جنگو

فریم‌ورک جنگو یکی از پراستفاده‌ترین فریم‌ورک‌های وبه. جنگو خیلی از مسائلی که برای توسعه وب مهمه (مثل امنیت، ارتباط با دیتابیس و …) رو خودش هندل می‌کنه و کمک می‌کنه تا توسعه‌دهنده روی پیاده‌سازی لاجیک بیزینس تمرکز داشته باشه.

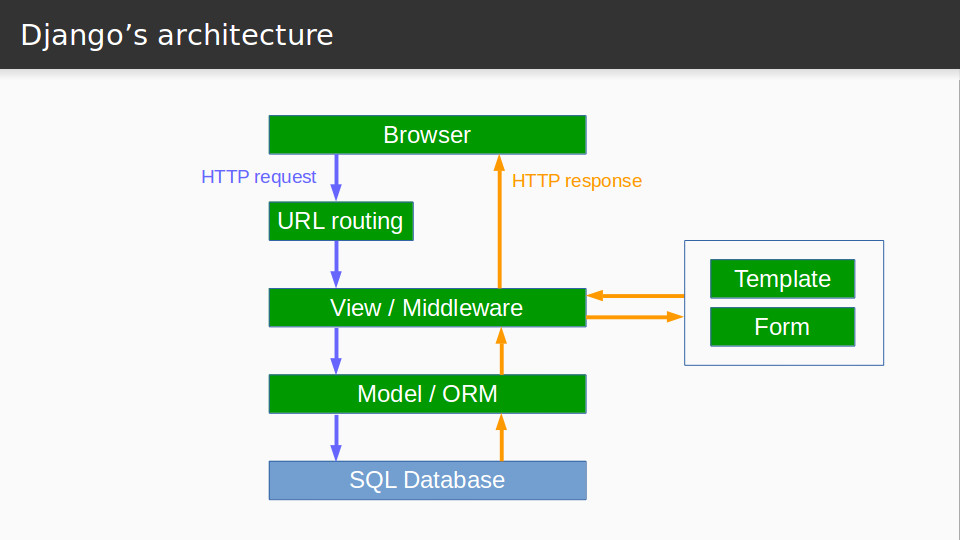
توی این دوره با استفاده از آموزش [سایت رسمی جنگو](https://www.djangoproject.com/) با ساختار کلی جنگو آشنا می‌شیم و از ویژگی‌هاش برای پیاده‌سازی یک پروژه‌ی ساده استفاده می‌کنیم.

ساختار پروژه

مثل هر فریم‌ورک دیگه‌ای جنگو هم یه ساختار برای فایل‌هاش داره که برای تعریف پروژه باید اونو رعایت کنیم.

|  |
| --- |
| این بخش [آموزش جنگو](https://docs.djangoproject.com/en/3.2/intro/tutorial01/) را برای آشنایی با ساختار کلی فریم‌ورک جنگو مطالعه کن و مطابق اون یک پروژه و یک اپ بساز و اون را با استفاده از دستور runserver تست کن. |

مسیر کلی که یک ریکوئست در جنگو طی می‌کنه تا به ریسپانس بره رو توی دیاگرام زیر می‌تونی ببینی. در ادامه توی بخش‌های مختلف این دیاگرام عمیق‌تر می‌شیم.



مدل‌ها

برای تعریف مدل‌هامون توی دیتابیس و استفاده از اونها جنگو ابزارهای خوبی در اختیارمون قرار می‌ده و ORM خوبی داره.

|  |
| --- |
| در مورد ORM این [مقاله](https://www.fullstackpython.com/object-relational-mappers-orms.html) رو خوبه مطالعه کنی. |

توی جنگو مدل‌ها رو باید توی فایل models.py تعریف کنیم و با استفاده از دستورهای makemigrations و migrate جدول‌های مربوط به اونها رو توی دیتابیس بسازیم. در نهایت برای استفاده از مدل‌ها از ORM جنگو استفاده می‌کنیم.

|  |
| --- |
| بخش‌های Database setup و Creating models و Activating models از این بخش [آموزش جنگو](https://docs.djangoproject.com/en/3.2/intro/tutorial02/) را برای آشنایی با نحوه‌ی ساخت مدل در دیتابیس با جنگو مطالعه کن. کانکشن پروژه‌ی خودت رو برای یک دیتابیس postgres کانفیگ کن. |

برای اینکه دیتابیس عوض کنی کافی هست که تو فایل settings بری تو قسمت DATABASE و configurationsها را ست کنی.

همون طور که دیدی برای مدیریت تغییرات روی دیتابیس جنگو از یه پرکتیسی استفاده می‌کنه به نام مهاجرت!

|  |
| --- |
| برای آشنایی بیشتر با migration و خوبی‌هاش می‌تونی این [مقاله](https://realpython.com/django-migrations-a-primer/) رو مطالعه کنی. |

محیط تعاملی جنگو و پنل ادمین دو اینترفیس بسیار خوب برای کار با APIهای جنگو و به خصوص ORM هستن.

|  |
| --- |
| بخش‌های Playing with the API و Introducing the Django Admin از این بخش [آموزش جنگو](https://docs.djangoproject.com/en/3.2/intro/tutorial02/) را برای آشنایی با محیط تعاملی و پنل ادمین جنگو مطالعه کن. |

پنل ادمین

پنل ادمین جنگو ابزاریه که کمک می‌کنه بدون نیاز به زدن کد فرانت‌اند و با کد بکندیِ خیلی کم پنل‌های مدیریتی (نه چندان زیبا) پیاده‌سازی کنیم! قابلیت سفارشی‌سازی ادمین خیلی زیاده و تقریباً هر تغییری که بخوایم می‌تونیم توش بدیم.

|  |
| --- |
| این بخش [آموزش جنگو](https://docs.djangoproject.com/en/3.2/intro/tutorial07/) رو برای آشنایی بیشتر با قابلیت‌های پنل ادمین جنگو برای سفارشی‌سازی مطالعه کن. |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| از توی اپ بازار بخش نظرات رو ببین. کاربرها می‌تونن برای برنامه‌های مختلف نظر ثبت کنن.   1. یه پروژه جدید بساز و به اون اپ نظرات رو اضافه کن. 2. مدل‌های کاربر، برنامه و نظر رو تعریف کن و مایگریشن‌هاش رو بساز و روی دیتابیس اعمال کن. فیلدهای پیشنهادی رو می‌تونی توی جدول زیر ببینی:  |  |  | | --- | --- | | فیلدها | مدل | | name | application | | user(fk to django django.contrib.auth.models.User), application, text, status, creation\_date | comment |   برای فیلد status از [choices](https://docs.djangoproject.com/en/4.0/ref/models/fields/#choices) استفاده کن. اینکه مقادیر ممکن چی می‌تونه باشه رو خودت بهش فکر کن و طراحیش کن. (کامنت‌های بازار بعد از اینکه ثبت میشن توسط یه نفر توی بازار تایید میشن و بعد نمایش داده می‌شن. اگر کامنتی تایید نشه هیچ وقت نمایش داده نمی‌شه.)   1. یه کدی بزن که مدل‌ها رو توی پنل ادمین نمایش بده. 2. یه مقداری ریکورد اضافه کن به دیتابیس! به هر روشی که دوست داری. 3. به بخش نظرات پنل ادمین فیلتر بر اساس وضعیت کامنت اضافه کن. |

ویوها

ویوها در جنگو دربردارنده‌ی لاجیک به دست آوردن ریسپانس با توجه به ریکوئست هستن.

|  |
| --- |
| این بخش [آموزش جنگو](https://docs.djangoproject.com/en/3.2/intro/tutorial03/) را برای آشنایی بیشتر با ویوها و تمپلیت‌ها در جنگو مطالعه کن. |

یکی از راه‌های معمول ورودی گرفتن از کاربر استفاده از فرم‌هاست. جنگو کلاس‌ها و فانکشن‌هایی داره که کار ولیدیت کردن، تمیز کردن، ذخیره کردن و تولید فرم رو برامون راحت‌تر می‌کنه.

|  |
| --- |
| بخش Write a minimal form رو از این بخش [آموزش جنگو](https://docs.djangoproject.com/en/3.2/intro/tutorial04/) برای آشنایی بیشتر با فرم‌ها در جنگو مطالعه کن. |

Class Base Views

ویوهای جنگو رو می‌شه علاوه بر مدل function based به صورت class based هم پیاده‌سازی کرد. پیاده‌سازی به صورت class based کمک می‌کنه با استفاده از generic views جنگو ویژگی‌های مورد نیازمون رو بدون پیاده‌سازی و با ارث‌بری داشته باشیم.

|  |
| --- |
| این [مقاله](https://medium.com/@ksarthak4ever/django-class-based-views-vs-function-based-view-e74b47b2e41b) رو مطالعه کن تا با تفاوت‌های ویوهای function based و class based بیشتر آشنا بشی. |

|  |
| --- |
| توی تمرین‌های زیر زیبا بودن اینترفیس زیاد مهم نیست. واسه همین وقتت رو خیلی سر css نذار.   1. یه صفحه اضافه کن که لیست برنامه‌ها توش باشه. روی هر کدوم از برنامه‌ها که زدیم وارد یک صفحه‌ی دیگه بشیم که نظرات تأیید شده‌ی اون اپ به ترتیب تاریخ توش نمایش داده بشه. 2. توی همون صفحه‌ی نظرات اپ یه فرم برای اضافه کردن نظر باشه. اگه کاربر نظر رو ثبت کرد به همون صفحه برگرده و کامنت خودش رو با هر وضعیتی که داره ببینه. برای لاگین کردن کاربر از تمپلیت‌های آماده‌ی جنگو استفاده کن. خوبه از همین الان حواست به موارد امنیتی باشه. مثلا توی [این داک](https://github.com/OWASP/CheatSheetSeries/blob/master/cheatsheets/Input_Validation_Cheat_Sheet.md) به طور خلاصه مواردی که باید برای ورودی‌های کاربر رو چک کنی نوشته شده. 3. به بخش نظرات پنل ادمین اکشن تأیید و اکشن رد نظرات انتخاب شده اضافه کن. مثلا یه دکمه جلو هر رکورد اضافه کن. برای این کار باید یه ویو بنویسی که اگه روی اون دکمه کلیک شد اطلاعات نظر براش فرستاده بشه. توی این ویو حواست به لاگین بودن کاربری که داره بررسی میکنه و is staff بودنش باشه. |

Logging

با کانفیگ کردن لاگینگ جنگو به راحتی می‌تونیم لاگ‌های پیش‌فرض جنگو رو با فرمت دلخواه‌مون و در جای دلخواه‌مون ببینیم و توی پروژه‌مون هم هر جا نیاز بود به سادگی لاگ اضافه کنیم.

|  |
| --- |
| این [مقاله](https://www.askpython.com/django/django-logging) رو برای آشنایی بیشتر با لاگ در جنگو مطالعه کن. |

|  |
| --- |
| لاگ پروژه‌ی خودت رو کانفیگ کن و برای اضافه شدن نظر و رد و تأیید شدن نظرات لاگ بذار.  (**امتیازی**: یه اسکریپت بنویس که فایل لاگ رو بخونه و میانگین زمان اضافه شدن نظر تا بررسی اون (رد یا تأیید) رو دربیاره. به نظرت این بررسی رو توی خود کد تایید یا رد می‌ذاشتیم بهتر بود؟ نه لزوما! چرا؟) |

REST API

ما توی شرکت برای ای‌پی‌آی‌های HTTP از استاندارد REST و فرمت JSON استفاده می‌کنیم.

|  |
| --- |
| این [راهنما](https://realpython.com/api-integration-in-python/) رو برای آشنایی با نحوه‌ی پیاده‌سازی REST API مطالعه کن. |

|  |
| --- |
| یه ویو بنویس که طبق استاندارد REST با متد GET لیست نظرات یک برنامه رو به صورت json بده و با متد POST اطلاعات یه نظر جدید رو به فرمت json بگیره و ذخیره کنه. |

WSGI

همون طور که دیدی تا الان داشتی کل پروژه‌ی جنگو رو با manage.py runserver ران می‌کردی. این کامند درواقع یه وب‌سرور (که با اپلیکیشن‌سرور فرق داره، و فرقش رو توی فصل Web گفتیم) خیلی سبک بالا میاره و در هر لحظه تنها یک ریکوئست رو جواب می‌ده. پایتون تک ترده و این یعنی هر پروسس پایتونی که ران می‌کنی فقط از یه ترد استفاده می‌کنه(به خاطر وجود [GIL](https://www.geeksforgeeks.org/what-is-the-python-global-interpreter-lock-gil/#:~:text=Python%20Global%20Interpreter%20Lock%20(GIL)%20is%20a%20type%20of%20process,be%20executed%20at%20a%20time.)). ولی خب همون طور که میدونی یه سی‌پی‌یو بیشتر از یه ترد داره و همچنین اکثر اوقات سی‌پی‌یو مشغول نیست. مخصوصا وقتی کار سرور، سرو کردن وب اپلیکیشن باشه. در اینجور مواقع بیشتر زمانی که پردازش یه ریکوئست زمان می‌بره صرف معطل بودن برگردونده شدن جواب از دیتابیس یا سرویس‌های دیگه است. بنابراین خیلی مهمه که ما جوری یه وب‌سرور راه بندازیم و وصلش کنیم به اپلیکیشنمون که در هر لحظه کلی ریکوئست هم‌زمان رو پردازش کنه و قبول کنه.

Gunicorn یکی از این سرورها است که پروتوکل wsgi رو می‌فهمه و می‌تونه با استفاده از اون چند پروسس پایتونیِ اپلیکیشن تو رو ران کنه.

|  |
| --- |
| ۱. یه سرچی بکن و با مفهوم WSGI آشنا شو.  ۲. با گونیکورن پروژه‌ات رو بالا بیار که همزمان بتونه به ۵ درخواست جواب بده. (چجوری تستش می‌کنی؟)  ۳. اگه بخوایم یه دیتایی رو بین پروسس‌هایی که به ریکوئست‌ها جواب میدن به اشتراک بذاریم باید چی کار کنیم؟ مثلا فرض کن می‌خوایم تعداد ریکوئست‌هایی که به جنگو می‌رسه رو بشماریم. |

Dockerize

واضحه که بهتره پروژه‌مون جوری باشه که بتونیم توی پلتفرم‌های مختلف بیاریم‌ش بالا و چه راهی بهتر از داکرایز کردن اون!

|  |
| --- |
| این [راهنما](https://docs.docker.com/samples/django/) رو برای آشنایی با نحوه‌ی داکرایز کردن پروژه‌ی جنگو مطالعه کن. |

|  |
| --- |
| برای پروژه‌ی خودت فایل‌های Dockerfile و docker-compose.yml رو بنویس و با استفاده از docker-compose پروژه‌ت رو بیار بالا (به جای دیتابیس sqlite که توی راهنما گفته شده بهتره از دیتابیس postgres استفاده کنی). |

Security

برای بردن کدمون رو پروداکشن (یعنی محیطی که واقعا اپ اجرا می‌شه و به ریکوئست‌های کاربرا جواب می‌ده) به جز اینکه باید محیط توسعه‌مون رو با استفاده از ابزارهایی مثل داکر نزدیک پروداکشن کنیم، ملاحظات امنیتی بیشتری رو هم باید حواسمون باشه.

|  |
| --- |
| [فصل ۱۶ کتاب django for professionals](https://dl.ebooksworld.ir/books/Django.for.Professionals.4.0.William.S.Vincent.WelcomeToCode.9781735467238.EBooksWorld.ir.pdf) هم به چک لیست دیپلوی می‌پردازه و هم دغدغه‌های دیگه‌ی امنیتی که باید موقع توسعه حواسمون بهش باشه رو می‌تونی بخونی. خوبه که روی سرفصل‌ها دقیق‌تر بشی و لینک‌هایی هم که داده بخونی. شاید بشه گفت مهم‌ترین اطلاعاتی که یه مهندس نرم‌افزار باید همیشه حفظ باشه و بهشون مسلط باشه همین موارده. چون در طول توسعه اگه به مشکل بخوریم می‌تونیم بریم بخونیم و مشکل رو حل کنیم ولی اگه یه مورد امنیتی رو رعایت نکنیم زمانی می‌فهمیم که خیلی دیر شده. پس نفست رو حبس کن و توی این مباحث عمیق شو :)  ***مطالعه‌ی بیشتر***  The OWASP Top Ten is a standard awareness document for developers and web application security. It represents a broad consensus about the most critical security risks to web applications. [This cheat sheet](https://cheatsheetseries.owasp.org/IndexTopTen.html) will help users of the OWASP Top Ten identify which cheat sheets map to each security category. This mapping is based on the [OWASP Top Ten 2021 version](https://owasp.org/www-project-top-ten/). |

|  |
| --- |
| هر اقدامی که لازمه رو برای آماده‌سازی اپ برای دیپلوی رو انجام بده. همچنین اگه قبلا سوتیِ امنیتی دادی و الان فهمیدی برگرد و درستش کن. |

Cache

اگر از شما نتیجه ۵ × ۳ را بپرسیم، پاسخ شما ۱۵خواهد بود. قطعا شما برای پاسخ به پرسش ما محاسبه انجام نمی دهید، در واقع شما در طول زندگی خود بارها آن را حساب کرده اید و دیگر نیاز به محاسبه ندارید. احتمالا شما به راحتی و بدون هیچ گونه فرایند پیچیده ی ذهنی نتیجه را به یاد می آورید. این دقیقا همان فرایندی است که در کش کردن اتفاق می افتد.

کش کردن (Caching) به فرآیند ذخیره‌سازی موقت یک نسخه کپی از داده‌ها یا فایل‌ها در حافظه(معمولا حافظه‌ای با سرعت بالا در استخراج داده) گفته می‌شود تا برنامه‌ها بتوانند سریع‌تر به آن‌ها دسترسی پیدا کنند.

|  |
| --- |
| برای آشنایی بیشتر با اینکه cache چی هست [این لینک](https://auth0.com/blog/what-is-caching-and-how-it-works/) رو مطالعه کن.  [این لینک](https://docs.djangoproject.com/en/4.0/topics/cache) هم رفرنس خود جنگو در زمینه‌ی استفاده از کش هست. یه نگاه کلی بهش بنداز و طبیعیه که همه‌اش و نیازی نیست که بخونی. |

|  |
| --- |
| ۱. به ویوی تمرین شماره‌ی ۱ بخش Class Based Views که نظرات تایید شده رو نشون میده کش رو اضافه کن. میتونی از دکوراتور @cache\_page استفاده کنی. بکند کش رو فعلا میتونی LocMemCache بذاری.  ۲. الان اگه فانکشنالیتی تایید کردن نظر رو چک کنی می‌بینی که تا زمانی که کش اکسپایر نشده با تایید نظر جدید کاربر اون رو نمیبینه. برای حل این مشکل یه کم باید استفاده از کش رو با جزئیات بیشتر پیاده کنی. پس به جای اینکه از دکوریتور استفاده کنی خودت از توابع get و set استفاده کن و کلید کش مناسب اون برنامه رو تولید کن. وقتی هم که یه نظری تایید میشه کش مربوط به اون برنامه رو پاک کن که با ریکوئست بعدی هم کش آپدیت بشه و هم دیتای به‌روز به کاربر نشون داده بشه. |

|  |
| --- |
| ***مطالعه‌ی بیشتر***  کش فضای بی‌نهایت که نداره! حالا اگه فضای کش پر بشه و نیاز باشه که یه رکورد جدید ست بشه کدوم رکورد باید پاک بشه؟ اینجا استراتژی‌/پالیسی‌های مختلف eviction کش وجود داره. چندتا از معروف‌ترین‌هاش LRU, LFU, FIFO هستن که میتونی توضیحشو رو توی [ویکی‌پدیا](https://en.wikipedia.org/wiki/Cache_replacement_policies) بخونی.  *There are 2 hard problems in computer science: cache invalidation, naming things, and off-by-1 errors.* |

گاهی اوقات بار سیستم انقدر بالا می‌ره که ریکوئست زدن به یه کش مرکزی هرچقدر هم که سریع‌ باشه باز سربار زیادی داره و هم به کش مرکزی فشار زیادی وارد می‌کنه. در اینجا از یه تکنیک استفاده می‌کنن به نام کش چند لایه. این تکنیک اینجوریه که چند مدل کش مختلف که به ترتیب سرعت مرتب شدن پشت هم قرار میگیرن و کد موقع گرفتن اطلاعات از اولین لایه شروع می‌کنه و اگه نداشت سراغ لایه‌ی بعدی می‌ره.

|  |
| --- |
| [ویکی‌پدیا](https://en.wikipedia.org/wiki/Cache_hierarchy) توضیح دقیق‌تری آورده که می‌تونی بخونی و [این پیاده‌سازی](https://github.com/nuclearfurnace/django-layered-cache/blob/master/layered_cache/backends/cache.py) ابر ساده هم میتونه بهت دید بده!  [Multilevel Cache Performance - Georgia Tech - HPCA: Part 4](https://www.youtube.com/watch?v=nrj78Vbik_U&t=10s) |

|  |
| --- |
| ۱. حالا برای همون ویو که کش نوشتی، یه کش دولایه پیاده کن. کش لایه‌ی اولت رو بذار یه دیکشنری پایتون و کش لایه‌ی دوم هم از همون api کش جنگو استفاده کن.  ۲. احتمالا متوجه می‌شی که چون ریکوئست‌ها توسط پروسس‌های مجزا پردازش می‌شن بنابراین این دیکشنری که تعریف کردی بین پروسس‌ها شیر نیست و اگه یه نظری تایید بشه نمی‌تونیم دیکشنری همه‌ی پروسس‌ها رو آپدیت کنیم. حل کردن این مشکل هیچ راه حل آسونی نداره. درواقع همیشه یه trade-off هست بین درستی و یکپارچگی دیتا و پرفورمنس. ساده‌ترین کار اینکه همون دیکشنری رو یه جوری پیاده کنی که هر کلید یه زمان انقضا داشته باشه و این زمان انقضا رو مقدار معقول و کمی بذاری. یه جوری برنامه رو بنویس که این عدد رو از توی فایل ستینگ جنگو بشه عوضش کرد. |

  DRF (امتیازی!)

برای پیاده‌سازی ای‌پی‌آی‌های RESTful با استفاده از جنگو، کتابخونه‌ی django-rest-framework توسعه داده شده. با استفاده از اون می‌تونیم URL هامون رو ساده‌تر تعریف کنیم، با تعداد خط کد کمتر ویوهای جنرال‌تری داشته باشیم و با استفاده از سریالایزرها دیتا رو راحت‌تر تبدیل کنیم. در نهایت کدی که با این کتابخونه زده شده به مراتب تمیز تر از خود جنگو است!

|  |
| --- |
| این [راهنما](https://www.django-rest-framework.org/tutorial/quickstart/) رو برای آشنایی با کتابخونه‌ی django-rest-framework مطالعه کن. برای تسلط بیشتر می‌تونی [راهنمای کامل](https://www.django-rest-framework.org/tutorial/1-serialization/) رو مطالعه کنی. |

|  |
| --- |
| ویوسِت لیست نظرات یک برنامه، با امکان اضافه کردن نظر جدید و حذف نظر با استفاده از drf و طبق استاندارد رست‌فول پیاده‌سازی کن. |