

به نام خدا

پروژه دوره گردانی

موضوع : سیستم جامع اطلاعات استانی

ارئه شده به

گروه کامپیوتر

دانشکده شهید شمسى پور

به راهنمایی

استاد

محمد مهدى آهنگرى

توسط دانشجو

محمد رسول طارمى

به شماره دانشجوئى

۹۶۱۱۱۰۳۳۱۵۴۰۳۶

تاریخ دفاع

۹۸/۰۶/



در شروع این مستند باید این را ذکر کنم که این پروژه با استفاده از زبان برنامه نویسی سطح بالای پایتون (python) نگارش شده.

سیستمی طراحی شده که در این سیستم از طریق یکپارچه سازی و جامع سازی مکان های گردش

گری ایران از طریق هدف مند سازی صنعت توریسم به شکوفایی و شناخت هر چه بیشتر مکان های گردش گری ایران کمک شود. در این سیستم شما میتوانید بر اساس سلیقه یا استان مورد نظر خودتان منطقه را پیدا و اطلاعاتی در مورد آن کسب کنید تا با دانشی کامل تر به سفر بروید. شما میتوانید با انتخاب استان مورد نظر در صفحه ی اول و فیلتر کردن نوع مکان گردش گری که میتواند مذهبی، تاریخی، گردشگری، طبیعی یا انواع و اقسام مکان ها که خودتان میتوانید به انها اضافه کنید مکان مورد نظر را پیدا کرده یا با مکان های دیگر مقایسه کنید هدف از ساخت این پروژه جمع آوری و یکپارچه سازی تمام مکان های تاریخی، تفریحی، گردش گری، توریستی، مذهبی و... به طوری که هر شخص به تواند با در نظر گرفتن استان و نوع مکانی مورد نظر به راحتی به تمامی اطلاعات آن مکان دسترسی داشته باشد.

مراحل انجام شده برای ساخت و تکمیل این پروژه به این ترتیب است طراحی پایگاه داده، کد نویسی بخش کنترل ها یا بخش پشتی، طراحی ظاهر و در نهایت تست در بخش

میزان کار انجام شده تا این پروژه به سرانجام برسد معادل ۶-۷ روز بده که بخش اعظم آن مربوط به بخش طراحی و نیاز سنجی بود این زمان معادل ۳-۴ روز بود و بخش کد نویسی و پیاده سازی طراحی های انجام شده در مجموع ۳-۴ روز بوده.

- [چکیده](#)
- [پیشگفتار](#)
- [شرح کامل و مفصل](#)
- [ساختمان داده](#)
 - [پایگاه داده در سیستم](#)
 - [طراحی جداول](#)
 - [طراحی جدول استان ها](#)
 - [طراحی جدول موضوع ها](#)
 - [طراحی جدول عکس ها برای محل](#)
 - [طراحی جدول محل ها](#)
 - [طراحی جدول رابط](#)
- [دستورات](#)
 - [تمپلیت ها](#)
 - [خواندن فایل به صورت ایستا](#)
 - [متد ها](#)
 - [مسیریابی](#)
 - [تنظیمات پنجره مدیریت](#)
 - [نمایش اطلاعات در صفحه](#)
- [چگونگی نصب](#)
 - [حداقل سیستم](#)
 - [نرم افزار و کتابخانه های ضروری](#)
 - [نرم افزار های غیر ضروری ولی مهم](#)
- [راهنمای کاربردی](#)
- [پیشنهاد و نتیجه گیری](#)
- [ضمائم](#)
- [منابع و مآخذ](#)

قبل از شروع درباره خود پروژه تصمیم دارم کمی درباره ی جنگو و برنامه نویسی جنگو صحبت بکنم. همان طور که توضیح داده شده جنگو یکی از بزرگترین کتابخانه های زبان قدرت مند پایتون است که در کمک رسانی هر چی بیشتر برنامه نویس در طراحی هر چه آسان تر یک وبسایت روز به روز در حال قوی تر شدن است.

اما نقش برنامه نویس کوچکی چون من در این طراحی و توسعه چیست؟

اگر با CMS های طراحی وبسایت آشنا باشید حتما با وردپرس آشنایی دارید. وردپرس یک رابط بین کاربر و کد ها است که با استفاده از چند کلیک میتوانید یک وبسایت را کاملا بالا بیاورید.

اما جنگو وردپرس نیست، بلکه شباهات بسیار زیادی به آن دارد. این شباهت ها به این معنا نیست که شخص هیچ گونه کدی نمیزد، بلکه جنگو بسیاری از کارهای روتینی که هر برنامه نویس مجبور است در هر پروژه تکرار کند به طوری که آن کد ها هیچ فرقی با هم ندارند را جمع آوری کرده و از قبل در قالب یک پروژه نیمه آماده در اختیار ما قرار میدهد

این کار های روتین که لازم نیست همه آن را فرا بگیرند یا در هر پروژه آن را از اول شروع به نوشتن بکنند عبارت اند از

web sockets : برای این که یه پروژه تحت وب اجرا شود برنامه نویس حتما باید وب سوکت را بلد باشد تا با استفاده از پورت های شبکه خط انتقال دیتا را کنترل کند.

templates : برای کوتاه کردن و خوانا کردن هر چه بیشتر کد ها برنامه نویس مجاب است این بخش را خودش پیاده سازی کند که کاری بسیار سخت و دشوار است.

Data Base : برای ساخت هر گونه وب سایتی حداقل به چند جدول یا رابط نیاز منید که این ها هم باید توسط کاربر طراحی شود.

و...

تمامی این مواردی بخشی از قدرت مندی جنگو است به نحوی که تمام آن ها توسط برنامه های بسیار خلاق در سرار دنیا به بهترین نحو انجام شده و ما کفایت از آنها استفاده کنیم.

با توجه به توضیحات بالا ممکن است در کد ها قطعه کدی باشد که دقیقاً ندانم چه کار خاصی را انجام میدهند ولی به طور کلی میدانم چه نوع فعالیتی انجام میدهد.

در اینجا توضیح مختصری از پایتون گفته میشود و در ادامه با تعریف کامل تری از پایتون آشنا میشویم.

یک زبان برنامه نویسی همه منظوره، سطح بالا، شیءگرا، اسکریپتی و متن باز است که توسط خیدو فان روسوم در سال ۱۹۹۱ در کشور هلند طراحی شد. فلسفه ایجاد آن تأکید بر دو هدف اصلی خوانایی بالای برنامه های نوشته شده و کوتاهی و بازدهی نسبی بالای آن است. کلمات کلیدی و اصلی این زبان به صورت حداقلی تهیه شده اند و در مقابل کتابخانه هایی که در اختیار کاربر است بسیار وسیع هستند.

بر خلاف برخی زبان های برنامه نویسی رایج دیگر که بلاک های کد در آکولاد تعریف می شوند (به ویژه زبان هایی که از گرامر زبان سی پیروی می کنند) در زبان پایتون از نویسه فاصله و جلو بردن متن برنامه برای مشخص کردن بلاک های کد استفاده می شود. به این معنی که تعدادی یکسان از نویسه فاصله در ابتدای سطرهای هر بلاک قرار می گیرند، و این تعداد در بلاک های کد درونی تر افزایش می یابد. بدین ترتیب بلاک های کد به صورت خودکار ظاهری مرتب دارند.

پایتون مدل های مختلف برنامه نویسی (از جمله شیءگرا و برنامه نویسی دستوری و تابع محور) را پشتیبانی می کند و برای مشخص کردن نوع متغیرها از یک سامانه پویا استفاده می کند.

این زبان از زبان های برنامه نویسی مفسر بوده و به صورت کامل یک زبان شیءگرا است که در ویژگی ها با زبان های تفسیری پرل، روبی، اسکیم، اسمال تاکو تی سی ال مشابهت دارد و از مدیریت خودکار حافظه استفاده می کند.

پایتون پروژه ای آزاد و متن باز توسعه یافته است و توسط بنیاد نرم افزار پایتون مدیریت می گردد.

جنگو (Django) یک فریم ورک سطح بالا، رایگان و ¹Open Source برای ساخت Web Application ها می باشد که در پایتون نوشته شده است. با استفاده از فریم ورک جنگو می توانید وب اپلیکیشن های خود را آسان و سریع تر توسعه دهید. هدف از ساخت این فریم ورک اتصال اجزای مشابه سایت است که به دیتابیس نیاز دارند مانند: ثبت نام، ورود و خروج از سیستم، پنل مدیریت، فرم ها، آپلود فایل ها و غیره. با استفاده از فریمورک جنگو نیازی به نوشتن کد های اضافی ندارید.

فریمورک جنگو بسیاری از ویژگی های پایتون را به ارث برده است. این فریم ورک قادر به ساخت وب سایت های پیچیده و حرفه ای در سریع ترین زمان و با امنیت بالا می باشد.

فریمورک جنگو از ساختار Model-View-Controller تبعیت می کند به همین دلیل کد های مربوط به بخش های کنترلی (Controller)، بخش داده ها (Model) و بخش مربوط به رابط کاربری (View) از هم جدا هستند



¹ به صورت کد باز در سطح اینترنت وجود دارد تا تمام برنامه نویسان روی آن کار کنند.

یکی از ابزارهای ذخیره و بازیابی اطلاعات، SQLite نام دارد. این نرم افزار مشهورترین سیستم ذخیره فایلی اطلاعات به شمار می رود. شهرت SQLite به دلیل پشتیبانی از انواع مختلف سیستم عامل ها از جمله ویندوز، لینوکس، آندروئید و مک او اس و همچنین رایگان و قدرتمند بودن آن است.

SQLite با زبان C برنامه نویسی شده است و به طور پیوسته در حال بهبود و توسعه است. به همین دلیل سرعت و کارایی بسیار بالایی دارد. در نگارش های جدید که در آینده منتشر خواهند شد، بهینه سازی های گسترده ای روی این سیستم به انجام رسیده است که سرعت عملکرد آن را بیش از پیش افزایش داده است.

پایگاه داده در سیستم

ساختمان داده یا به طور دقیق تر تعریف ساختمان داده در جنگو با دیگر زبان ها و کتابخانه ها فرق های بزرگ و اساسی دارد حداقل تا آنجایی که من میدانم.

به این شکل **code first** است ، به این طریق که شما تمام جداول و **relation** های مورد نیاز خود را نوشته و جنگو است که جداول را میسازد. البته این طریق برای استفاده از دیتابیس **sqlite** است، ولی شما مثل تمام زبان های دیگر مختارید در انتخاب دیتابیس.

بعد از طراحی نمونه اولیه از **structure** آدیتا بیس وقت آن است که تبدیل به جداول شوند

برای پروژه سیستم اطلاعات جامع استانی بعد از انجام مراحل بررسی موجودیت ها تصمیم بر آن شد که شامل ۴ جدول اصلی که شامل

✓ استان ها

✓ محل ها

✓ عکس ها

✓ و موضوع ها

و یک جدول رابط که رابط بین عکس ها و محل ها بود ساخته شود

به بررسی دقیق تر روی تک تک موجودیت ها باید صفات آنها نیز درست میشد تا در آینده به مشکلی

برنخوریم.البته جدوالی جنگو میسازد هم ۵ تا جدولی که ما میخوایم نمیسازد

جنگو بالغ بر ۱۱ جدول دیگر به صورت اتوماتیک میسازد. این جداول شامل جداول

✓ کاربران

✓ گروه ها

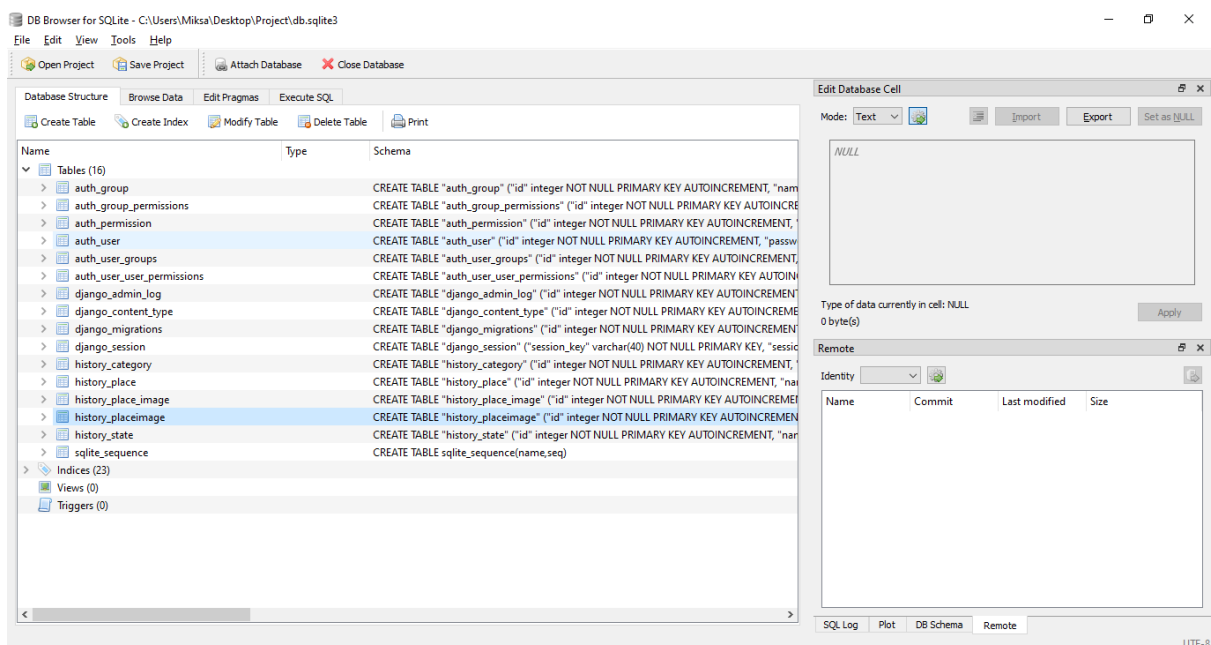
✓ دسترسی ها

✓ لاگی ہا

✓ سیشن ها

... ✓

برای همین موارد است که جنگو در ساخت یک پروژه بسیار سریع و کارآمد است. به طوری که برنامه نویس نیازی به طراحی دیتا بیس برای کاربران یا ... نیست و فقط کافیه از آنها استفاده کند یا در مسائل تخصصی تر آنها را ویرایش کند، چون جنگو تمام آن چیزی که ممکن است برنامه نویس به آن برخورد را پیش بینی و لحاظ کرده.



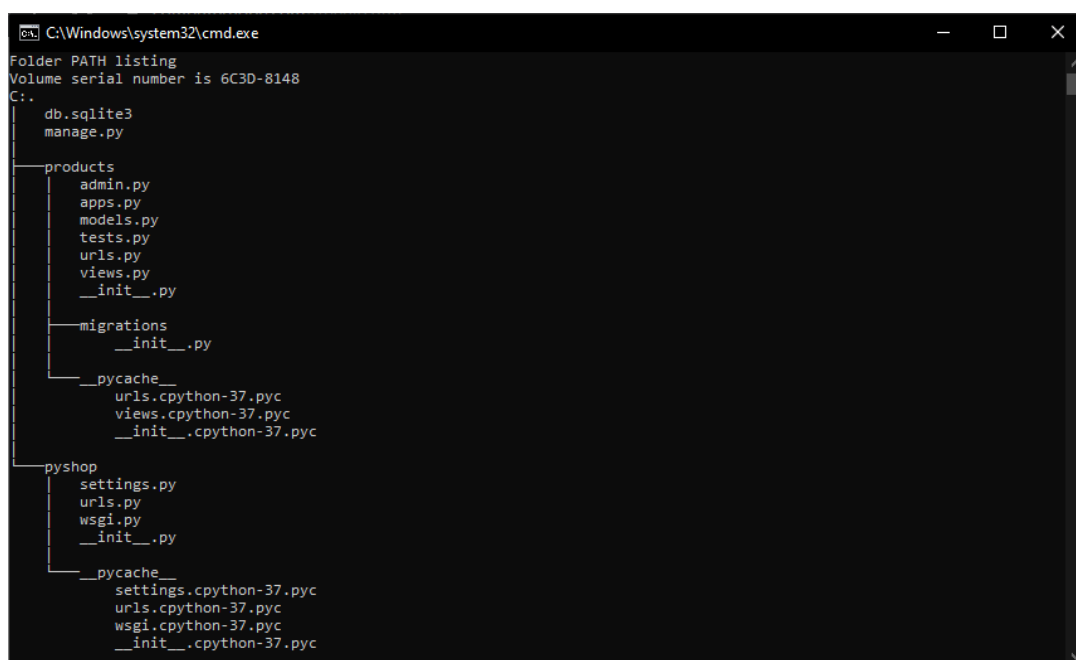
در تصویر بالا لیستی از تمام جدول ساخته شده توسط جنگو را میبینیم که شامل جدول طراحی شده برای پروژه نیز است.

طراحی جداول

همان طور که گفته شد طراحی جداول در جنگو متاول به بقیه زبان ها یا کتاب خانه ها نیست، به این صورت که با زدن کل های پایتون سیستم برای شما جدول درست میکند نه دستورات T-SQL

خب پس پیدا کردن موجودیت ها و صفات آنها شروع به کد زدن میکنیم اما کجا باید کد زد؟؟؟

وقتی یک پروژه با جنگو شروع میکنید، جنگو در دایرکتوری تعدادی فایل میسازد که فایل در جای خود کار خاصی میکند که بعدا تک تک آن ها را توضیح خواهیم داد

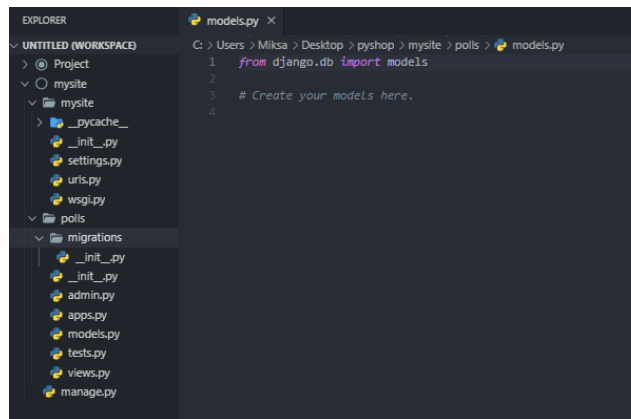


```
C:\Windows\system32\cmd.exe
Folder PATH listing
Volume serial number is 6C3D-8148
C:.
  db.sqlite3
  manage.py
  products
  |   admin.py
  |   apps.py
  |   models.py
  |   tests.py
  |   urls.py
  |   views.py
  |   __init__.py
  |
  |   migrations
  |   |   __init__.py
  |   |
  |   __pycache__
  |   |   urls.cpython-37.pyc
  |   |   views.cpython-37.pyc
  |   |   __init__.cpython-37.pyc
  |
  |   pyshop
  |   |   settings.py
  |   |   urls.py
  |   |   wsgi.py
  |   |   __init__.py
  |   |
  |   |   __pycache__
  |   |   |   settings.cpython-37.pyc
  |   |   |   urls.cpython-37.pyc
  |   |   |   wsgi.cpython-37.pyc
  |   |   |   __init__.cpython-37.pyc
```

در تصویر بالا تعداد فایل و دایرکتوری که جنگو به صورت اتوماتیک میسازد را میبینیم ولی فعلا در این بخش قصد توضیح همه این عناوین رو نداریم

خوب همان طور که گفته شد قصد بر این بود که جداول دیتابیس را طراحی کنیم، فایلی که با آن کار داریم در دایرکتوری یکی از app های است که ساختیم به نام **models.py**

در بخش تعریف جنگو نیز گفته شده بود که جنگو مبتنی بری MVC (Model-view-controller) است که به این معناست که کد های هر بخش از یکی دیگر جدا قرار دارند



تصویر مربوط به صفحه models.py بعد از شروع پروژه است

ساخت جدول استان ها

خب همان طور که گفته شد برای درک بهتر موجودیت ها باید سعی در طراحی هر چی بهتر صفات بشود به طوری که جامعیت اطلاعاتی داشته باشیم و از افزونگی پرهیز کند.

با همه این تفاسیر برای موجودیت استان در سیستم اطلاعات جامع نیاز مند چند صفت است با استفاده از ارتباطات سعی در جامعیت اطلاعات میکنیم که در اینجا آن صفات را بررسی میکنیم.

Id: این صفت برای یکایی هر استان طراحی شده به طوری که به صورت یکتا در سیستم ذخیره شود.

این صفت کلید اصلی P.K این جدول به شمار می آید. و درموجودیت های بعدی از آن به عنوان کلید خارجی یعنی F.K استفاده میشود.

این صفت بصورت خودکار برای همه جداول به صورت پیش فرض توسط جنگو گذاشته شده بنابراین نیازی به تغییر یا تعریف موردی در سیستم یا کد نیست.

Name: این نیز برای قرار گرفتن نام استان طراحی شده

این صفت باید نام پارسی استان هارا در خود ذخیره کند پس رد موقع تعریف باید ویژگی به این صفت داده شود که قابلیت ذخیره سازی به کلمات و حروف پارسی را دارا باشد.همچنین این صفت نیاز مند

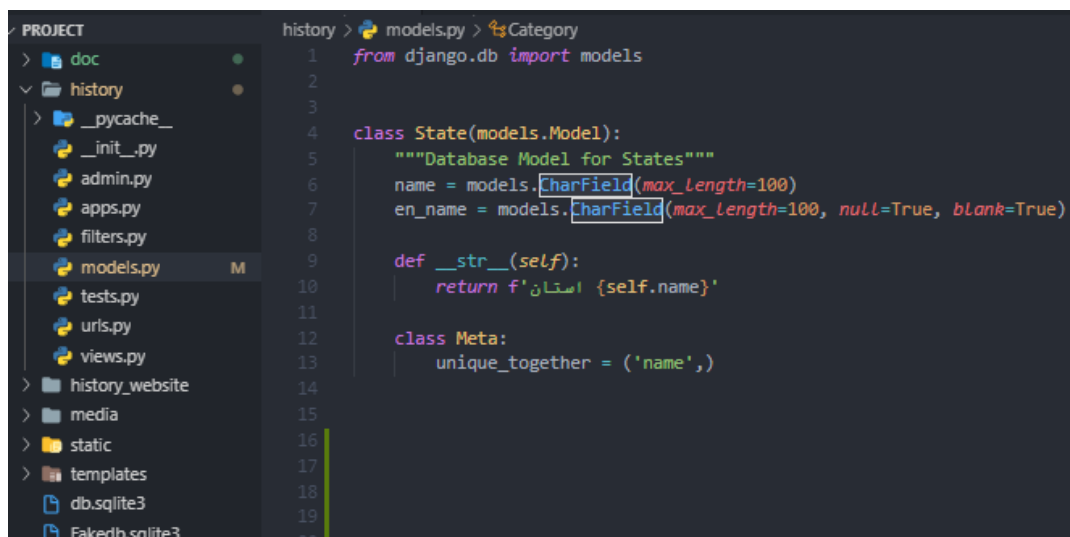
ذخیره سازی به صورت رشته های بلند نیست، زیرا فقط نام یک استان در آن ذخیره میشود مانند: گیلان، کهگیلویه و بویراحمد و...

En-name: روشی که بنده در فیلتر کردن اطلاعات داشتیم از طریق **url** سایت است و در این بخش نمیتوان تصمیم گرفتیم نام استان به صورت انگلیسی باشد.

البته قادر بودم در این بخش از **id** نیز استفاده کنم ولی برای درک بهتر کاربر از برنامه و قوی تر شدن **UX** و همچنین خوانا تر شدن **url** سایت تصمیم بر این شد که از **en-name** استفاده شود.

این صفت نیز شباهت های با صفت **name** دارا است و میتوان از همان ویژگی ها استفاده کرد.

خوب در این مرحله شروع به کد زدن در بخش **models.py** میکنیم.



```
PROJECT history > models.py > Category
1 from django.db import models
2
3
4 class State(models.Model):
5     """Database Model for States"""
6     name = models.CharField(max_length=100)
7     en_name = models.CharField(max_length=100, null=True, blank=True)
8
9
10 def __str__(self):
11     return f'استان {self.name}'
12
13 class Meta:
14     unique_together = ('name',)
15
16
17
18
19
20
```

باری شروع با کلاسی نوشته شود که میخوايد نام جدولمان باشد. بنابراین کلاس در اینجا همان جدول که میخوايد در دیتابیس ساخته شود و مغیر های این جداول همان صفات جداول هستند و **magic function** های نیز نوع عملکرد جداول.

خب میخوايم همان طور که گفته شد باید صفت **name** طراحی شود، با فراخوانی کتابخانه **models** میتوان ویژگی صفت خود را انتخاب کنید، در اینجا چون نیاز به ذخیره سازی حروف پارسی بود از متد **CharField** استفاده شده تا تمام کارکترها بدون مشکل ذخیره شوند.

در این صفت نیاز مند ذخیره سازی رشته های بلند نبوده پس تصمیم گرفتم طول کارکتر ها قابل ذخیره شدن را به ۱۰۰ کارکتر محدود کنم.

نوبت صفت `en-name` است، این صفت همانند صفت `name` دارای ویژگی های مشترک است از جمله طول یا متد ذخیره سازی تنها فرقی که با `name` داراست این است که صفت میتواند خالی باشد، که در این صورت سیستم به جای استفاده از `en-name` در `url` از همان `id` که بصورت خودکار پر میشود استفاده میکند. بنابراین ویژگی هایی `null` و `blank` را مساوی با `true` قرار دادم.

حال به `magic function` های میرسیم، قبل از شروع این `function` ها مجبورم توضیحات مختصری درباره `magic function` های بدهم تا با اصل قضیه آشنا شوید.

مجیک فانکشن ها، فانکشن های از پیش تعریف شده ی خود و پایون است که کار های بخصوصی انجام میدهند. این نوشتن کار ها ممکن است برای برنامه نویس کمی طولانی و بدون نتیجه باشد پس در `base` زبان پایتون این `function` ها تعریف شده.

در اینجا قصد در توضیح همه فانکشن ها ندارم ولی آنهایی که استفاده شده اند را تا جایی که بلد باشم توضیح میدهم. در ادامه لیستی از تمام مجیک فانکشن های مهم پایون میگذارم.




`__str__` ✓
`__init__` ✓
`__repr__` ✓
`__unicode__` ✓
`__format__` ✓
`__hash__` ✓
`__dir__` ✓

فانکشن `__str__` : این فانکشن به این صورت عمل میکند که هر جا نمونه ای این یک کلاس ساخته شد مقادر بازگشتی همان یک متغیر به صورت رشته ای یک رشته قابل تغییر و تعریف باشد در جدول استان ها تصمیم گرفتم رشته ای به شکل زیر باز گرداند

استان + `name`

به عنوان مثال : اگر در سیستم تهران ذخیره شده باشد هر جا که از `state` استفاده شود، مقدار "استان تهران" باز میگردد.

کلاس meta : این کلاس هم مانند همان magic fuction ها عمل میکند که در اینجا صفت name از طریق این کلاس یکتا کردم.

history_state			CREATE TABLE "history_state" ("id" integer NOT NULL PRI
 id	integer	"id" integer NOT NULL PRIMARY KEY AUTOINCREMENT	
 name	varchar(100)	"name" varchar(100) NOT NULL	
 en_name	varchar(100)	"en_name" varchar(100)	

در تصویر بالا جدول ساخته شده توسط جنگو را در دیتابیس با فیلد های طراحی شده میبینیم




نکته: اسم جدول در بالا گفته شده state است ولی در دیتابیس میبینیم ذخیره شده history_state این به این دلیل است که هر پروژه جنگو ممکن است شامل چندین app باشد که هر app به معنا سیستم های مختلف کل پروژه است، بنابراین ممکن است هر سیستم یا app نام جداول یکسانی داشته باشد از این طریق یکتایی جداول در دیتابیس که مخزنی برای نگه داری تمامی جداول در تمامی app ها است حفظ میشود

توجه: از اینجا به بعد در مورد ساخت جداول تعریف مختصر تری داده میشود، زیرا به نظرم در بخش طراحی جدول استان ها توضیحات کاملی بیان شد ولی اگر طول توضیح با ویژگی جدیدی آشنا شویم به طور مفصل توضیح خواهم داد.

ساخت جدول موضوع ها

برای کامل کردن هر چه بهتر اطلاعات ما از مکان های ثبت شده ، نیازمند این بودیم که بدانیم هر محلی که در سیستم ذخیره میشود جزو کدام دسته از مکان های گردش گری است. این یعنی کاربری که از سیستم استفاده میکند با دستی باز واطلاعات کامل بتواند محل سفر خود را انتخاب کند.

برای این موضوع جدولی طراحی شده که این جدول نیز مانند جدول استان ها همان کار یکسان را میکند. یعنی اطلاعاتی که در داخل این جدول ذخیره میشود به تنهایی قابل استفاده نیست، پس نیاز مند صفات زیادی نیست.

history_category			CREATE TABLE "history_category" ("id" integer NOT NULL PRI
 id	integer	"id" integer NOT NULL PRIMARY KEY AUTOINCREMENT	
 name	varchar(40)	"name" varchar(40) NOT NULL	
 en_name	varchar(40)	"en_name" varchar(40) NOT NULL	

خوب همان جور که میبینید این جدول نیز شباهات فراوانی با جدول استان ها دارد. اسم این جدول به اسم category نام گذاری شده و دارای ۳ صفت است.

Id : این صفت برای یکایی هر موضوع طراحی شده به طوری که به صورت یکتا در سیستم ذخیره شود.

این صفت کلید اصلی P.K این جدول به شمار می آید. و درموجودیت های بعدی از آن به عنوان کلید خارجی یعنی F.K استفاده میشود.

این صفت بصورت خودکار برای همه جداول به صورت پیش فرض توسط جنگو گذاشته شده بنابر این نیازی به تغییر یا تعریف موردی در سیستم یا کد نیست.

Name : این نیز برای قرار گرفتن نام موضوع طراحی شده

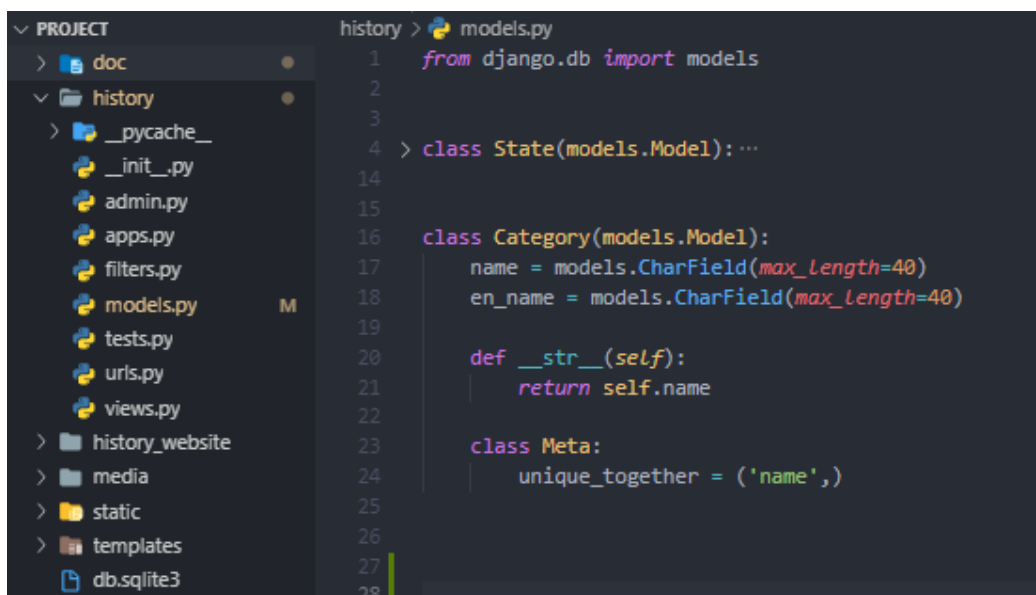
این صفت باید نام پارسی استان هارا در خود ذخیره کند پس رد موقع تعریف باید ویژگی به این صفت داده شود که قابلیت ذخیره سازی به کلمات و حروف پارسی را دارا باشد. همچنین این صفت نیاز مند ذخیره سازی به صورت رشته های بلند نیست، زیرا فقط نام یک موضوع در آن ذخیره میشود مانند: تاریخی، طبیعی و...

این صفت رشته های طولانی از کارکتر هارا نیاز ندارد ذخیره کند پس نیاز ندارد حافظه ای بزرگی را اشغال کند.

در این سیستم ما موضوع هارا به چهار گروه اصلی مذهبی، طبیعی، تاریخی، و تفریحی تقسیم کردیم ولی این به معنای آن نیست که این سیستم فقط به درد توریست میخورد ما با اضافه کردن گروه هایی مثل مکان های سیاسی، دولتی، روستوران ها یا... میتوانیم به قدرت مند کردن هرچه بیشتر این سیستم کمک کنیم.

En-name : این صفت نیز مانند صفت en-name در جدول استان ها به همین منظور طراحی شده که در بخش UX قوی تر نمایان شود.

این صفت در ویژگی طول نیز از صفت name تابعیت میکند و فرقی با آن ندارد.



```
1 from django.db import models
2
3
4 > class State(models.Model): ...
14
15
16 class Category(models.Model):
17     name = models.CharField(max_length=40)
18     en_name = models.CharField(max_length=40)
19
20     def __str__(self):
21         return self.name
22
23     class Meta:
24         unique_together = ('name',)
25
26
27
28
```

خوب به models.py میریم و صفات گفته شده را مینویسیم.

اسم این جدول category نام گذاری شده است دارای دو صفت name و en-name است. از کتابخانه models متد CharField را فراخوانی میکنیم و مقدار ۴۰ کارکتر مجاز را برایش تعریف میکنیم.

مانند کلاس state از مجیک فانکشن __str__ برای بازگرداندن رشته استفاده میکنیم

و از کلاس از پیش تعیین شده ی Meta به عنوان یکتا کردن صفت name استفاده میکنیم.

ساخت جدول عکس های محل

همان طور که میدانید هر مکان در صفحه مختص به آن شامل چندین عکس از آن مکان است که این یعنی ماهیت ارتباطی آن به هر مکان یک ارتباط یک به چند یا 1:n است.

برای این منظور مجبور با ساخت یک جدول جدا برای نگه داری تک تک این عکسا شدم.

این جدول مانند دو جدول یکسان است یعنی اطلاعاتی که در آن قرار میگیرد یک آدرس عکس است و یک نام که برای راحت تر خواندن عکس در بخش وارد کردن محل جدید استفاده میشود.

اسم این جدول را به نام **PleceImage** نام گذاری کردم که بعدا با جدول **Place_Image** دچار مشکل نشود. زیرا جدول **Place_name** یک جدول ارتباطی بین جدول **place** و **PleceImage** است و ماهیت **1:n** را برقرار میکند

همان طور که میدانید هر مکان در صفحه مختص به آن شامل چندین عکس از آن مکان است که این یعنی ماهیت ارتباطی آن به هر مکان یک ارتباط یک به چند یا **1:n** است.





برای این منظور مجبور با ساخت یک جدول جدا برای نگه داری تک تک این عکسا شدم، بنابراین جدولی ساختم که این جدول شامل صفتی به عنوان کلید خارجی یا **F.K** است که در داخل این صفت کلید اصلی موجودیت محل در آن قرار دارد. از این طریق توانستم ارتباط یک به چند را برقرار بکنم.

فعلا جدول **PleceImage** رو توضیح بدم و بعد توضیح کامل در مورد جدول **Plece** که مهمترین جدول ماست ، جدول **Place_Image** را توضیح خواهم داد که یک جدول رابط بین دو جدول است.

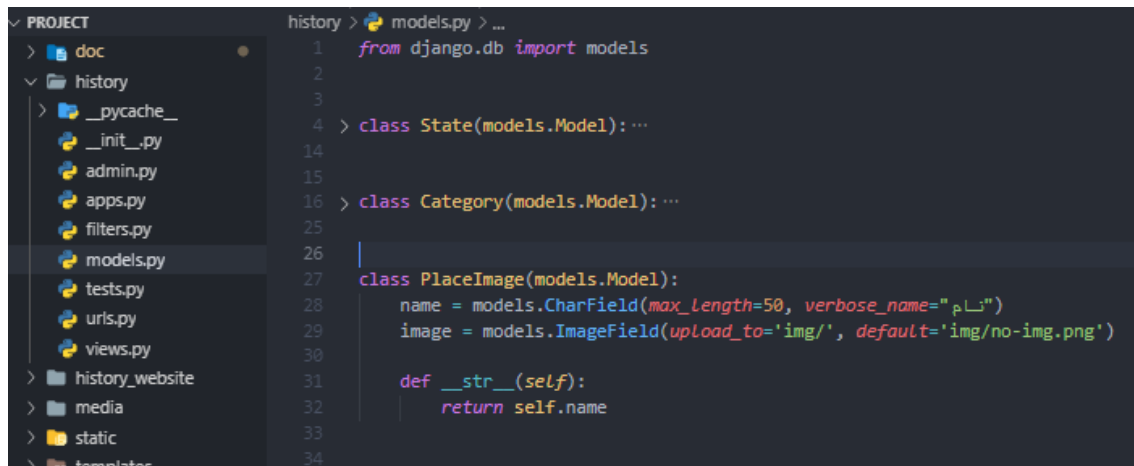
همان طور که گفته شد این جدول پیچیدگی خاصی ندارد و تنها نکته آن این است که شما با استفاده از پنل ادمین هر چه تعداد عکس بخواید میتوان اضافه کنید و با قرار دادن یک نام به عکس آنرا در موقعی که میخواهید محل جدید اضافه کنید شناسایی کنید و چندین عکس را برگزینید.

این جدول شامل صفت **name** است که همانطور گفته شد برای شناسایی آن استفاده میشود. این صفت مانند تمام صفات نام که در جداول قبلی بود نیازمند ذخیره طولانی رشته ها نیست

صفت بعدی این جدول به نام **Image** نام دارد که آدرس ذخیره شده ی عکس در پروژه را داراست، این صفت باید قابلیت ذخیره سازی رشته های بلند را دارا باشد ، چون ممکن است نام عکس به اندازه قابل توجهی طولانی باشد.

 history_placeimage	CREATE TABLE "history_placeimage" ("id" integer NOT NULL PRI	
 id	integer	"id" integer NOT NULL PRIMARY KEY AUTOINCREMENT
 name	varchar(50)	"name" varchar(50) NOT NULL
 image	varchar(100)	"image" varchar(100) NOT NULL

در بالا تصویر ساخته شده جدول در دیتابیس توسط جنگو را مبینیم و همان طور که گفته شد دارای خصوصیت **id** است که ما آن را تعریف نکردیم.



```
1 from django.db import models
2
3
4 > class State(models.Model): ...
14
15
16 > class Category(models.Model): ...
25
26
27 class PlaceImage(models.Model):
28     name = models.CharField(max_length=50, verbose_name="نام")
29     image = models.ImageField(upload_to='img/', default='img/no-img.png')
30
31     def __str__(self):
32         return self.name
33
34
```

و اما بخش کد نویسی آن کلاسی ساخته میشود به نام **PlaceImage** که نام جدول ما است و صفت **name** که یکی از متغیرها ما است.

تنها چیزی که در اینجا در جداول قبلی فرق میکند مقدار **verbose_name** است که عملکرد یکسانی با **magic function** ها دارد بهطوری که هر جا **PlaceImage.name** صدا زده شود مقدار کارکتری **name** بازگردانده میشود که این مورد فقط در پنل ادمین وجود دارد ، برای خوانا تر شدن هرچه بهتر پنل مدیریت.

صفت **image** همان طور که گفته شده بود در آن آدرس مهم ذخیره ی عکس است. از کتابخانه **models** متد **ImageField** را فرا خوانی میکنیم که مخصوص ذخیره سازی عکس است، مقداری که میگیرد ۱ : **upload_to** است که آدرس دقیق دایرکتوری که میخوايد در آن ذخیره شود، دقت فرمایید مسیری که عکس ها در آن ذخیره میشود پوشه **static** در شاخه اصلی پروژه است و من با قرار داد پوشه **img** در این دایرکتوری محل ذخیره سازی را انتخاب کردم.

۲: **default** که در صورت انتخاب نکردن یک عکس توسط کاربر به صورت خودکار یک عکس از پیش تعیین شده جای آن را میگیرد.

و در انتها مجیک فانکشن **__str__** که قبلا توضیحات کاملی از آن بیان شد.

ساخت جدول محل ها

همان طور که قبلا گفته شد مهمترین جدول در این سیستم جدول محل ها است. اطلاعاتی که در این جدول ذخیره میشود مجموعه ای اطلاعاتی که در ۴ جدول دیگر ذخیره میشود است، که شامل استان ، نوع و عکس های آن است.

این جدول شامل صفات زیادی است که در زیر آنها را بررسی میکنیم

صفت **name** : باید نام محل مورد نظر در آن قرار بگیرد و قابلیت نگه داری رشته ها با طول متوسط را داشته.

صفت **description** : این صفت باید توضیحاتی در مورد مهم در خود نگه دارد این صفت باید دارا قابلیت نگه داری رشته ها به صورت نامحدود باشد.

صفت **url** : این صفت باید نگهدارند یک لینک باشد این لینک در پروژه همان صفحه گوگل مپ است که مختصات دقیق آن مکان را در سیستم نگه داری میکند.

صفت **state** : این صفت یک کلید خارجی یا به اصطلاح F.K است. در اینجا ما نگه دارندر کلید اصلی یا P.K جدول **state** ها هستیم که از این طریق میتوان از هر مکان پی به استان آن مکان برد. و در صورت انتخاب استان به تمام مکان هایی که در استان ثبت شده برسیم.

صفت **category** : این صفت مانده صفت **state** یک F.K است که نگه دارند P.K جدول **category** است. از این طریق میتوان محل ها را بر اساس نوع آنها طبقه بندی کرد، یعنی کاربر فقط محل های تاریخی را ببیند و میتوان در صورت داشتن یک محل به نوع آن محل که چه نوع است پی برد.

صفت **thumbnail** : صفتی است که در آن یک عکس قرار میگیرد، همان عکس که در صفحه لیست محل ها بر روی اتم آن میبینیم. اینکار برای این انجام شد که یک عکس شاخص بر روی تمام اتم ها قرار گرفته باشد.

صفت **image** : این صفت همان طور که گفته شده بود برای اتصال چندی عکس به یک محل درست شده، البته هنوز به بررسی جدول **Piece_image** که جدول رابط است نرسیدیم و در جای خود توضیح داده خواهد شد.

history_place		CREATE TABLE "history_place" ("id" integer NOT NULL PRI
id	integer	"id" integer NOT NULL PRIMARY KEY AUTOINCREMENT
name	varchar(50)	"name" varchar(50) NOT NULL
thumb	varchar(100)	"thumb" varchar(100) NOT NULL
desc	text	"desc" text
url	varchar(200)	"url" varchar(200) NOT NULL
category_id	integer	"category_id" integer NOT NULL
state_id	integer	"state_id" integer NOT NULL

در تصویر بالا جدو ساخته شده توسط جنگو در دیتا بیس را میبینیم.

و اما میرسیم به بخش کد نویسی در فایل models.py

```

1  from django.db import models
2
3
4  > class State(models.Model): ...
14
15
16 > class Category(models.Model): ...
25
26
27 > class PlaceImage(models.Model): ...
33
34
35 class Place(models.Model):
36     """Database Model for Places"""
37     name = models.CharField(max_length=50, verbose_name="نام")
38     thumb = models.ImageField(upload_to="img/")
39     image = models.ManyToManyField(PlaceImage)
40     desc = models.TextField(blank=True, null=True)
41     url = models.URLField()
42     category = models.ForeignKey("Category", on_delete=models.CASCADE, verbose_name="موضوع")
43     state = models.ForeignKey("State", on_delete=models.CASCADE, verbose_name="استان")
44
45     # magic function
46     def __str__(self):
47         return self.name
48
49     # part of models
50     class Meta:
51         unique_together = ('name',)
52

```

همان طور که در دتصور میبینید کلاس ساخته شده به نام place که نمایان گر اسم جدول است.

صفت thumbnail که در اینجا به اختصار thumb تعریف شده و از متد ImageField برای ذخیره

کردن عکس استفاده شد

صفت description که به اختصار desc در اینجا تعریف شده باید مقدار نامحدودی از رشته را در خود

ذخیره کند برای این کار از متد های کتابخانه models متد TextField انتخاب شد که این قابلیت را

داراست. همچنین برای این ویژگی null پذیر تعریف شده تا کاربر بتواند مقایر خالی در سیستم ذخیره کند.

صفت url که همان طور گفته شد باید یک لینک در خود ذخیره کند و با استفاده از متد URLField اینکار

را امکان پذیر کردم.

خوب می‌رسیم به صفت **state** این صفت همان طور که گفته شد باید یک **P.K** از یک جدول دیگر در خود ذخیره کند بنابراین در بین متد ها، از متد **ForeignKey** استفاده کردم، این متد در **parameters** های ورودی خود نام یک جدول را می‌گیرد که در اینجا جدول **state** داده شده است.

دیگر **parameters** های این متد ورودی **on_delete** است که به این معناست اگر خواسته شد محلی پاک شود سیستم **state** مربوط به آن محل را پاک نکند.

صفت **category** هم دقیقاً شبیه به صفت **state** است و هیچ گونه تفاوتی با آن ندارد، تنها فرقی که با **state** داراست این است که در بخش انتخاب یک جدول برای **F.K** باید آن را به جدول **category** متصل کرد تا ارتباط به درستی انجام شود.




دقت فرمایید در دیتا بیس فیلدی به نام **Image** وجود ندارد دلیل این را در بخش بعد توضیح خواهد داد.

ساخت جدول رابط

این جدول رابط بین جدول **Placeimage** و **Place** قرار دارد تا بتواند از این طریق ارتباط را برقرار کند، دقت داشته باشید ساخت این جدول توسط ما انجام نمیشد و خود جنگو این جدول را می‌سازد پس دنبال کد آن در فایل **Models.py** نگردید.

همان طور که در بخش قبل گفته شده بود فیلد **Image** جدول **Place** وجود ندارد، میدانید که در اینجا از ارتباط **n:m** استفاده شده و جنگو با استفاده از همان صفت و متد **ManyToManyField** جدول رابط را به نام **Place_Image** ساخته.

طرز کار این جدول به این شکل است که کلید اصلی جدول **Place** را فیلد **Place_id** خود قرار میدهد و در فیلد بعدی که **PlaceImage_id** است کلید اصلی عکس های ذخیره شده در جدول **PlaceImage** را قرار میدهد و با استفاده از کلید اصلی خود جدول، تمام عکس هایی که به نام آن محل ثبت شده را بارگزاری میکند.

history_place_image		CREATE TABLE "history_place_image" ("id" integer NOT NULL PRI
 id	integer	"id" integer NOT NULL PRIMARY KEY AUTOINCREMENT
 place_id	integer	"place_id" integer NOT NULL
 placeimage_id	integer	"placeimage_id" integer NOT NULL

دستورات و کدهای اجرایی

به مهم ترین بخش این مستند میرسیم، قسمت دستورات و کدهای که باعث اجرا برنامه میشه، توجه داشته باشید که در این قسمت بنده همه ی کدهای رو نردم یعنی پس از شروع پروژه با استفاده از جنگو، خود جنگو اکثر تنظیمات هارا از قبل انجام میده و برنامه نویس فقط کافیس رو پروژه ی خود تمرکز کند.

در اینجا قصد نداریم تمام تنظیمات یا کدهای زده شده را توضیح دهیم والا برای این کار اصلا مدیا نوشته مناسب نیست و حتما باید در فیلم و به صورت عملی آن را نشان داد دوما برای این کار فک کنم ۱۰-۱۵ ساعت باید ویدیو آموزشی ساخت تا تمام کارهایی که انجام شده را توضیح و دلیل آن را بگوییم. اگر علاقه مند به نحوه کار جنگو هستید در بخش [منابع و مآخذ](#) میتوانید آنها را دنبال کنید.

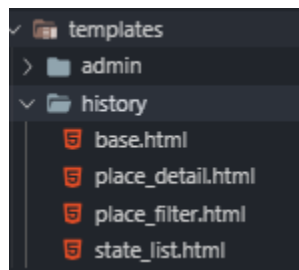
تمپلیت ها

یکی از مهم ترین بخش های سایت **templates** است، زیرا تمام آن چیزی که کاربر میبینید یعنی کدهای HTML و CSS در اینجا قرار دارد.

خب همان طور که میدانید پروژه دارای چند صفحه است که و هر صفحه با صفحات دیگر در بخش هایی مثل **header** ، **head** ، **footer** و... مشترک است و بخش **content** یا محتوا هست که با یکدیگر متمایز هستند، حال فرض کنید ی تکه از قطعه کد در بخش مشترک را تغییر یا حذف کنیم؟

خب اگر با **template** ها آشنا نباشید طبیعتا در هر صفحه شروع به پیدا کردن آن کد میکنید و آن را تغییر میدهید. شاید برای ۱ یا ۲ صفحه کاری چندان سخت نباشد ولی اگر تعداد صفحات شما به ۱۰-۱۵ یا حتی بیشتر برسد میخوايد چه کاری بکنید؟ اینجااست که تمپلیت ها به داد ما میرسند، شما میتوانید به نگه داشت یک قطعات مشترک کد در یک صفحه آنرا به یک صفحه مرجع تبدیل کنید و تنها با گذاشتن پرچم در این صفحه **content** هایی که هر کدام با یک دیگر متمایز هستند دار راهنمایی کنید که در کجا بنشینند

البته این همه مزیت templates نیست، یکی دیگر از مزیت های templates های کمتر شدن کد ها و در نتیج سریع تر شدن سرعت بارگزاری پروژه است. در ضمن از این طریق کد ها هم دارای تمیزی قابل توجهی میشود و شما دچار مشکلات ساده و پیش پا افتاده برنامه نویسی نخواید شد.

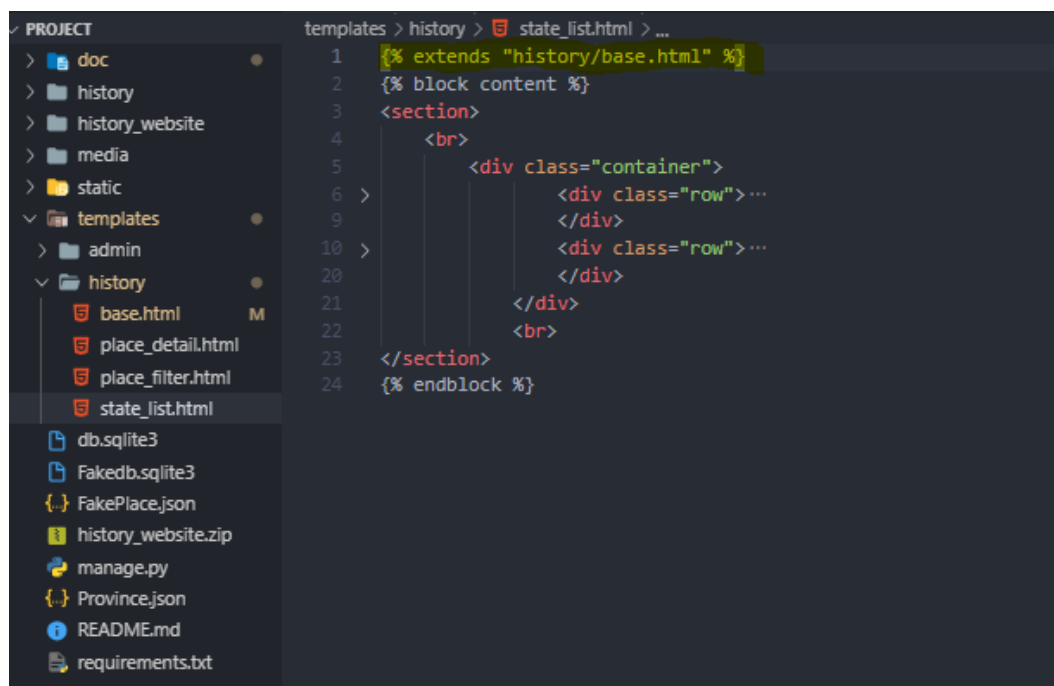


همان طور که در تصویر میبینید در این پروژه چهار صفحه طراحی شده که یکی از آنها به نام base.html صفحه مرجع است و سه صفحه دیگر یعنی place_detail.html ، place_filter.html و state_list.html که صفحات content هستند در صفحه مرجع جایگذاری میشوند.

```
PROJECT templates > history > base.html > html > body > script
1 {% load static %}
2 <!DOCTYPE html>
3 <html lang="en">
4
5 <head>...
27 </head>
28
29 <body>
30 <!-- Page Preloader -->
31 <div id="preloader">...
32 </div>
33
34 <!-- Header section -->
35 <header class="header-section">...
36 </header>
37
38 <!-- Header section end -->
39
40 <!-- Page info -->
41 <div class="page-info-section set-bg" data-setbg="{% static 'index/img/page-bg/2.png' %}">...
42 </div>
43 <!-- Page info end -->
44
45 <!-- search section -->
46 <section class="search-section ss-other-page">...
47 </section>
48 <!-- search section end -->
49
50 <!-- block content -->
51 {% block content %}
52 {% endblock %}
53 <!-- footer section -->
54 <footer class="footer-section pb-0">...
55 </footer>
56 <!-- footer section end -->
57
58 <script src="https://cdnjs.cloudflare.com/ajax/libs/uikit/3.1.6/js/uikit.min.js"
59 integrity="sha256-v789mr/zBbgR53mfydCI78CSAF+9+nRqu+JRFs1UPg0=" crossorigin="anonymous"></script>
60 <script src="https://cdnjs.cloudflare.com/ajax/libs/uikit/3.1.6/js/uikit-icons.min.js"
61 integrity="sha256-1+Am2G1Fz41J+gms80qC7fas1JDbberZDhJEsmdmQy8s=" crossorigin="anonymous"></script>
```

خب همان طور که میبینید در این فایل base.html تمام کد های HTML و تمام کتابخانه های استفاده شده قرار دارد این همان صفحه مرجعی است که گفته شده بود.

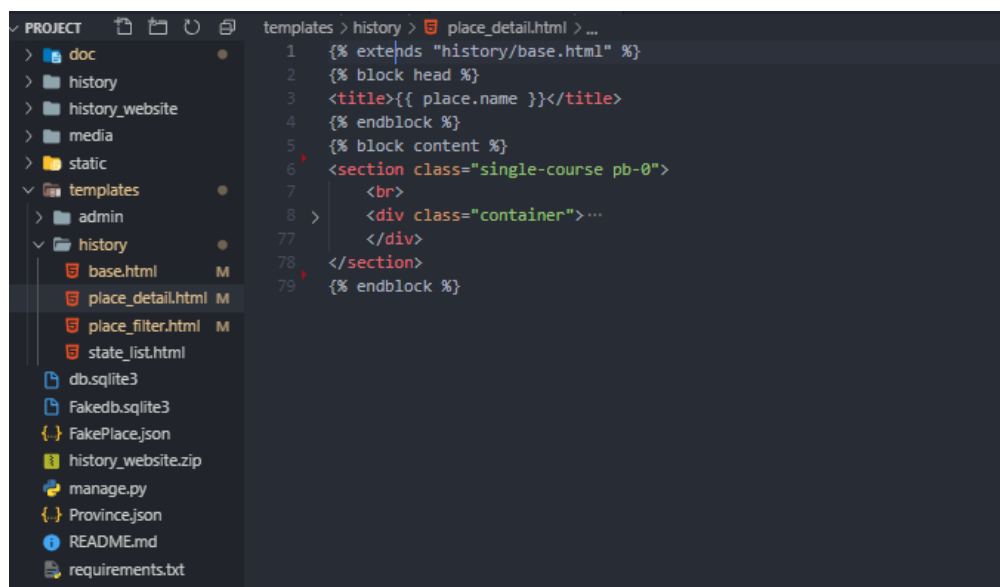
اگر به وسط صفحه دقت کنید بخشی را زرد رنگ کردم این همان پرچمی است که توضیح داده ام



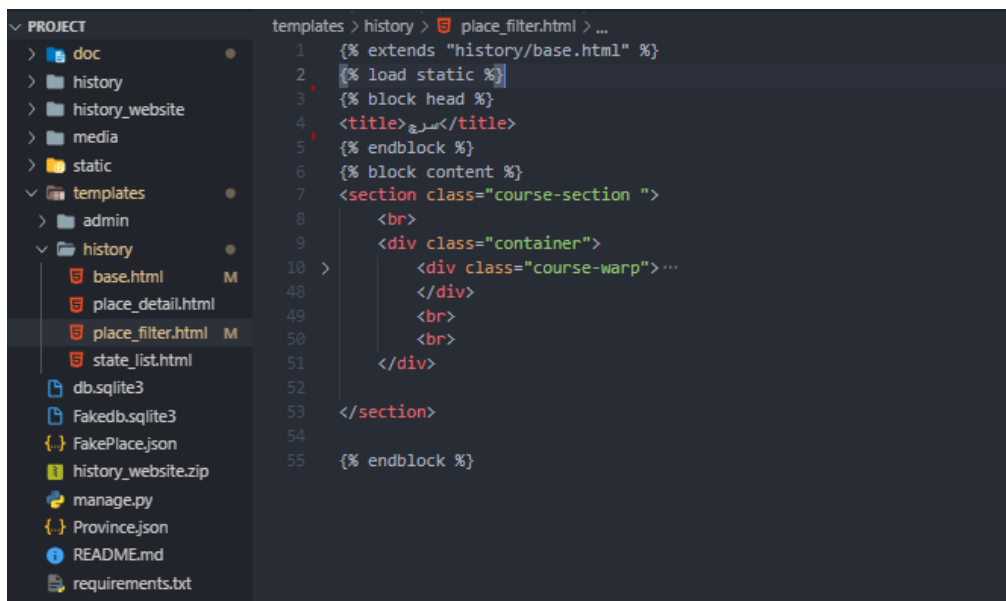
```
1 {% extends "history/base.html" %}
2 {% block content %}
3 <section>
4     <br>
5     <div class="container">
6 >         <div class="row">...
9         </div>
10 >         <div class="row">...
20         </div>
21     </div>
22     <br>
23 </section>
24 {% endblock %}
```

خب یکی از صفحات **content** است که میبینید، همان طور که با رنگ زرد مشخص کردم این صفحه به صفحه مرجع **base.html** وصل میشود و به طوری کلی نمایش میدهد

اگر دقت کنید این صفحه هیچ یک از کدهای اساسی برای بارگزاری یک صفحه ی استاندارد را ندارد یعنی بدون تگ های مهم **<html>** ، **<body>** ، **<head>** و...



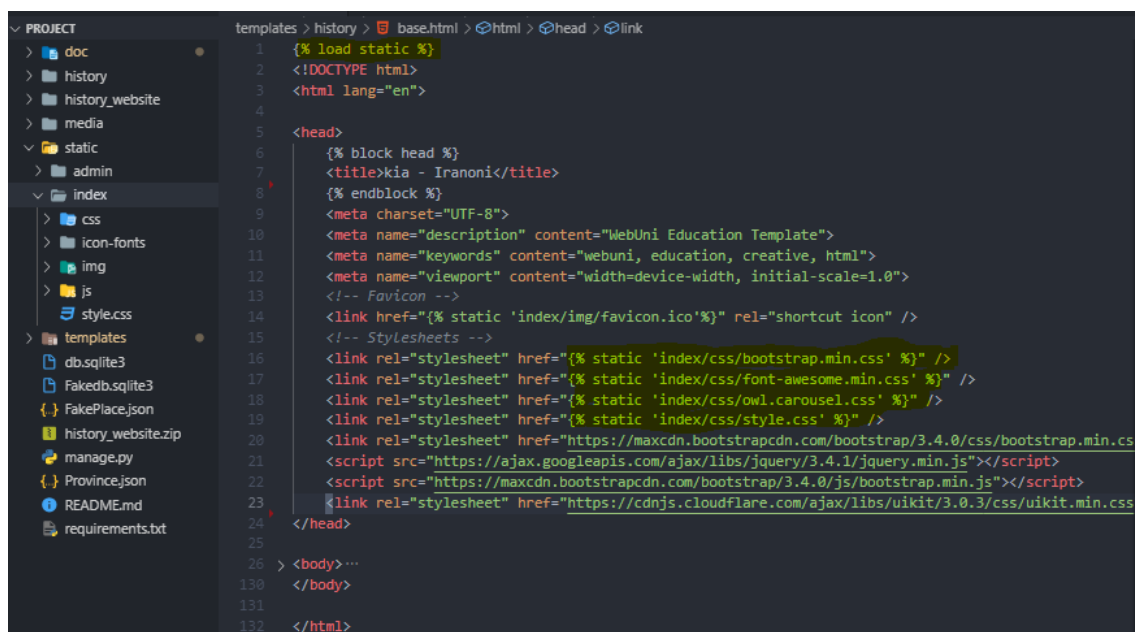
```
1 {% extends "history/base.html" %}
2 {% block head %}
3 <title>{{ place.name }}</title>
4 {% endblock %}
5 {% block content %}
6 <section class="single-course pb-0">
7     <br>
8 >     <div class="container">...
77     </div>
78 </section>
79 {% endblock %}
```



```
1 {% extends "history/base.html" %}
2 {% load static %}
3 {% block head %}
4 <title>سریع</title>
5 {% endblock %}
6 {% block content %}
7 <section class="course-section">
8     <br>
9     <div class="container">
10         <div class="course-warp">...
48     </div>
49     <br>
50     <br>
51     </div>
52
53 </section>
54
55 {% endblock %}
```

دو تصویر بالا دو فایل دیگر content^۳ هستند که در فایل مرجع جای میگیرند.

خواندن فایل ها به صورت ایستا



```
1 {% load static %}
2 <!DOCTYPE html>
3 <html lang="en">
4
5 <head>
6     {% block head %}
7     <title>kia - Iranoni</title>
8     {% endblock %}
9     <meta charset="UTF-8">
10    <meta name="description" content="WebUni Education Template">
11    <meta name="keywords" content="webuni, education, creative, html">
12    <meta name="viewport" content="width=device-width, initial-scale=1.0">
13    <!-- Favicon -->
14    <link href="{% static 'index/img/favicon.ico'" rel="shortcut icon" />
15    <!-- Stylesheets -->
16    <link rel="stylesheet" href="{% static 'index/css/bootstrap.min.css' %}" />
17    <link rel="stylesheet" href="{% static 'index/css/font-awesome.min.css' %}" />
18    <link rel="stylesheet" href="{% static 'index/css/owl.carousel.css' %}" />
19    <link rel="stylesheet" href="{% static 'index/css/style.css' %}" />
20    <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.0/css/bootstrap.min.css" />
21    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
22    <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.0/js/bootstrap.min.js"></script>
23    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/uikit/3.0.3/css/uikit.min.css" />
24 </head>
25
26 <body>...
130 </body>
131
132 </html>
```

یکی دیگر از مطالب مهمی که باید بیان میشد لود کردن فایل ها به صورت ایستا است به نحوی که با تغییر مسیر پروژه فایل های پروژه بهم نخورد ، این کار را از طریق خط هایی که زرد رنگ شده نشان داده ام. با این روش در صورت نوشتن کلمه static قبل آدرس سیستم پویتر خواندن فایل را به پوشه static در شاخه اصلی پروژه می شناسد.

^۳ مطالب

نوبت به مسیریابی در سایت میرسد. اهمیت این بخش بسیار بالاست زیرا این بخش اگر به خوبی عملگردد خود را نشان ندهد پروژه دچار مشکلات فراوانی میشود. از جمله این مشکلات عدم دسترسی به محل یا استان مورد نظر و... برخی مشکلات دیگه

اما اهمیت این بخش به این قسمت منتهی نمیشود و زیرا در این بخش به به خوبی سیستم MVC جنگو آشنا میشویم

همان قبلا گفته شده بود در اینجا قصد ندارم تمام کتابخانه هایی که استفاده کردم رو توضیح بدهم زیرا این قسمت در تمام پروژه ها جنگو وجود دارد، یعنی مهم نیست سطح و وسعت پروژه شما در چه حدی باشد، همین که در آن از مسیریابی و متد های مربوطه استفاده کرده باشید یعنی آشنایی کامل با نحوه اضافه کردن کتاب خانه ها دارید.

بریم سر اصل مطلب همان طور که در قسمت های قبل دیدین این پروژه شامل ۳ صفحه میشود که این یعنی کل پروژه در همین سه صفحه خلاصه میشود ، با همین منطق طبیعتا کاربر ۳ صفحه قابل جا به جایی در آن ها را دارد ، که این یعنی ۳ url باید ساخته شود و هر url نیز باید متصل به یک متد مشخص باشد تا بتواند از این طریق بتوانیم صفحات HTML را با دیتا های خودمان پر کنیم یا...

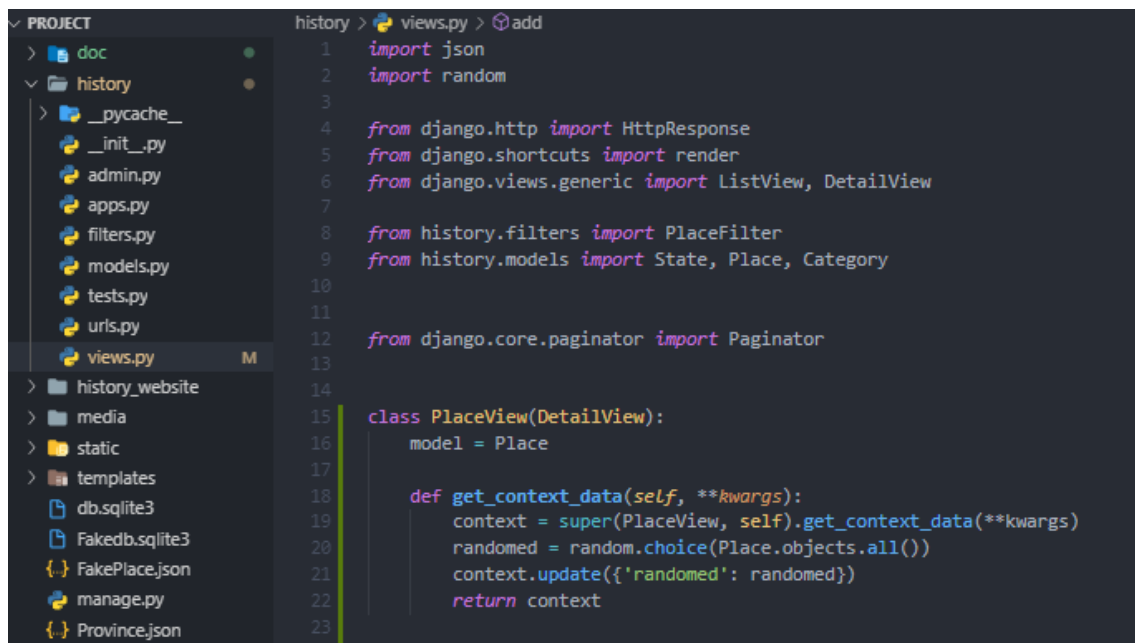
قبلا دیدم که صفحه ای به نام `place_detail.html` را داریم، ماهیت صفحه همان طور که از اسمش پیداست نمایش دهنده جزئیات یک محل است مانند گالری تصاویر، توضیحات و... ولی جنگو یا مروگر ما از کجا باید بفهمد که کدام مکان را برای ما به نمایش بگذارد؟

جواب این سوال در مبانی علم شبکه و طراحی صفحات وب است، به این صورت که ما میتوانیم با استفاده از url سایت به سیستم بگوییم چه چیزی میخوایم، برای توضیح بهتر ترجیح میدهم با یک مثال این موضوع را شرح دهم.

فرض کنید که میخواهیم جزئیات محل شماره ۱۶ را در صفحه ببینیم برای این کار با استفاده url به سیستم بگوییم خب به این صورت به سیستم میگوییم

/place/16

این عدد ۱۶ یا هر عددی که بعد از آن قرار میگیرد همان id در جدول place است، از این طریق میتوانیم مشخص کنیم چه چیزی را میخواهیم.



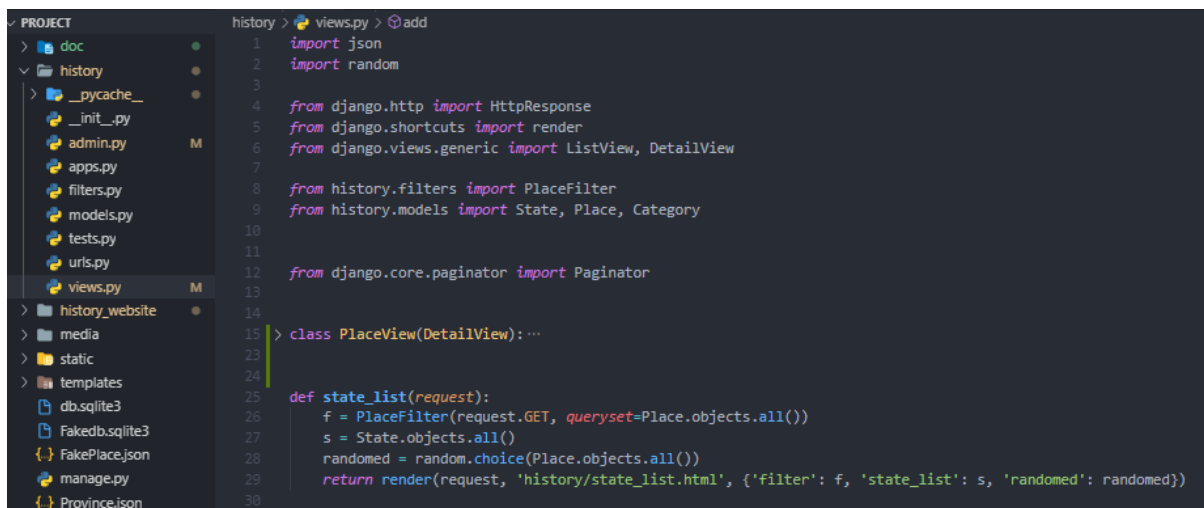
```
1 import json
2 import random
3
4 from django.http import HttpResponse
5 from django.shortcuts import render
6 from django.views.generic import ListView, DetailView
7
8 from history.filters import PlaceFilter
9 from history.models import State, Place, Category
10
11 from django.core.paginator import Paginator
12
13
14
15 class PlaceView(DetailView):
16     model = Place
17
18     def get_context_data(self, **kwargs):
19         context = super(PlaceView, self).get_context_data(**kwargs)
20         randomed = random.choice(Place.objects.all())
21         context.update({'randomed': randomed})
22         return context
23
```

همان طور که در تصویر میبینید کلاس PlaceView برای این طراحی شده که بتواند با توجه به آن url که توضیح داده شد محل مورد نظر را نمایش دهد.

البته فایل views.py به تنهایی کار نمیکند به این صورت که در تعریف دقیقی از ادرس url نشده که این در بخش بعد یعنی تشریح فایل urls.py میگذرد.

همان طور که میبینید در بخش import های این فایل میبینیم که models.py و تمام آن جداول که توضیح داده شد در این صفحه اضافه شده است، پس از این طریق میتون از object های آن کلاس استفاده کرد.

نکته دیگری که در این صفحه قابل توجه است این است که از کتابخانه رندوم استفاده شده، دلیل این برای بخش "نمیدونم کجا برم؟؟؟" در صفحه اصلی است با استفاده در اختیار داشتن تمام object ها models محل ها به صورت رندوم یک مورد را انتخاب میکند و در اخر تمام این اطلاعات return میشود.




```
PROJECT history > views.py > add
1 import json
2 import random
3
4 from django.http import HttpResponse
5 from django.shortcuts import render
6 from django.views.generic import ListView, DetailView
7
8 from history.filters import PlaceFilter
9 from history.models import State, Place, Category
10
11
12 from django.core.paginator import Paginator
13
14
15 > class PlaceView(DetailView): ...
16
17
18
19
20
21
22
23
24
25 def state_list(request):
26     f = PlaceFilter(request.GET, queryset=Place.objects.all())
27     s = State.objects.all()
28     randomized = random.choice(Place.objects.all())
29     return render(request, 'history/state_list.html', {'filter': f, 'state_list': s, 'randomed': randomized})
30
```

مورد بعدی که یکی از متد های **views** است، متد **state_list** است این متد در صفحا اصلی به کار میرود در همان قسمتی که لیست کامل تمام استان ها را به نمایش میگذارد.

توجه فرمایید که **state_list** هم مانند **Place** عمل میکند یعنی برای به نمایش گذاشتن تمام محل هایی که در استان مورد نظرمون است آن ها را از طریق **url** فیلتر میکند

/state/Tehran

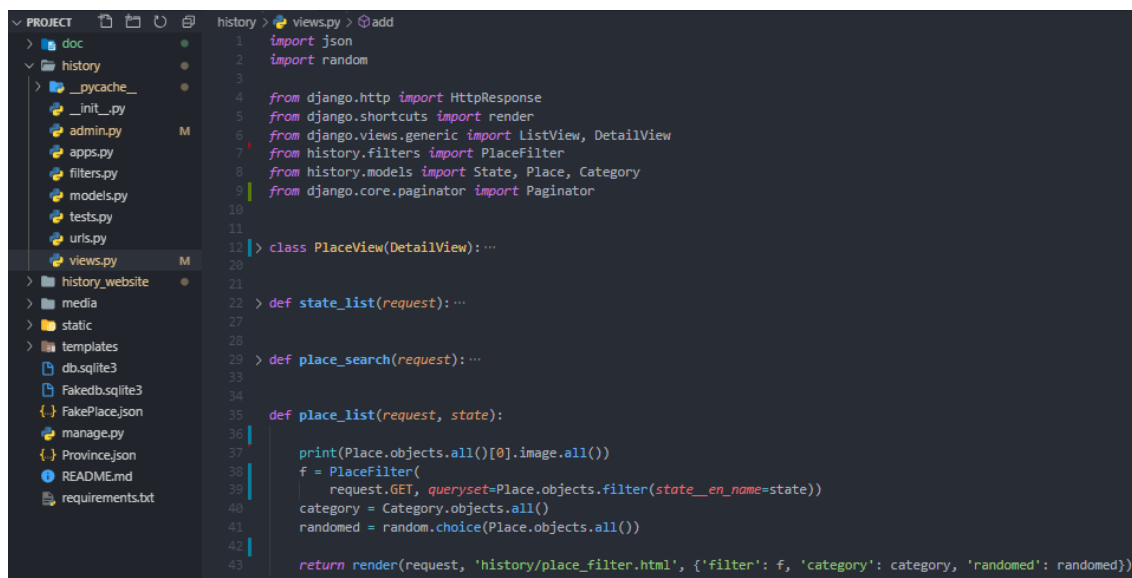
با این تفاوت که در **place** برای پیدا کردن از **id** آن مکان استفاده میشد ولی در **state** از **en_name** که در قسمت **models** ها تعریف کردیم



```
PROJECT history > views.py > add
1 import json
2 import random
3
4 from django.http import HttpResponse
5 from django.shortcuts import render
6 from django.views.generic import ListView, DetailView
7
8 from history.filters import PlaceFilter
9 from history.models import State, Place, Category
10
11
12 from django.core.paginator import Paginator
13
14
15 > class PlaceView(DetailView): ...
16
17
18
19
20
21
22 > def state_list(request): ...
23
24
25
26
27
28
29 def place_search(request):
30     f = PlaceFilter(request.GET, queryset=Place.objects.filter())
31     randomized = random.choice(Place.objects.all())
32     return render(request, 'history/place_filter.html', {'filter': f, "randomed": randomized})
33
```

متد بعدی که **place_search** است این متد برای این طراحی شده که بتوانیم در بخش سرچ سایت از آن استفاده کنیم و محل مورد نظر خودمون را بین تمام محل ها پیدا کنیم.

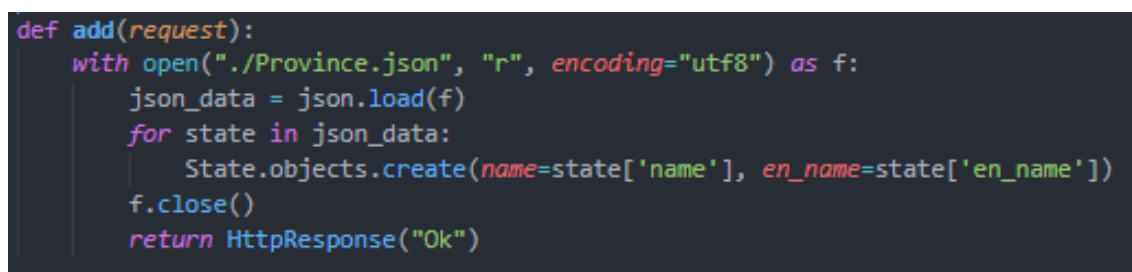
مقداری که این متد باز میگرداند مساوی با تمام محل ها است ولی بعدا این این object ها توسط form که داخل صفحه است محدود و یلتر میشود، این را در بخش کد های اجرایی به طور کامل توضیح خواهیم داد.



```
1 import json
2 import random
3
4 from django.http import JsonResponse
5 from django.shortcuts import render
6 from django.views.generic import ListView, DetailView
7 from history.filters import PlaceFilter
8 from history.models import State, Place, Category
9 from django.core.paginator import Paginator
10
11
12 class PlaceView(DetailView):...
13
14 def state_list(request):...
15
16 def place_search(request):...
17
18 def place_list(request, state):
19     print(Place.objects.all()[0].image.all())
20     f = PlaceFilter(
21         request.GET, queryset=Place.objects.filter(state__en_name=state))
22     category = Category.objects.all()
23     randomized = random.choice(Place.objects.all())
24
25     return render(request, 'history/place_filter.html', {'filter': f, 'category': category, 'randomed': randomized})
```

متد بعدی، متد place_list است این متد با استفاده از url سایت که هرچی باشد یعنی /state/Tabriz یا... محل ها را بر اساس آن state فیلتر میکند.

موقعی که شما در صفحه ای اصلی بر روی یکی از state ها کلیک میکنید اگر به url خود دقت کنید تغییر میکند و این زمان است که این متد شروع به کار میکند و تمام محل هایی که از آن استان انتخاب کرده اید را به شما نشان میدهد.



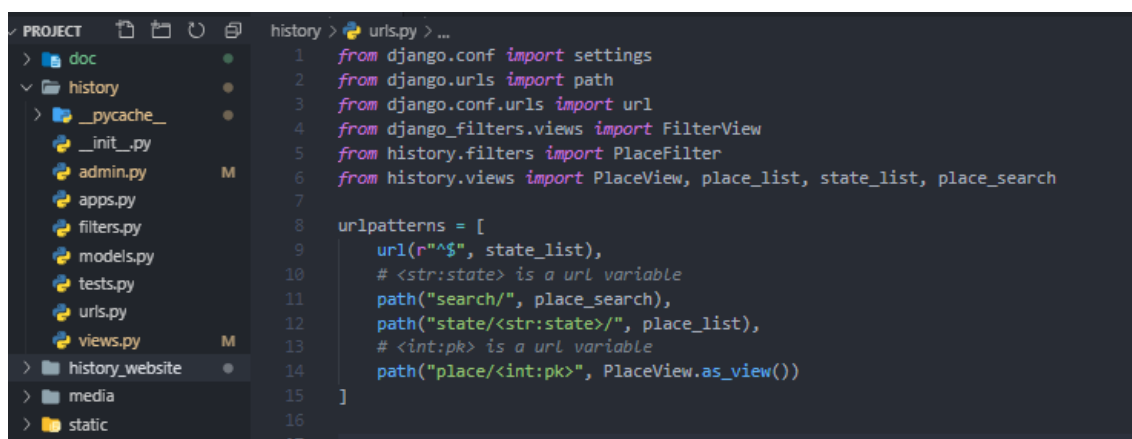
```
def add(request):
    with open("./Province.json", "r", encoding="utf8") as f:
        json_data = json.load(f)
        for state in json_data:
            State.objects.create(name=state['name'], en_name=state['en_name'])
        f.close()
    return JsonResponse("Ok")
```

و اما متد اخر که متد بسیار ساده ای است، البته این را باید متذکر بشوم که این متد در روند اجرایی پروژه هیچ گونه دخالتی ندارد و فقط برای راحتی حال شما در اضافه کرد تمام استان های ایران نوشته شده است. همان طور که از ظاهر این متد پیداست تمام object های فایل Province.json را میخواند و در models State مینویسد.

مسیریابی

همان طور که گفته شد مسیر یابی یکی از مهم ترین بخش های این پروژه است، حال به دور از توضیحات اضافی شروع به تفسیر کد ها میکنیم.

خب کد های این بخش در فایل `urls.py` قرار دارد، ولی همان جوری که فایل `views.py` کاربردی خاصی به جز تعریف کار ها کار خاص دیگری انجام نمیدهد و مکمل این آن `urls.py` است که میتواند متد ها را قابل دسترسی کند.



```
1 from django.conf import settings
2 from django.urls import path
3 from django.conf.urls import url
4 from django_filters.views import FilterView
5 from history.filters import PlaceFilter
6 from history.views import PlaceView, place_list, state_list, place_search
7
8 urlpatterns = [
9     url(r'^$', state_list),
10    # <str:state> is a url variable
11    path('search/', place_search),
12    path('state/<str:state>/', place_list),
13    # <int:pk> is a url variable
14    path('place/<int:pk>', PlaceView.as_view())
15 ]
16
17
```

همان طور که گفته شد میبینیم در قسمت `import` ها تمام متد های `views` اضافه شده

خب در `urlpatterns` تمام `path` های که ممکن است کاربر به آن ها مراجعه کند را نسبت به متد ها نوشته ام.

`url` اول همان صفحه اصلی سایت است که متدی در آن اجرا میشود، این متد `state_list` است، همان متدی که لیست کامل استان هارا باز میگرداند.

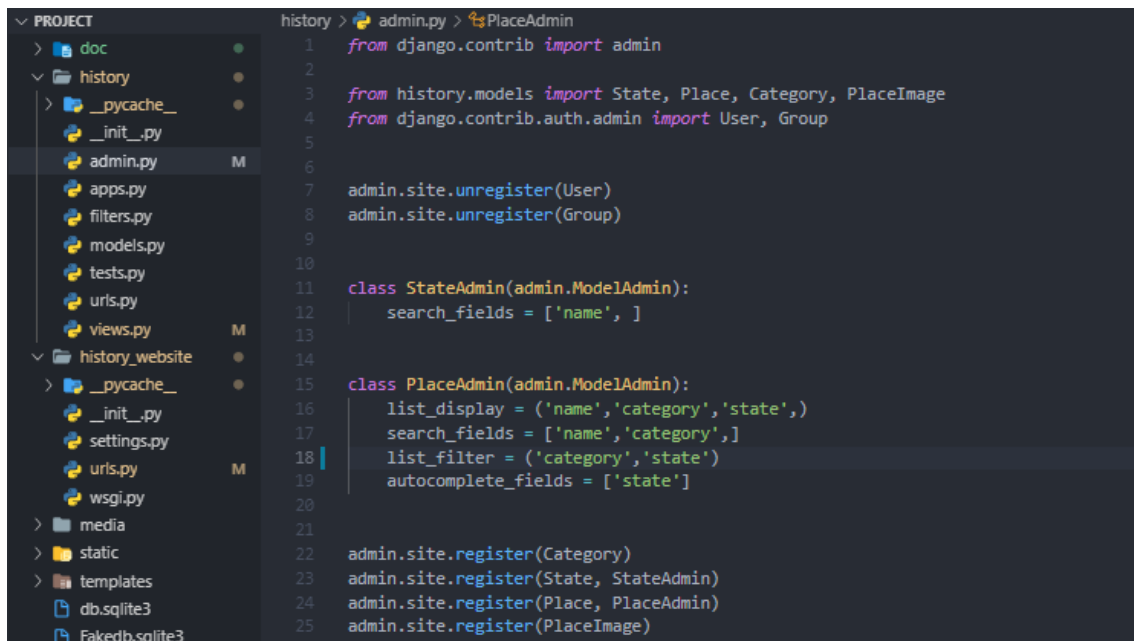
`url` دوم `/search` است که متد آن `place_search` است همان متدی که در سرچ یک `place` به کار میرود.

`url` بعدی `/state` است فرق این `path` با قبلی این است که این آدرس به تنهایی کار نمیکند و باید در ادامه ی آن یک رشته نوشت که هان `en_name` جدول `state` ها از و در اینجا استفاده میشود.

`url` اخر هم `palce` است دقیقا همانند `/state` کار میکند با این تفاوت که در مقابل آن باید `place id` را وارد کنید.

تنظیمات پنجره مدیریت

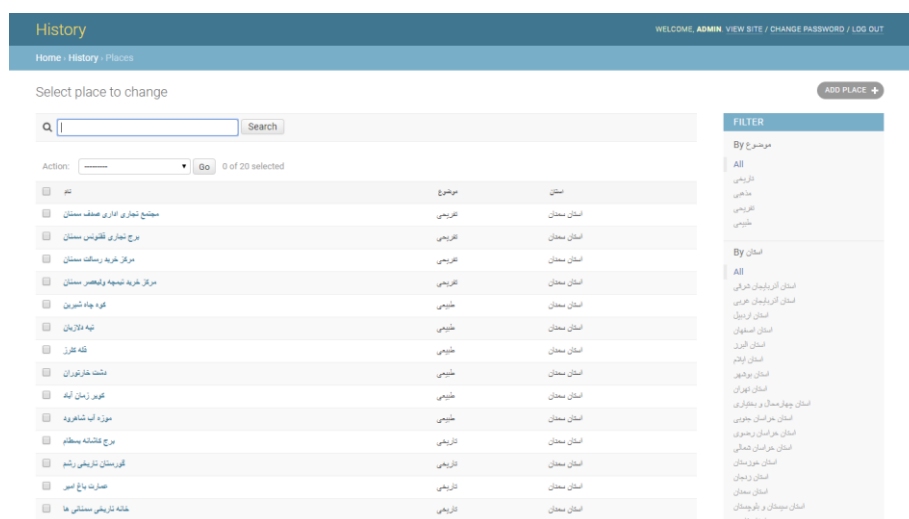
تنظیمات پنجره مدیریت یکی از بخش هایی است که فاصله کاربر را با محیط کد نویسی کمتر میکند و سطح دسترسی و مدیریت سایت را ارتقا میدهد، لازم به ذکر است که این بخش کد نویسی الگوریتمی ندارد و فقط pattern هایی که خود جنگو معرفی کرده را استفاده میکنیم.



```
PROJECT
├── doc
├── history
│   ├── __pycache__
│   ├── __init__.py
│   └── admin.py
├── apps.py
├── filters.py
├── models.py
├── tests.py
├── urls.py
├── views.py
├── history_website
│   ├── __pycache__
│   ├── settings.py
│   ├── urls.py
│   └── wsgi.py
├── media
├── static
├── templates
├── db.sqlite3
└── Fakedb.sqlite3

history > admin.py > PlaceAdmin
1 from django.contrib import admin
2
3 from history.models import State, Place, Category, PlaceImage
4 from django.contrib.auth.admin import User, Group
5
6 admin.site.unregister(User)
7 admin.site.unregister(Group)
8
9
10
11 class StateAdmin(admin.ModelAdmin):
12     search_fields = ['name', ]
13
14
15 class PlaceAdmin(admin.ModelAdmin):
16     list_display = ('name', 'category', 'state',)
17     search_fields = ['name', 'category',]
18     list_filter = ('category', 'state')
19     autocomplete_fields = ['state']
20
21
22 admin.site.register(Category)
23 admin.site.register(State, StateAdmin)
24 admin.site.register(Place, PlaceAdmin)
25 admin.site.register(PlaceImage)
```

قصد ندارم خط به خط توضیح بدهم که هر کد دقیقا چه کاری انجام میدهد، و فکر کنم کد ها به اندازه ای تمیز و قابل فهم است که بتوانید درک کنید. این اطلاعات شامل این است که مثلا در جدول نمایش اطلاعات محل ها چه اطلاعاتی از آن قابل رویت باشد یا در باکس سرچ فیلد، چه مواردی سرچ شود.



نمایش اطلاعات در صفحه

اطلاعات همانطور که میدانید از فایل `models.py` به `views.py` میرسد، در اینجا کاربر `path` در فایل `urls.py` تعریف شده متد های که تعریف شده صدا میزنند، ایم متد های به فایل `html` ای که به آن ها تعریف شده میروند و در آنجا رندر میشوند. بعد این فایل رندر شده که همراه با کد های `html` است به صفحه مرجع خود میروند که در اینجا `base.html` که است. این از طریق `templates` ها امکان پذیر شده است. این چرخه گردش اطلاعات از دیتا بیس تا کد های `html` است که کاربر در صفحه مرورگر خود میبیند، در اینجا قصد داریم روش رندر شدن اطلاعات در مرحله `views to html` را توضیح دهیم.

```
templates > history > state_list.html > section
1  {% extends "history/base.html" %}
2  {% block content %}
3  <section>
4  <br>
5  <div class="container">
6      <div class="row">
7          <div name="categorys" id="categorys">
8          </div>
9      </div>
10     <div class="row">
11         {% for state in state_list %}
12             <div class="col-3 state uk-margin-small-bottom">
13                 <a href="state/{{ state.en_name }}">
14                     <button class="btn btn-default" type="button">{{ state.name }}</button>
15                 </a>
16             </div>
17             {% empty %}
18                 <div class="col-2">
19                     هیچ استانی موجود نیست
20                 </div>
21             {% endfor %}
22         </div>
23     </div>
24 </section>
25 {% endblock %}
```

همان طور که میبینید در صفحه `state_list` در وسط کد ها یک حلقه `for` قرار تا پس از گرفتن لیست تمام استان ها در متغیر `state_list` آنها را تک به تک با استفاده در `template` که قبلا طراحی شده بارگزاری میکند. بقیه صفحات هم از همین طریق یعنی با استفاده از حلقه ی `for` تمام اطلاعات خود را بارگزاری میکنند و دارا نکته خاص یا جدید نیستند که شایان ذکر باشد.

بقیه تغییرات انجام شده به قدری کوچک و ناچیز هستند که ارزش بازگو کردن در اینجا را به آنها نمیبینم. فقط یک تصور میماند که گفتم شاید مهم باشد، تنظیمات مرب.ط به زبان و ساعت پنل مدیریت و آدرس دایرکتوری `static`، تصور تمام این موارد را در پایین میگذارم

```
# language settings
LANGUAGE_CODE = 'en'
TIME_ZONE = 'Asia/Tehran'
USE_I18N = True
USE_L10N = True
USE_TZ = True

# Static files (CSS, JavaScript, Images)
# https://docs.djangoproject.com/en/2.2/howto/static-files/
STATIC_URL = '/static/'
STATIC_ROOT = ''
STATICFILES_DIRS = (
    os.path.join(BASE_DIR, 'static/'),
)
MEDIA_URL = '/media/'
MEDIA_ROOT = os.path.join(BASE_DIR, 'media/')
GOOGLE_MAPS_API_KEY = 'AIzaSyB5y1to0agby0BZNC4LwqiaG9m0Bg'
```

حداقل سیستم مورد نیاز برای نصب و راه اندازی

- ✓ Processors: Intel Atom® processor or Intel® Core™ i3 processor
- ✓ Disk space: 1 GB
- ✓ Operating systems: Windows* 7 or later, macOS, and Linux

نرم افزار و کتاب خانه های ضروری

- ✓ Python* versions: 2.7.X, 3.6.X
- ✓ astroid* versions: 2.2.5
- ✓ autopep8* versions: 1.4.4
- ✓ Django* versions: 2.2.3
- ✓ django-filter* versions: 2.2.0
- ✓ django-widget-tweaks* versions: 1.4.5
- ✓ isort* versions: 4.3.21
- ✓ lazy-object-proxy* versions: 1.4.1
- ✓ mccabe* versions: 0.6.1
- ✓ Pillow* versions: 6.1.0
- ✓ pycodestyle* versions: 2.5.0
- ✓ pylint* versions: 2.3.1
- ✓ pytz* versions: 2019.1
- ✓ six* versions: 1.12.0
- ✓ sqlparse* versions: 0.3.0
- ✓ typed-ast* versions: 1.4.0
- ✓ wrapt* versions: 1.11.2

* بعدا به طور مفصل طریقه نصب این کتابخانه ها توضیح داده خواهید شد.

نرم افزار های غیر ضروری ولی مهم در اجرای هر چه بهتر پروژه

- ✓ Google Chrome* version 76.0.3809.100 (Official Build)
- ✓ FireFox* version 68.0.2
- ✓ PyCharm 2019.2 (Community Edition) Runtime version: 11.0.3+12-b304.10 amd64
- ✓ VScode* version: 1.37.0 (user setup)
- ✓ Atom* version 1.39.1

* تمامی برنامه های ذکر شده رایگان هستند و با یک سرچ ساده میتوان آن هارا نصب کنید

چگونگی نصب و راه اندازی

توجه: قبل از انجام مراحل نصب حتما به مواردی که در بخش [راهنمای کاربردی](#) ذکر شده بروید تا دچار مشکلاتی نشوید.

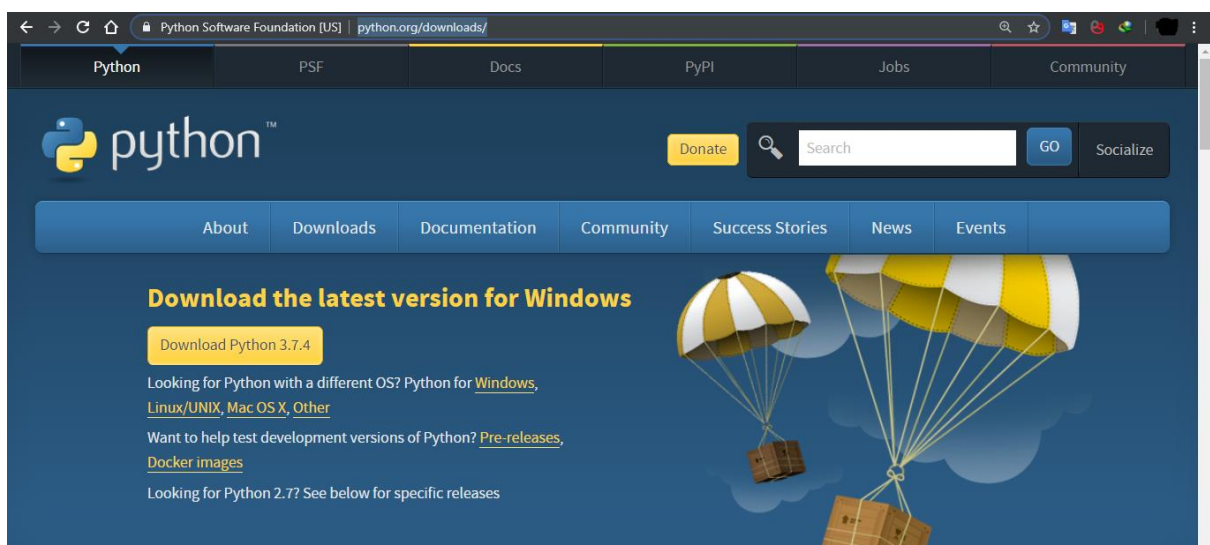
برای نصب و راه اندازی پروژه به سریع ترین شکل ممکن پیشنهاد میشود فایل [README.md](#) را مطالعه فرمایید.

ولی اگه در هنگام نصب به مشکلی برخوردید میتوانین از راهنمایی هایی که در داخل این بخش خواهد شد استفاده نمایید.

توجه: اگر از macOS یا هر توضیح linux استفاده میکنید نیاز به طی کردن مراحل ۱ و ۲ نیست، پایتون بصورت دیفالت در سیستم عامل شما قرار دارد.

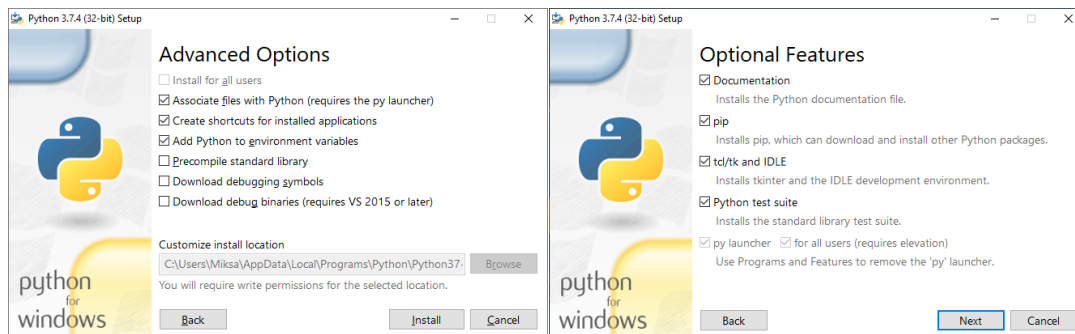
مرحله ۱: یکی از مهم ترین برنامه هایی که باید نصب بشه کامپایلر python است که وظیفه ای آن خواندن کد ها و اجرای آن ها است

برای نصب python میتوانید به مراجعه به سایت www.python.org در نوار Downloads و با کلیک بر روی دکمه Download Python 3.7.X شروع به دانلود و نصب آن بکنید.

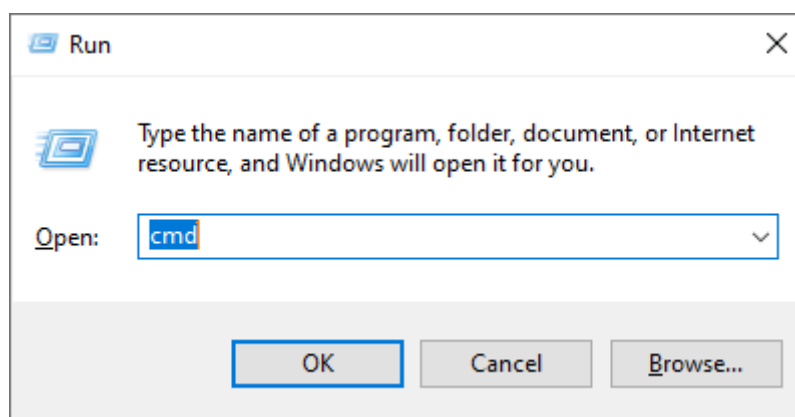


توجه: در زمان نگارش این پروژه از پایتون ورژن ۳.۷.۴ استفاده شده است. لذا اگر دچار مشکل شدید پیشنهاد میشود این نسخه را نصب یا virtual environment^۴ کنید

مرحله ۲: این مرحله بسیار آسان است و چند کلیک بر روی next مراحل نصب تمام میشود فقط باید به این نکته توجه کرد که در مراحل نصب، تیک add Python to environment variable^۵ رو بزنید تا در مراحل بعدی به مشکل بر نخورید.



مرحله ۳: با فشار داد کلید های windows+R پنجره run باز میشود. کافیست بنویسید cmd تا Command Prompt باز شود.



مرحله ۴: در command Prompt بنویسید

```
pip install -r requirements.txt
```

^۴ محیطی شبیه سازی شده که در آن میتواند نرم افزارهایی نصب کرد که در سیستم به آن نیاز ندارید یا ورژن پایین تر یا بالاتری نیاز دارید

^۵ اضافه کردن فایل اجرایی پایتون به command prompt

```
C:\Windows\system32\cmd.exe
C:\Users\Miksa\Desktop\Project>pip install -r requirements.txt
```

توجه داشته باید که این کد و کدهای دیگر را باید در شاخه **root** پروژه اجرا نمایید!

منتظر شوید تا **pip** تمام [کتابخانه های ضروری](#) شما را نصب کند

توجه : این مرحله نیازمند اتصال به اینترنت است.

مرحله ۵ : حال وقت آن رسیده که تنظیمات پایگاه را انجام دهید

در همان **Command Prompt** دو دستور زیر را به ترتیب وارد کنید تا پایگاه داده و تمامی جداول آن ساخته شود

```
Python manage.py makemigrations
```

```
Python manage.py migrate
```

```
C:\Windows\system32\cmd.exe
[01/Sep/2019 04:55:49] "GET /static/admin/fonts/Roboto-Bold-webFont.woff HTTP/1.1" 200 86184
[01/Sep/2019 04:55:56] "GET /admin/history/place/ HTTP/1.1" 200 12376
[01/Sep/2019 04:55:56] "GET /admin/jsi18n/ HTTP/1.1" 200 3223
[01/Sep/2019 04:55:56] "GET /static/admin/js/jquery.init.js HTTP/1.1" 200 363
[01/Sep/2019 04:55:56] "GET /static/admin/css/changelists.css HTTP/1.1" 200 6170
[01/Sep/2019 04:55:56] "GET /static/admin/js/actions.js HTTP/1.1" 200 6766
[01/Sep/2019 04:55:56] "GET /static/admin/js/admin/RelatedObjectLookups.js HTTP/1.1" 200 6918
[01/Sep/2019 04:55:56] "GET /static/admin/js/prepopulate.js HTTP/1.1" 200 1530
[01/Sep/2019 04:55:56] "GET /static/admin/js/core.js HTTP/1.1" 200 7099
[01/Sep/2019 04:55:56] "GET /static/admin/js/urlify.js HTTP/1.1" 200 8941
[01/Sep/2019 04:55:56] "GET /static/admin/img/search.svg HTTP/1.1" 200 458
[01/Sep/2019 04:55:56] "GET /static/admin/js/vendor/xregexp/xregexp.js HTTP/1.1" 200 128820
[01/Sep/2019 04:55:56] "GET /static/admin/js/vendor/jquery/jquery.js HTTP/1.1" 200 271817
[01/Sep/2019 04:55:56] "GET /static/admin/img/tooltag-add.svg HTTP/1.1" 200 331
[01/Sep/2019 04:56:32] "GET / HTTP/1.1" 200 14829
[01/Sep/2019 04:56:32] "GET /static/index/css/style.css HTTP/1.1" 304 0
[01/Sep/2019 04:57:46] "GET / HTTP/1.1" 200 14813
[01/Sep/2019 04:57:46] "GET /static/index/img/favicon.ico HTTP/1.1" 200 8186
[01/Sep/2019 05:01:43] "GET /admin/ HTTP/1.1" 200 7221
[01/Sep/2019 05:01:46] "GET /admin/history/state/ HTTP/1.1" 200 11883
[01/Sep/2019 05:01:46] "GET /admin/jsi18n/ HTTP/1.1" 200 3223
[01/Sep/2019 05:10:28] "GET /admin/ HTTP/1.1" 200 7221

C:\Users\Miksa\Desktop\Project>py manage.py migrate
Operations to perform:
  Apply all migrations: admin, auth, contenttypes, sessions
Running migrations:
  No migrations to apply.

C:\Users\Miksa\Desktop\Project>
```

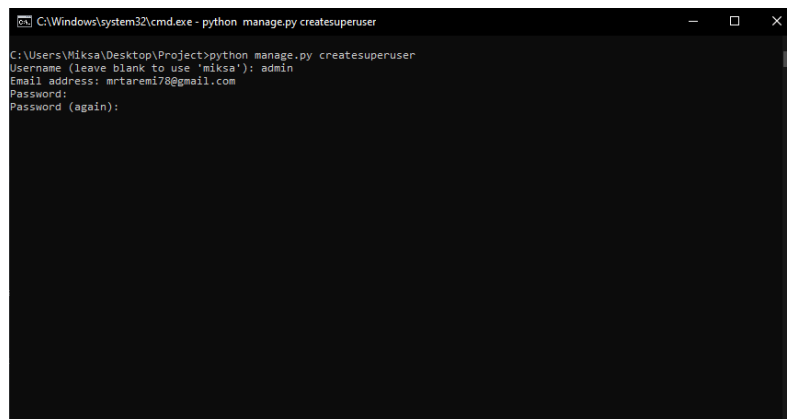
مرحله ۶ : وقت آن است که یک ادمین سایت معرفی کنید تا بتوانید سایت را مدیریت کنید

با استفاده از دستور زیر در Command Prompt کار را شروع کنید

```
python manage.py createsuperuser
```

در اینجا سیستم از نام کاربری ، ایمیل، و رمز معتبر درخواست میکند

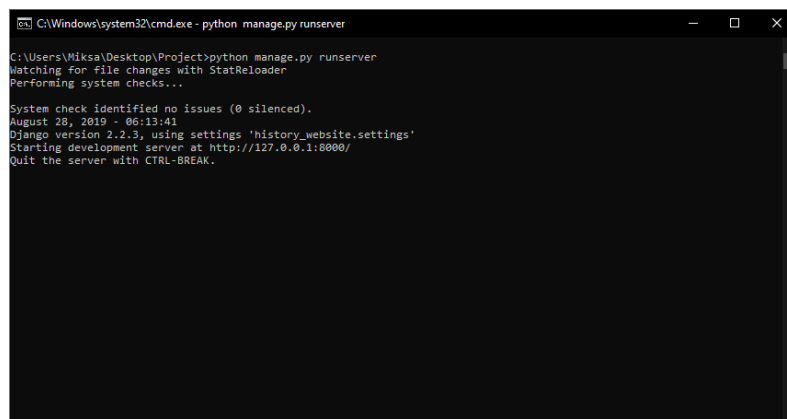
توجه : در صورت وارد نکردن یک رمز قوی ممکنه است به شما خطاری بدهد، که میتوان آن را به زدن دکمه Y رد کنید.



```
C:\Windows\system32\cmd.exe - python manage.py createsuperuser
C:\Users\Miksa\Desktop\Project>python manage.py createsuperuser
Username (leave blank to use 'miksa'): admin
Email address: mrtaremi78@gmail.com
Password:
Password (again):
```

مرحله ۷ : حال همه چیز آماده است. پروژه را Run کنید. با استفاده از دستور

```
python manage.py runserver
```



```
C:\Windows\system32\cmd.exe - python manage.py runserver
C:\Users\Miksa\Desktop\Project>python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
August 28, 2019 - 06:13:41
Django version 2.2.3, using settings 'history_website.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

توجه : نباید این پنجره Command Prompt را تا موقعی که میخوايد پروژه run باشد ببندید. در صورت

اتمام کار کافست در این پنجره کلید ترکیبی Ctrl+C را وارد کنید تا سرور قطع شود.

برای ورد به صفحه اصلی پروژه کافست در مرورگر خود آدرس <http://127.0.0.1:8000> را وارد

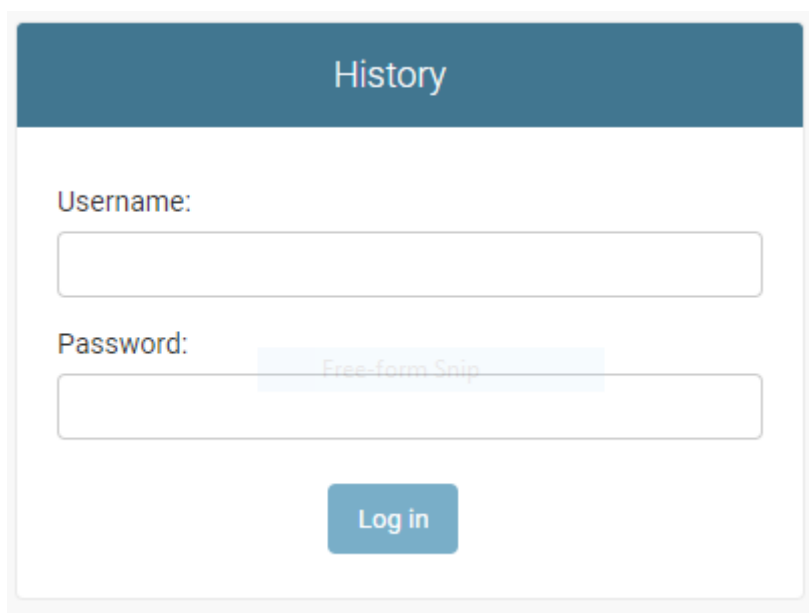
کنید تا به صفحه اصلی منتقل شوید

مرحله ۸: در انتها برای راحتی شما کافیسیت کانفیگ های اولیه را در مورد استان ها است را با استفاده از ورود به سایت <http://localhost:8000/add> انجام دهید.

توجه: این مرحله اجباری نیست و میتوانید خودتان آن را دستی انجام دهید.

شروع به کار و اضافه کردن محل مورد نظر

بعد از انجام مراحل نصب و اجرای پروژه وقت آن است که آن را مدیریت کنید. به این منظور به آدرس <http://127.0.0.1:8000/admin> رفته یا بر روی سر برگ مدیریت کلیک کنید. با این کار به صفحه ای زیر منتقل میشود.



History

Username:

Password:

Free-form Snip

Log in

با وارد کردن نام کاربری و رمز عبوری که انتخاب کردید به پنل مدیریت میرسید، البته اگر از دیتابیس بنده برای انجام کار های خودتون استفاده میکنید نام کاربری admin و رمز 2211 است.

History

WELCOME, ADMIN VIEW SITE / CHANGE PASSWORD / LOG OUT

Site administration

HISTORY

Categories	+ Add	Change
Place images	+ Add	Change
Places	+ Add	Change
States	+ Add	Change

Recent actions

My actions

[+](#) [مجموعه تجاری اداری صنعتی سمدان](#)
[Place](#)

[+](#) [مجموعه تجاری اداری صنعتی سمدان](#)
[Place image](#)

[+](#) [برج تجاری قناتون سمدان](#)
[Place](#)

[+](#) [برج تجاری قناتون سمدان](#)
[Place image](#)

[+](#) [برج تجاری قناتون سمدان](#)
[Place image](#)

[+](#) [برج تجاری قناتون سمدان](#)
[Place image](#)

[+](#) [برج تجاری قناتون سمدان](#)
[Place image](#)

[+](#) [مرکز خرید رسالت سمدان](#)
[Place](#)

[+](#) [مرکز خرید رسالت سمدان](#)
[Place image](#)

[+](#) [مرکز خرید رسالت سمدان](#)
[Place image](#)

بعد از ورود به پنل مدیریت با چنین صفحه ای بر میخورید. در گوشه بالای صفحه گزینه های خروج و تعویض رمز عبور را میبینید، در سمت چپ همان مقادیری که در صفحه `models.py` ها وارد کردیم یعنی جدول ها که شامل استان ها، موضوع ها، محل ها و عکس ها میشود.

برای اضافه کردن استان کافیت بر روی `state` کلیک کنید تا به صفحه مربوط به آن بروید.

History

WELCOME, ADMIN VIEW SITE / CHANGE PASSWORD / LOG OUT

Home · History · States

Select state to change

ADD STATE +

Q

Search

Action: Go 0 of 31 selected

STATE

☐

استان یزد

☐

استان همدان

☐

استان هرمزگان

☐

استان مرکزی☐☐☐☐☐☐☐☐☐☐

در اینجا شما میتوانید لیست کامل استان ها را مشاهده کنید اگر میخواهید مقادیری عوض کنید کافیت بر روی آن کلیک کنید و یا اگر میخواهید اضافه کنید بر روی `ADD STATE` کلیک کنید.

Change state

HISTORY

Name:

یزد

En name:

Yazd

Delete

Save and add another

Save and continue editing

SAVE

Add state

Name:

En name:

Save and add another

Save and continue editing

SAVE

جدول های موضوع و عکس ها هم دقیقا به همین شکل هستند یعنی قابلیت اضافه کردن و ویرایش کردن دارند.

History

WELCOME, ADMIN VIEW SITE / CHANGE PASSWORD / LOG OUT

Home » History » Places

ADD PLACE +

Select place to change

Q

Search

Action:

Go

0 of 20 selected

نام	موضوع	استان
<input type="checkbox"/> مجتمع تجاری اداری هدف سمنان	تفریحی	استان سمنان
<input type="checkbox"/> برج تجاری فلکونس سمنان	تفریحی	استان سمنان
<input type="checkbox"/> مرکز خرید رسالت سمنان	تفریحی	استان سمنان
<input type="checkbox"/> مرکز خرید تیمچه ولیعصر سمنان	تفریحی	استان سمنان
<input type="checkbox"/> کوچه چاه شیرین	طبیعی	استان سمنان
<input type="checkbox"/> تپه دلازیان	طبیعی	استان سمنان
<input type="checkbox"/> تکه نظرز	طبیعی	استان سمنان
<input type="checkbox"/> دشت خاخروران	طبیعی	استان سمنان
<input type="checkbox"/> کویر زمان آباد	طبیعی	استان سمنان
<input type="checkbox"/> موزه آب شاهرود	طبیعی	استان سمنان
<input type="checkbox"/> برج گلشاهه بسطام	تاریخی	استان سمنان
<input type="checkbox"/> گورستان تاریخی رشم	تاریخی	استان سمنان
<input type="checkbox"/> عمارت باغ امیر	تاریخی	استان سمنان
<input type="checkbox"/> خانه تاریخی سمبلی ها	تاریخی	استان سمنان

FILTER

By موضوع

All

تاریخی

مذهبی

تفریحی

طبیعی

By استان

All

استان آذربایجان شرقی

استان آذربایجان غربی

استان اردبیل

استان اصفهان

استان البرز

استان ایلام

استان بوشهر

استان تهران

استان چهارمحال و بختیاری

استان خراسان جنوبی

استان خراسان رضوی

استان خراسان شمالی

استان گلستان

استان زنجان

استان سمنان

استان مازندران

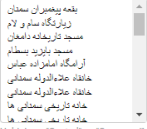
استان قزوین

تصویر بالا جدول محل ها را مشاهده میکنید، تفاوت این صفحه با صفحه های دیگه در این است که شما میتوانید محل خود را از طریق منو سمت راست بر اساس استان و موضوع فیلتر کنید تا راحت تر به محل مورد نظر خودتان برسید.

Add place

نام:

Thumb: No file chosen

Image:  +

Desc:

Url:

موضوع: +

استان: +

در بالا نمونه صفحه ADD PLACE رو میبینید که میتونید محل جدید به استان و موضوع مورد نظرتان اضافه کنید.

راهنمای کاربردی

در این بخش راهنمایی هایی که در تجربه هر چه بهتر از پروژه میتوانید داشته باشید را بازگو میکنم.

(۱) حتما از در موقع نصب و اجرا از وصل بودن به اینترنت مطمئن باشید. زیرا در موقع نصب چندین پکیج (که لیست آنها را گفته بودم) شروع به دانلود شدن میکنند.

و در زمان اجرا از لینک هایی (CDN^۷) استفاده شده که از طریق اینترنت خوانده میشود.

(۲) ممکن است برای دانلود بعضی اپدیت ها و پکیج ها نیاز به فیلتر شکن باشد.

حد امکان فیلتر شکن خود را روشن کنید.

⁷ سیستم ارسال اطلاعات از طریق نزدیک ترین سرور به کاربر

۳) تمام مراحل نصب قبلا طی شده است و تمام دیتا های تا حدی که در توانم بود وارد دیتابیس شده اند

بنابراین نیاز به طی کردن مراحل که در [راهنمای نصب](#) گفته شده نیست.

فقط برای اجرا کافیت [مرحله ۷](#) را اجرا کنید و نیازی به طی کردن مراحل دیگر نیست.

۴) اگر در هر صورت خواستید دیتابیس را پاک کرده و یک بار دیگر تمام دیتا هارا وارد کنید. کافیت فایل **db.sqlite3** را که در شاخه اصلی پروژه است را پاک کنید. و برای راه اندازی مجدد مراحل که در [راهنمای نصب](#) گفته شده راه طی کنید.

۵) برای سهولت در تست پروژه سعی شده یک دیتابیس به نام **Fakedb.sqlite3** نوشته شده که شامل چند صد دیتای فیک است. (مسیر این دیتابیس در شاخه اصلی پروژه قرار دارد)

برای تست این دیتابیس مراحل زیر را دنبال کنید

۱) اگر پروژه **Run** است آنرا متوقف کنید

۲) دیتابیس اصلی را پاک یا در صورت نیاز مجدد تغییر مسیر دهید

۳) دیتابیس **Fakedb.sqlite3** را به **db.sqlite3** تغییر نام دهید.

۴) پروژه را دوباره **Run** کنید.

توجه : این دیتابیس با دیتا های فیک ساخته شده در سایت <https://mockaroo.com/> است. بنابراین ممکن است در بخش لینک ها که برای صفحه ی گوگل لینک نا مناسبی باشد.

بنده هیچ گونه مسئولیتی در قبال لینک ها ندارم.

در قسمت آخر این مستند باید چند نکته را باید شایان ذکر بشم.

نکته اول این است که پروژه به صورت **responsive** طراحی شده یعنی شما میتوانید در تمام پلتفرم ها از جمله موبایل، تبلت، تلویزیون های هوشمند و... استفاده کنید.

شاید این پروژه به اندازه کافی بزرگ یا تخصصی نباشد، اما پروژه به نحوی کد زده شده که پتاسیل افزایش امکانات و توانایی ها رو داراست، تمام این پتاسیل از کتابخانه قدرمت جنگو داراست که به هر چه راحت تر شدن کد نویسی کمک میکند این که این پروژه قابلیت توسعه فراوانی دارد چون از زبان قدرتمندی استفاده شده، به این نحو که بستر این پروژه در پلتفرم گوشی باشد و نیازی به مرورگر نباشد. و یا با استفاده از پیامرسان قدرتمند تلگرام تمام کارایی های موجود را به ربات های تلگرام انتقال دهیم تا حتی نیاز به نصب برنامه ای خاصی روی تلفن همراه نباشد. این قابلیت ها را میتوان با استفاده از زبان مشترک بین اپلیکیشن های یعنی **API**^۸ اجرایی کرد، حتی میتوان این اطلاعات را در اختیار دیگر توسعه دهندگان قرار داد و آن ها هم از این بانک اطلاعاتی قوی استفاده نمایند.

از دیگر امکاناتی که میشود اضافه کرد بهبود بخش "نمیدونم کجا برم؟؟؟" به صورتی که با استفاده از مختصات کاربر نزدیک ترین مکان ثبت شده را پیشنهاد دهد، میتوان بخش ها چون رویداد ها، نمایشگاه ها و غیره را نیز اضافه کرد و با کاربران اطراف حوزه آن مکان اطلاع داد. میتوان بخش جذابی چون پر بازدید ترین مکان ها به پروژه افزود تا اگر توریستی با ایران آشنا میشود و قصد سفر دارد با مهم ترین مکان های گردش گری ایران آشنا شود. همچنین قابلیت افزودن چندین زبان، میتوان بخش مهمی چون نظرات کاربران یا سیستم نمره دهی به نحوه سرویس دهی یا میزان رسیدگی در سایت قرار داده شود تا در شناخت و درک بهتر به کاربر کمک کند، البته این بخش فقط به کاربران کمک نکند، بلکه مسئولین نیز میتوانند با دریافت بازخورد ها با دید باز تری مشکلات را حل کنند و ایرانی زیبا تر برای ایرانیان و توریست ها بسازیم.

به طور کلی هدف اصلی این پروژه در وهله اول آشنا سازی مکان های مهم ایران که میتواند گردشگری یا هر چیز دیگری به مردم خوب ایران تا به راحت ترین شکل ممکن مشکلات خود راه حل کنند یا با مکان

^۸ API مخفف واژگان Application Programming Interface است که به صورت تحت الفظی می توان آن را به «رابط برنامه نویسی

نرم افزار» ترجمه کرد.

های گردشگری ایران آشنا شوند و در وهله بعدی آشنا سازی توریست با ایران در جهت درآمد زایی کلان مانند کشور هایی چون ترکیه و ایتالیا که فقط با صنعت توریسم درآمدی برابر با صنعت نفتی ما دارند و شناساندن تاریخ و تمدن غنی ایران به جهانیان.

پیشنهاد میشه قبل از شروع به استفاده از پروژه حتما قسمت راهنمای کاربردی را بخوانید تا دچار مشکل نشوید..

موفق و پیروز باشید، امیدوارم تجربه ی خوبی از پروژه داشته باشید.

محمدرسول طارمی

شهریور ۹۸

<https://github.com/miksart/project>

https://www.amazon.com/gp/product/1491962291/ref=as_li_tl?ie=UTF8&tag=codewithmosh-20&camp=1789&creative=9325&linkCode=as2&creativeASIN=1491962291&linkId=c9cb035eec225b25d53274111951d95c

https://github.com/CoreyMSchafer/code_snippets

https://www.amazon.com/gp/product/1593276036/ref=as_li_tl?ie=UTF8&camp=1789&creative=9325&creativeASIN=1593276036&linkCode=as2&tag=codewithmosh-20&linkId=f61bef4447a86881181a90444ca06ed0

https://www.amazon.com/gp/product/1593275994/ref=as_li_tl?ie=UTF8&tag=codewithmosh-20&camp=1789&creative=9325&linkCode=as2&creativeASIN=1593275994&linkId=e55ad39c6469889a09dd16f4a6e4cbef

https://www.amazon.com/gp/product/B077Z55G3B/ref=as_li_tl?ie=UTF8&tag=codewithmosh-20&camp=1789&creative=9325&linkCode=as2&creativeASIN=B077Z55G3B&linkId=b26f8a05bec004e37060386fbc60f9e3

https://www.amazon.com/gp/product/1549617214/ref=as_li_tl?ie=UTF8&tag=codewithmosh-20&camp=1789&creative=9325&linkCode=as2&creativeASIN=1549617214&linkId=58b1ecf739a460e84d8dceb8df156ec0

https://www.amazon.com/Learning-Python-5th-Mark-Lutz/dp/1449355730/ref=sr_1_1?keywords=python&qid=1566940298&s=gateway&sr=8-1

https://www.amazon.com/Django-Beginners-Build-websites-Python/dp/1983172669/ref=sr_1_1?crid=34X0ZCT4ZX4QF&keywords=python+django&qid=1566940344&s=gateway&sprefix=python+djan%2Caps%2C322&sr=8-1

https://www.amazon.com/Django-Web-Development-Cookbook-practical/dp/1788837681/ref=sr_1_2?crid=34X0ZCT4ZX4QF&keywords=python+django&qid=1566940344&s=gateway&sprefix=python+djan%2Caps%2C322&sr=8-2

https://www.amazon.com/Test-Driven-Development-Python-Selenium-JavaScript/dp/1491958707/ref=sr_1_4?crid=34X0ZCT4ZX4QF&keywords=python+django&qid=1566940344&s=gateway&sprefix=python+djan%2Caps%2C322&sr=8-4

<https://www.python.org/>

<https://www.djangoproject.com/>

<https://www.youtube.com/>

https://www.youtube.com/channel/UCCezIgC97PvUuR4_gbFUs5g

<https://www.youtube.com/channel/UCWv7vMbMWH4-V0ZXdmDpPBA>

<https://www.w3schools.com>

<https://github.com/>

<https://.quora.com>

<https://medium.com/>

<https://adamj.eu/>

<https://wsvincent.com/>

<https://stackoverflow.com/>

<https://ultimatedjango.com/>

<https://programmingwithmosh.com/>

<https://codewithmosh.lpages.co/python-cheat-sheet/>

[https://en.wikipedia.org/wiki/Django_\(web_framework\)](https://en.wikipedia.org/wiki/Django_(web_framework))

<https://faradars.org/courses/fvpht9611-django-web-based-framework-using-python>

<https://hamrahyad.com/courses/%D8%A2%D9%85%D9%88%D8%B2%D8%B4-%D8%AC%D9%86%DA%AF%D9%88/>

<http://arzandehnia.com/>

<https://amoozesh.org/django/>

<https://zarinserver.com/blog/what-is-django/>

<https://git.ir/django/>

<https://learnfiles.com/course/%D8%A2%D9%85%D9%88%D8%B2%D8%B4-%D8%AC%D9%86%DA%AF%D9%88/>

<https://software.intel.com>

<https://hackernoon.com/top-10-python-web-frameworks-to-learn-in-2018-b2ebab969d1a>

<https://www.shabakeh-mag.com>

<http://python.coderz.ir/>

با تشکر از راهنمایی که دوست خوبم علیرضا ربیعی در ساخت این پروژه به من داشت.