.com/c/1510aa80463174b1d0847ab936a70e8e?family=IRANYekanWeb" rel="stylesheet" type="text/css"/>
```

Copyright

#2 Using @import CSS directive, put the following line in add to your css file.(http | https)

```
@import url("//db.onlinewebfonts.com/c/1510aa80463174b1d0847ab936a70e8e?family=IRANYekanWeb");
```

Explanation

#3 Use font-face declaration Fonts.(http | https)

```
@font-face {font-family: "IRANYekanWeb"; src: url("//db.onlinewebfonts.com/t/1510aa80463174b1d0847ab936a70e8e.eot"); src: url("//db.onlinewebfonts.com/t/1510aa80463174b1d0847ab936a70e8e.eot?#iefix") format("embedded-opentype"), url("//db.onlinewebfonts.com/t/1510aa80463174b1d0847ab936a70e8e.woff2") format("woff2"), url("//db.onlinewebfonts.com/t/1510aa80463174b1d0847ab936a70e8e.woff") format("woff"), url("//db.onlinewebfonts.com/t/1510aa80463174b1d0847ab936a70e8e.ttf") format("truetype");}
```

4 Total Downloads 8

4:31 PM 12/16/2022

۴. لینک را در فایل layout.html قرار دهید.

Resume > researchActivities > templates > researchActivities > layout.html

```
<!DOCTYPE html>
{% load static %}
<html lang="en" dir="rtl">
    <head>
        <meta charset="UTF-8">
        <title>{{ block title }} {{ endblock title }}</title>
        <link rel="stylesheet" type="text/css" href="{% static 'researchActivities/css/bootstrap.css' %}">
        <link rel="stylesheet" type="text/css" href="{% static 'researchActivities/css/fontawesome.css' %}">
        <link href="//db.onlinewebfonts.com/c/1510aa80463174b1d0847ab936a70e8e?family=IRANYekanWeb" rel="stylesheet" type="text/css"/>
        <link rel="stylesheet" type="text/css" href="{% static 'researchActivities/css/style.css' %}">
    </head>
    <body>
        <div class="container">
            <div class="row">
                <div class="col-md-12">
                    <nav class="navbar navbar-expand-lg bg-light fixed-top">
                        <div class="container-fluid">
                            <a class="navbar-brand" href="#">
```

Choose direction Hide notification Don't show again

Visual layout of bidirectional text can depend on the base direction (View | Bidi Text Base Direction)

Project: Resume

Structure

File Edit View Navigate Code Refactor Run Tools VCS Window Help

Resume [D:\MRMA\ DjangoProjects\Resume] - layout.html

Terminal: C:\Windows... runserver +

[16/Dec/2022 16:20:37] "GET /media/paperImages/14.jpg HTTP/1.1" 200 292784

[16/Dec/2022 16:20:37] "GET /media/paperImages/4.jpg HTTP/1.1" 200 495307

Not Found: /favicon.ico

[16/Dec/2022 16:20:43] "GET /favicon.ico HTTP/1.1" 404 2593

Version Control Python Packages TODO Python Console Problems Terminal Database Changes Services

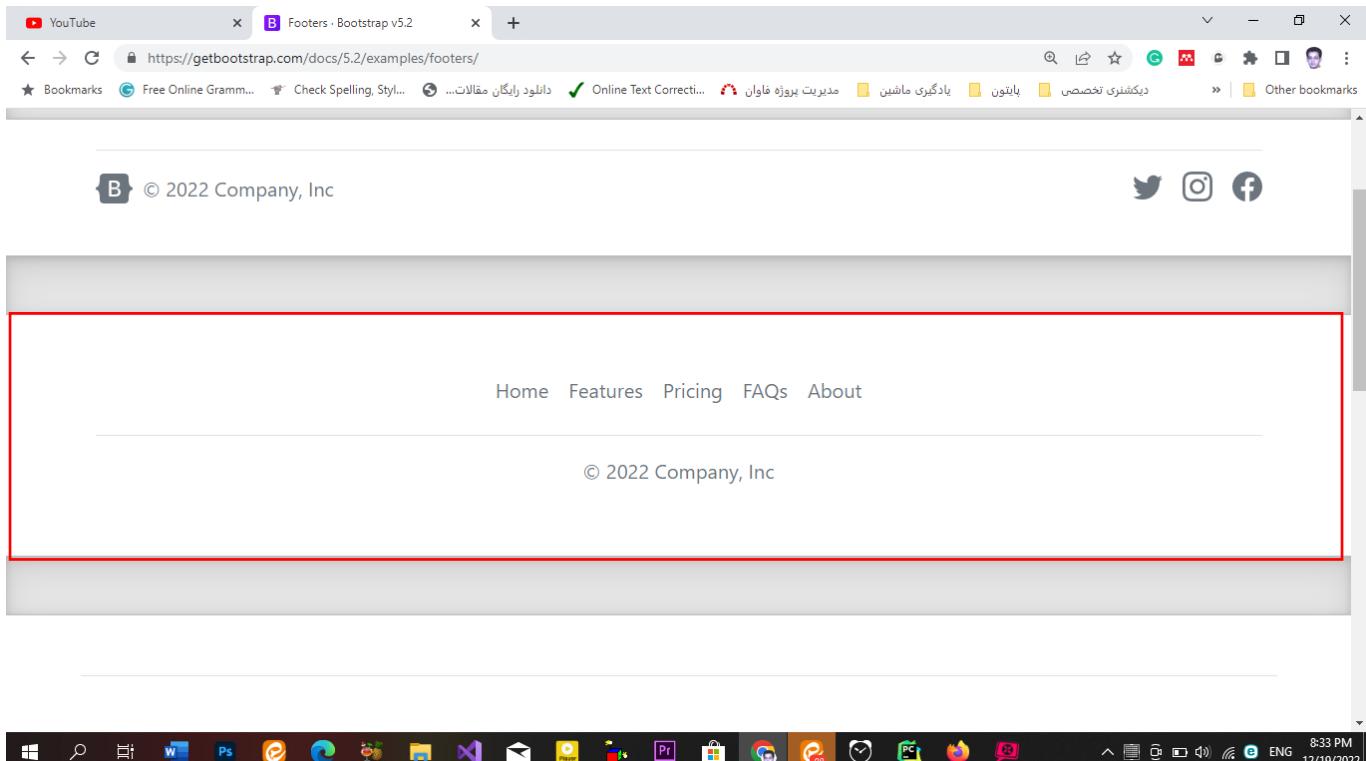
17:48 CRLF UTF-8 4 spaces Python 3.10 (venv) (4) 4:28 PM 12/16/2022

ایجاد فوتر

۱. وارد سایت <https://getbootstrap.com> شوید.

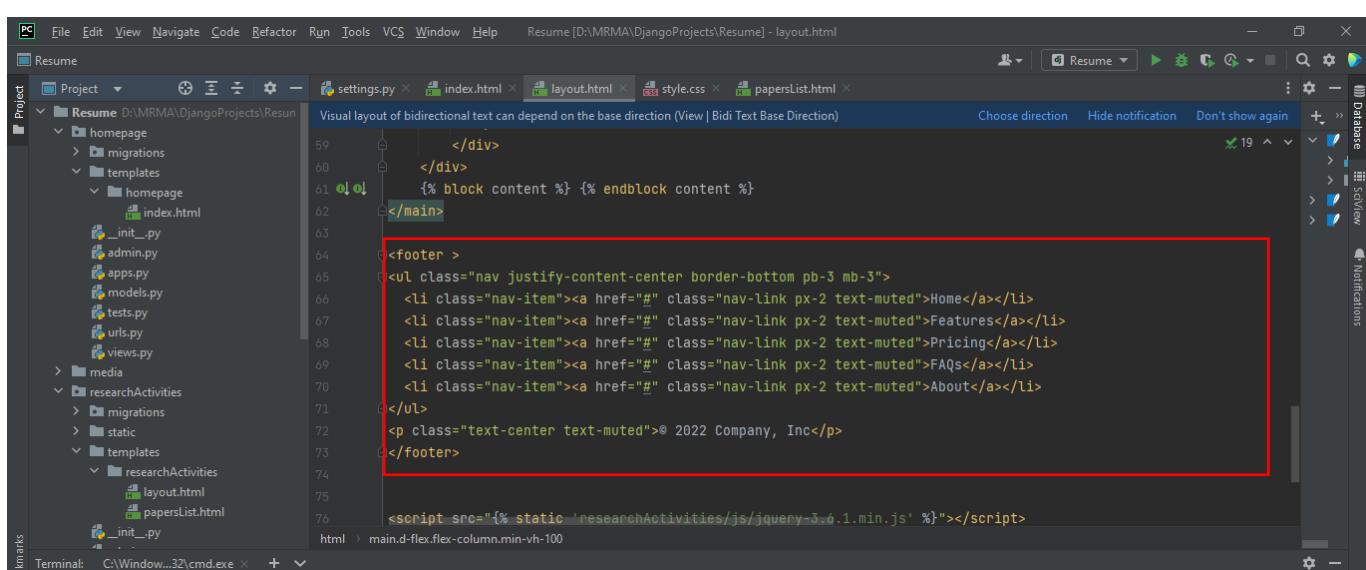
۲. در مسیر <https://getbootstrap.com/docs/5.2/examples/footers> تعدادی فوتر نمونه وجود دارد.

۳. فوتر زیر را را می‌خواهیم در سایت داشته باشیم



۴. روی صفحه راست کلیک کنید و وارد گزینه inspect شوید سپس کد مرتبط با فوتر فوق را کپی کنید.

۵. کد را داخل فایل layout.html قبل از تگ </body> کپی کنید.



۶. برنامه را اجرا کنید. فوتر را داریم:

A screenshot of a web browser window displaying a homepage. The address bar shows the URL <http://127.0.0.1:8000>. The page content includes a navigation bar with links like 'About', 'FAQs', 'Pricing', 'Features', and 'Home'. At the bottom, there is a copyright notice: 'Company, Inc 2022 ©'. A red rectangular border surrounds the main content area of the page.

تمرين: ايجاد آپ جديد با نام [homepage](#)

carousel

استفاده از carousel در هوم پیج سایت

۱. کپی کد carousel مدنظر از سایت [getbootstrap](https://getbootstrap.com/docs/5.2/components/carousel)

The screenshot shows the Bootstrap 5.2 Components documentation page. On the left, there's a sidebar with categories like Dropdowns, List group, Modal, etc. The main content area displays the HTML code for a carousel. At the top right of the content area, there's a "Copy to clipboard" button. To the right of the content area, there's a sidebar titled "On this page" with various links related to the carousel component.

```
<div id="carouselExampleCaptions" class="carousel slide" data-bs-ride="false">
  <div class="carousel-indicators">
    <button type="button" data-bs-target="#carouselExampleCaptions" data-bs-slide-to="0" class="active" aria-current="true" aria-label="Slide 1"></button>
    <button type="button" data-bs-target="#carouselExampleCaptions" data-bs-slide-to="1" aria-label="Slide 2"></button>
    <button type="button" data-bs-target="#carouselExampleCaptions" data-bs-slide-to="2" aria-label="Slide 3"></button>
  </div>
  <div class="carousel-inner">
    <div class="carousel-item active">
      
      <div class="carousel-caption d-none d-md-block">
        <h5>First slide label</h5>
        <p>Some representative placeholder content for the first slide.</p>
      </div>
    </div>
    <div class="carousel-item">
      
      <div class="carousel-caption d-none d-md-block">
        <h5>Second slide label</h5>
        <p>Some representative placeholder content for the second slide.</p>
      </div>
    </div>
    <div class="carousel-item">
      
      <div class="carousel-caption d-none d-md-block">
        <h5>Third slide label</h5>
        <p>Some representative placeholder content for the third slide.</p>
      </div>
    </div>
  </div>

```

۲. قرار دادن کد داخل بلاک content از فایل index.html

The screenshot shows the PyCharm IDE interface. The project structure on the left shows a "Resume" project with a "homepage" directory containing "index.html". The code editor on the right displays the content of "index.html". A specific section of the code, which is the copied carousel code, is highlighted with a red box. The terminal at the bottom shows command-line output related to static file collection.

```
<% block title %> و سایت شخصی مادرخانی
<% endblock title %>
<% block content %>
  <div id="carouselExampleCaptions" class="carousel slide" data-bs-ride="false">
    <div class="carousel-indicators">
      <button type="button" data-bs-target="#carouselExampleCaptions" data-bs-slide-to="0" class="active" aria-current="true" aria-label="Slide 1"></button>
      <button type="button" data-bs-target="#carouselExampleCaptions" data-bs-slide-to="1" aria-label="Slide 2"></button>
      <button type="button" data-bs-target="#carouselExampleCaptions" data-bs-slide-to="2" aria-label="Slide 3"></button>
    </div>
    <div class="carousel-inner">
      <div class="carousel-item active">
        
        <div class="carousel-caption d-none d-md-block">
          <h5>اروپاتان همراه با مادرخانی</h5>
          <p>معرفت مهندسی ایرانی مادرخانی</p>
        </div>
      </div>
      <div class="carousel-item">
        
        <div class="carousel-caption d-none d-md-block">
          <h5>عنوان های خاص و معرفت مهندسی ایرانی مادرخانی</h5>
          <p>معرفت مهندسی ایرانی مادرخانی</p>
        </div>
      </div>
      <div class="carousel-item">
        
        <div class="carousel-caption d-none d-md-block">
          <h5>دانشجویی مادرخانی</h5>
          <p>دانشجویی مادرخانی</p>
        </div>
      </div>
    </div>
  </div>
```

۳. کپی سه تصویر مدنظر داخل ساختار دایرکتوری زیر:

```

File Edit View Navigate Code Refactor Run Tools VCS Window Help Resume [D:\MRMA\ DjangoProjects\Resume] - views.py
Project Research D:\MRMA\ DjangoProjects\Resume
  > homepage
    > migrations
    > static
      > homepage
        > images
          1.jpg
          2.jpg
          3.jpeg
    > templates
      __init__.py
      admin.py
      apps.py
      models.py
      tests.py
      urls.py
      views.py
  > media
  > researchActivities
    > migrations
      > static
        > templates
          paperListView.html
  Terminal: C:\Windows\cmd.exe

```

```

def paperListView(request):
    SearchForm = searchForm(request.GET)
    if SearchForm.is_valid():
        searchText = SearchForm.cleaned_data["searchText"]
        papers = Paper.objects.filter(title__contains=searchText)
    else:
        papers = Paper.objects.all()

    context = {
        "papers": papers,
        "papersCount": papers.count(),
        "SearchForm": SearchForm,
    }

    return render(request, "researchActivities/papersList.html", context)

```

۴. اجرای کد زیر در ترمینال

`python manage.py collectstatic`

فرم در جنگو

در HTML، فرم مجموعه‌ای از عناصر درون `<form>...</form>` است که به بازدیدکننده اجازه می‌دهد کارهایی مانند وارد کردن متن، انتخاب گزینه‌ها، دستکاری اشیاء یا کنترل‌ها و غیره را انجام دهد و سپس آن اطلاعات را به سرور ارسال کند.

برخی از این عناصر رابط فرم - ورودی متن یا چک باکس ها - نسبتاً ساده هستند و در خود HTML ساخته شده‌اند. دیگران بسیار پیچیده‌تر هستند.

فرم علاوه بر عناصر `<input>` آن، باید دو چیز را مشخص کند:

- ✓ کجا: آدرس اینترنتی که داده‌های مربوط به ورودی کاربر باید به آن برگردانده شود
- ✓ چگونه: روش HTTP که داده‌ها باید توسط آن برگردانده شوند

POST و GET

GET و POST تنها روش‌های HTTP هستند که هنگام کار با فرم‌ها استفاده می‌شوند.

- ✓ فرم login جنگو با استفاده از روش POST بازگردانده می‌شود، که در آن مرورگر داده‌های فرم را بسته بندی می‌کند، آن را برای انتقال رمزگذاری می‌کند، آن را به سرور ارسال می‌کند و سپس پاسخ آن را دریافت می‌کند.
- ✓ در مقابل، GET داده‌های ارسالی را در یک رشته دسته‌بندی می‌کند و از آن برای ایجاد URL استفاده می‌کند. URL حاوی آدرسی است که داده‌ها باید در آن ارسال شوند همچنین کلیدهای داده‌ای و مقادیر. مثال از یک URL:

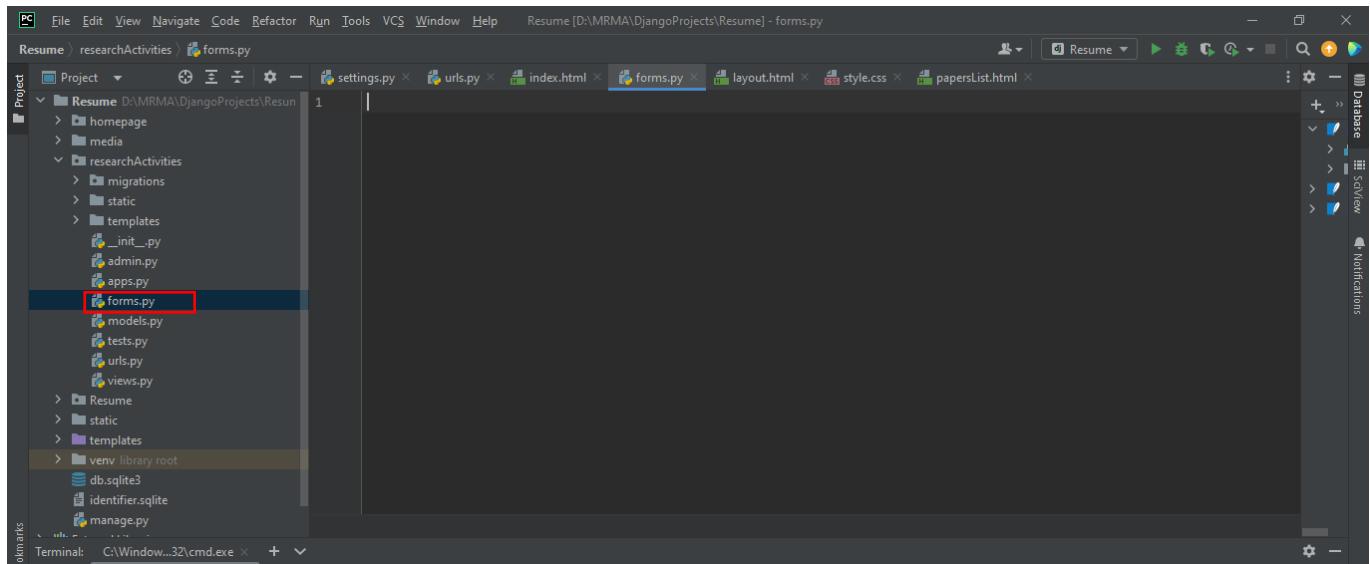
<https://docs.djangoproject.com/search/?q=forms&release=1>

GET و POST معمولاً برای اهداف مختلف استفاده می‌شوند:

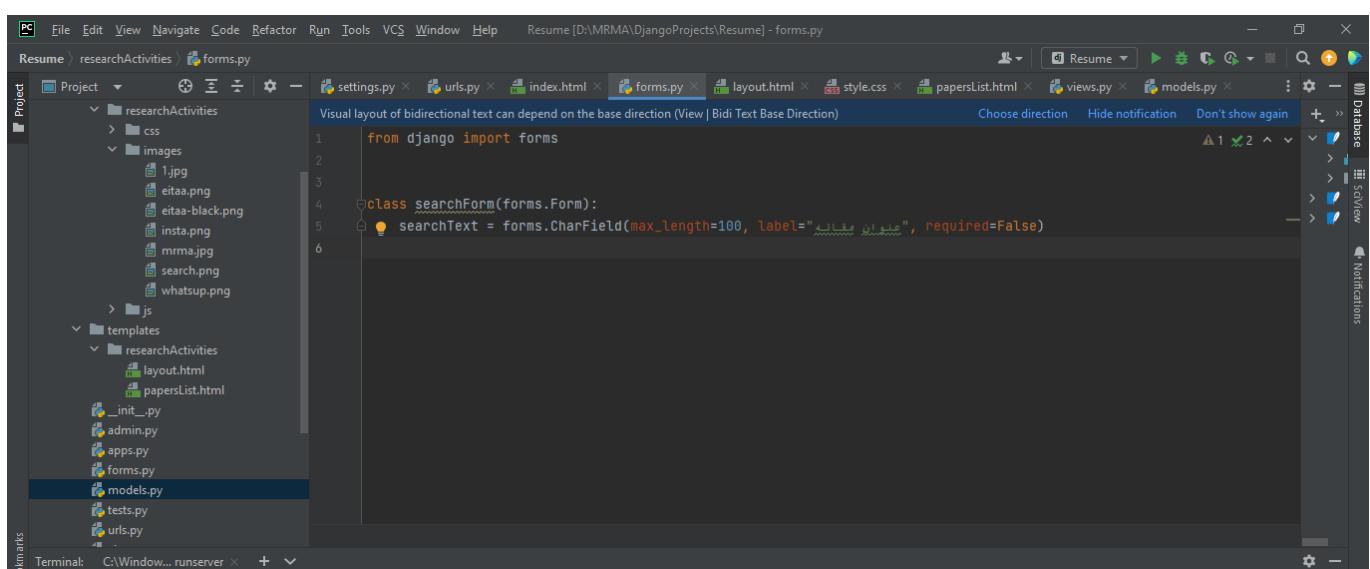
- ✓ هر درخواستی که می‌تواند برای تغییر وضعیت سیستم استفاده شود - برای مثال، درخواستی که تغییراتی را در پایگاه داده ایجاد می‌کند - باید از POST استفاده کند.
- ✓ GET باید فقط برای درخواست‌هایی استفاده شود که بر وضعیت سیستم تأثیری ندارند.
- ✓ GET همچنین برای فرم حاوی رمز عبور نامناسب است، زیرا رمز عبور در URL و بنابراین، همچنین در تاریخچه مرورگر و گزارش‌های سرور، همه به صورت متن ساده ظاهر می‌شود.
- ✓ GET برای مقادیر زیاد داده یا برای داده‌های باینری، مانند تصویر، مناسب نیست.
- ✓ یک برنامه وب که از درخواست‌های GET برای فرم‌های ادمین استفاده می‌کند، یک خطر امنیتی است: می‌تواند برای مهاجم آسان باشد که درخواست یک فرم را برای دسترسی به بخش‌های حساس سیستم تقلید کند.
- ✓ POST، همراه با سایر حفاظت‌ها مانند حفاظت CSRF جنگو، کنترل بیشتری بر دسترسی ارائه می‌کند.
- ✓ از سوی دیگر، GET برای مواردی مانند فرم جستجوی وب مناسب است، زیرا URL‌هایی که یک درخواست GET را نشان می‌دهند می‌توانند به راحتی نشانک‌گذاری شوند، به اشتراک گذاشته شوند یا دوباره ارسال شوند.

ایجاد فرم جستجو

۱. ایجاد یک فایل با نام `researchActivities` داخل آپ `forms.py`



۲. ایمپورت کتابخانه `forms.py` به فایل `forms.py`، سپس تعریف کلاس فرم موردنظر با ویژگی‌های دلخواه (آنچه قرار است در فرم جستجو نشان داده شود):



نکته: چون فرم جستجو به هیچ مدلی وابسته نیست از `forms.Form` ارث بری صورت می‌گیرد.

۳. پاس دادن فرم از طریق ویوی تولید کننده تمپلیت `paperList.html` به تمپلیت:

۴. نمایش باکس فرم جستجو در فایل paperList.html

```

from django.shortcuts import render
from researchActivities.models import Paper
from researchActivities.forms import searchForm

# Create your views here.

def paperListView(request):
    SearchForm = searchForm()
    papers = Paper.objects.all()
    context = {
        "papers": papers,
        "papersCount": papers.count(),
        "SearchForm": SearchForm,
    }
    return render(request, "researchActivities/paperList.html", context)

```

۵. برنامه را اجرا کنید:

```

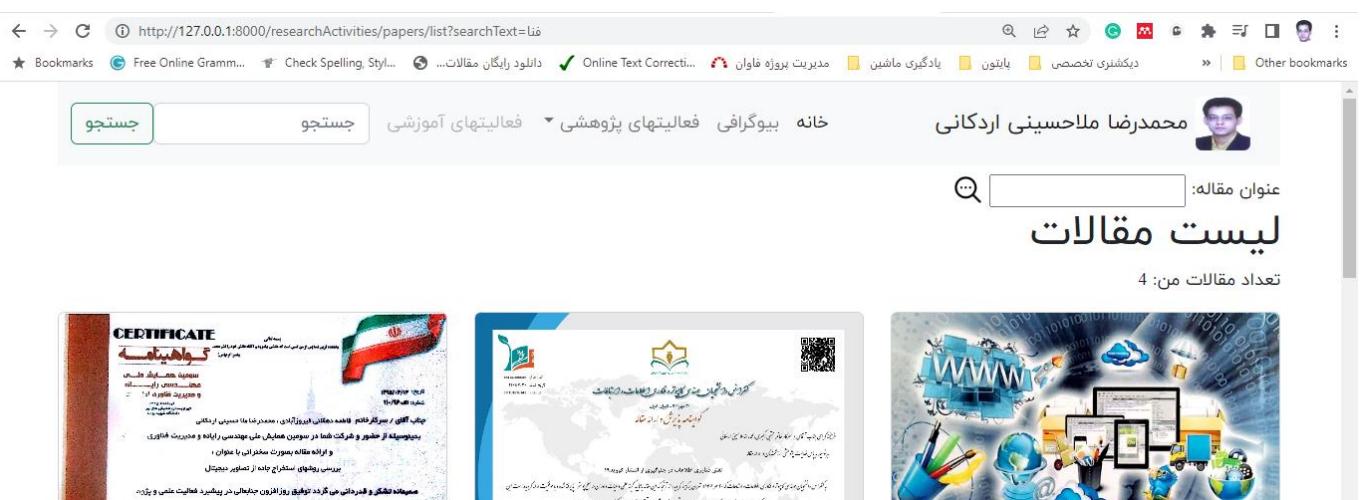

<form action="" method="get">
    {{ SearchForm }}
    <a href=""></a>
</form>



# لیست مقالات


{% if papers %}
    <p>تعداد مقالات: {{ papersCount }}</p>
    <div class="row">
        {% for paper in papers %}


```



۶. دریافت اطلاعات فرم در ویو و ...

The screenshot shows the PyCharm IDE interface. The top navigation bar includes File, Edit, View, Navigate, Code, Refactor, Run, Tools, VCS, Window, Help, and Resume. The current file is views.py, located at D:\MRMA\ DjangoProjects\Resume. The left sidebar displays the project structure under 'Project', showing files like settings.py, urls.py, index.html, forms.py, layout.html, style.css, papersList.html, and models.py. Below the project tree is the code editor with the following content:

```
def paperListView(request):
    SearchForm = searchForm(request.GET)
    if SearchForm.is_valid():
        searchText = SearchForm.cleaned_data["searchText"]
        papers = Paper.objects.filter(title__contains=searchText)
    else:
        papers = Paper.objects.all()

    context = {
        "papers": papers,
        "papersCount": papers.count(),
        "SearchForm": SearchForm,
    }

    return render(request, "researchActivities/papersList.html", context)
```

The code editor has syntax highlighting and several yellow circular markers indicating potential issues or warnings. A status bar at the bottom shows the terminal path as C:\Windows... runserver.

نکته: متدهای `is_valid()` برای انجام اعتبارسنجی برای هر فیلد فرم استفاده می‌شود که در کلاس Django Form تعریف شده است. اگر داده‌ها معتبر باشند، `True` را برمی‌گرداند و همه داده‌ها را در ویژگی `cleaned_data` قرار می‌دهد.

cleaned_data حاوی یک کلید برای فیلد های تعریف شده در فرم است

۷. بیانیه را اجرا و نتیجه را بینید