دانشگاه یزد

دانشکده مهندسی کامپیوتر

پایان‌نامه کارشناسی مهندسی کامپیوتر

گرایش نرم افزار /فناوری اطلاعات

طراحی خزشگروب

نگارش:

ابوالفضل ابوئی

استاد راهنما:

دکتر محمد قاسم زاده

پاییز ۱۴۰۰


بسم الله الرحمن الرحيم

کلیه حقوق مادی و معنوی بر نتایج

مطالعات، ابتکارات و نوآوری های

ناشی از تحقیق موضوع این پایان نامه

متعلق به دانشکده مهندسی کامپیوتر دانشگاه یزد است.

تقدیم به

مقدس ترین واژه ها در لغت نامه دلم، مادر مهربانم که زندگیم را مدیون مهر و عطوفت آن می دانم .

پدر، مهربانی مشفق، بردبار و حامی .

خواهرانم همراهان همیشگی و پشتوانه های زندگیم .

سپاس گذاری

شکر شایان نثار ایزد منان که توفیق را رفیق راهم ساخت تا این پایان نامه را به پایان برسانم .

از استاد فاضل و اندیشمند جناب آقای دکتر قاسم زاده به عنوان استاد راهنما که همواره نگارنده را مورد لطف و محبت خود قرار داه اند و همه اساتید محترم گروه مهندسی کامپیوتر کمال تشکر را دارم.

چکیده

در مسیر ادامه تحصیلات تکمیلی و مهاجرت یکی از اقداماتی که باید انجام داد پیدا کردن استادی است که حداقل حوزه پژوهشش با حوزه پژوهش ما تقریبا یکی باشد و پیدا کردن اساتید امری زمان بر میباشد؛ مخصوصا اینکه باید دانشگاه های زیادی را جست و جو کرد و اساتیدی که حوزه پژوهشان با ما یکی است را بیابیم؛ و این خود زمان زیادی ممکن است از ما بگیرد

برای سهولت کار، ما میتوانیم سایتی طراحی کنیم که با وارد کردن حوزه، دانشگاه، شهر و یا کشور لیست اساتید به همراه ایمیل و حوزه کاریشان را نمایش بدهد

برای طراحی موتور جستجوی اساتید خزشگری طراحی شده است که لیست اساتید به همراه راه های ارتباطیشان را پیدا کرده و وارد پایگاه داده سایت میکند

واژه های کلیدی : مهاجرت – خزشگر وب – وب اسکرپینگ – موتور جستجوی اساتید

فهرست

چکیده ..... ج
فصل اول ..... د
۱.۱. معرفی ۱
۱.۱.۱. پیشگفتار ۱
۲.۱.۱. هدف ۱
۲.۱. خزشگر وب ۱
۱.۲.۱. خزشگروب چکونه کار میکند ۱
۲.۲.۱. سه خزشگر برتر وب ۲
۳.۱. وب اسکرپینگ ۲
۱.۳.۱. چرا وب اسکرپینگ ۳
۲.۳.۱. کاربرد وب اسکرپینگ ۳
۳.۳.۱. انواع وب اسکرپرها ۷
۴.۳.۱. آیا وب اسکریپینگ قانونی است؟ ۸
۵.۳.۱. آیا اسکرپینگ و کراولینگ یکی هستند؟ ۸
۶.۳.۱. محافظت از وب سایت ها در برابر وب اسکرپینگ ۸
فصل دوم ۱۰
۱.۲. بک – اند (Back – end) ۱۱
۱.۱.۲. پایتون ۱۱
۲.۱.۲. کتابخانه های مورد نیاز برای وب اسکرپینگ ۱۲
۳.۱.۲. پایگاه داده ۱۴
۲.۲. فرانت – اند (Front - End) ۱۷

HTML ۱۷
فصل سوم ۱۸
۱.۳. پلتفرم و ابزار ها ۱۹
۱.۱.۳. اشنایی با نحوه کار وب اسکرپینگ ۱۹
۲.۱.۳ خزشگر ۲۱
۳.۱.۳. پیاده سازی یک خزشگر کامل ۲۸
۵.۱.۳. نتیجه گیری ۳۲
مراجع ۳۳

فصل اول

۱.۱. معرفی

۱.۱.۱. پیشگفتار

مسیر تحصیلات تکمیلی و مهاجرت تحصیلی مسیری پر چالش است و در این مسیر زمان بسیار با ارزش است. یکی از چالش هایی که در برابر دانشجویان متقاضی مهاجرت وجود دارد مکاتبه با اساتید دانشگاه مورد هدفشان است.

از انجایی که پیدا کردن اساتید از خود سایت دانشگاها کاری زمان بر است وجود یک سایتی که تمام اساتید به همراه راه ارتباطی و حوزه کاریشان را گرد اوری کرده باشد، بسیار خالی است و وجود این سایت میتواند به طور چشمگیری به دانشجویان در پیدا کردن اساتید در کمترین زمان ممکن کمک کند.

۲.۱.۱. هدف

هدف طراحی یک موتور جستجو اساتید جمع اوری مشخصات اساتید است که میتواند برای دانشجویان بسیار مفید و ارزشمند باشد و کمکی هرچند کوچک برای انان که برای بدست اوردن علم، مرز ها را پشت سر میگذارند باشد

۲.۱. خزشگر وب

خزشگر وب به ربات هایی میگویند که به صورت خودکار به صفحات وب میروند و با توجه المان های داده شده اطلاعات صفحات را استخراج میکنند در واقع خزشگرهای وب یک اسکریپت اتوماتیک هستند که با الگوریتم های مشخص در وبسایت هایی که برایشان تعریف شده است به گردش میپردازند و محتوای صفحات را استخراج و در پایگاه داده ذخیره میکنند .

۱.۲.۱. خزشگروب چکونه کار میکند

فضای اینترنت دائماً در حال تغییر و تحول می‌باشد و روز به روز بر گستره ان اضافه میشود . در این بین از آن‌جایی که به تعداد بسیار زیادی صفحه وب در اینترنت وجود دارد، خزشگرها کار خود را از بینURL [1]های آشنا و معروف‌تر آغاز می‌کنند. در این پروسس خزشگرها با شروع کار ایندکس‌گذاری خود از صفحات آشنا، در

[1] Uniform Resource Locator

حین این بررسی با هایپرلینک‌هایی رو به رو می‌شوند. این ربات‌ها در ادامه، URLهای مربوط به این هایپرلینک‌ها را نیز به لیست جست و جوی خود اضافه کرده و بدین ترتیب دامنه فعالیت خود را بتدریج گسترش می‌دهند.

۲.۲.۱. سه خزشگر برتر وب

۱. Google bot

یکی از محبوب ترین کراولر وب، googlebot می باشد. این خرشگرگوگل لیستی از محتوا را برای موتور جستجوی گوگل تهیه میکند. خزنده وب گوگل ابزار های بسیاری دارد که جهت بررسی و کنترل محتوا می توانیم از آنها استفاده کنیم.

۲. Ahrefs

این ابزار برای برسی و کنترل و همچنین تحلیل بک لینک های سایت مورد استفاده قرار میگیرد که بهترین ایندکس های بک لینک را در مقایسه ی با دیگر ابزار ها دارد.

۳. SEMrush

این ابزار یکی از ابزار هایی است که با خزشگرش اطلاعات سایت را برای انالیز و تحلیل جمع اوری میکند.

۳.۱. وب اسکرپینگ

به فرایندی که میتوان با استفاده از داده ها و محتوای صفحات وب را استخراج کرد وب اسکرپینگ میگویند که میتوان این اطلاعات را در فرمتی که مورد نظر خودمان است ذخیره کرد.

۱.۳.۱. چرا وب اسکرپینگ

از انجایی که حجم اطلاعات موجود در فضای وب از زتابایت[۲] عبور کرده است و ما در زندگی روزمره به همه اطلاعات و داده ها نیاز نداریم وب اسکرپینگ یک روش مفید برای گرد اوری هدفمند داده ها از سطح وب میباشد.

۲.۳.۱. کاربرد وب اسکرپینگ

از وب اسکرپینگ در بسیاری از کسب و کارهای اینترنتی استفاده میشود تا بتوان داده های صفحات وب را بر اساس پارامترهای خودمان فیلتر و استخراج کنیم، که با این عمل میتوان در اولین فرصت به داده های مورد نیازمان دسترسی پیدا کنیم.

۱. نظارت بر رقبا

با نظارت بر رقبا میتوانیم از استراتژی انها با خبر شویم و به جدید ترین داده های انها دسترسی پیدا کنیم که با این عمل نسبت به موارد زیر بینش پیدا میکنیم.

- قیمت گذاری رقبا
- استراتژی تبلیغاتی انها
- و....

مثال نظارت بر رقبا در صنعت تجارت الکترونیک

[2] 1 Zettabyte = 10^{21} byte

اگر صاحب یک فروشگاه آنلاین هستیم، می‌توانیم اطلاعات محصولات مانند فروشندگان، تصاویر و قیمت آن‌ها را از وب‌سایت‌هایی مانند دیجی کالا جمع‌آوری کنیم. از این طریق می‌توانیم اطلاعات دست اولی را از بازار به دست آوریم و بر همین اساس، استراتژی بازاریابی خود را تنظیم کنیم.

۲. بررسی نظرات و تمایلات افراد در شبکه های اجتماعی

با جمع اوری داده ها و نظرات شبکه های اجتماعی میتوانیم انها را تجزیه و تحلیل کنیم و با تحلیل انها سوگیری مردم و جامعه را در مورد مسائل جامعه درک کنیم.

- تصمیم گیری برای سرمایه‌گذاری(Investment Decision Making)
- نظارت بر محصولات(Product Monitoring)
- نظارت بر شرکت و برند(Brand and Company Monitoring)
- توسعه محصول(Product Development)
- سیاست و مبارزات(Politics and Campaigns)

بررسی نظرات و تمایلات افراد در شبکه های اجتماعی۱

۳. تحقیق بازار

یکی دیگر از کاربرد های وب اسکرپینگ تحقیق بازار است که با ان میتوانیم به دقیق ترین اطلاعات موجود دست پیدا کنیم که وب اسکرپینگ در موارد زیر به ما کمک میکند.

- تجزیه و تحلیل روند بازار(Market Trend Analysis)
- دستیابی به قیمت بازار(Market Pricing)
- تحقیق و توسعه(R&D/ Research & Development)
- نظارت بر رقبا(Competitor Monitoring)
- ورود با اطمینان و اعتماد به یک صنعت(Optimizing Point of Entry)

مثال وب اسکرپینگ برای ورود به بازار

فرض کنید شما وارد کننده گوشی هستید و میخواهید گوشی وارد کنید با بررسی سرچ ها و مقدار فروش گوشی ها میتوان پرفروش ترین مدل را پیدا کرد و بر اساس ان وارادات انجام داد.

٤. نظارت بر اخبار

با وب اسکرپینگ میتوان اخبار و اطلاعات را از سایت های خبری استخراج کرد .

از این اطلاعات میتوان در زمینه های زیر استفاده کرد.

- تصمیم‌گیری برای سرمایه گذاری(Investment Decision Making)
- تجزیه و تحلیل عقاید عمومی آنلاین(Online Public Sentiment Analysis)
- مبارزات سیاسی(Political Campaigns)

مثال نظارت بر اخبار

استفاده از داده های خبری برای تبلیغات نامزد ها در زمان رای گیری های مختلف اشاره کرد .

۵. یادگیری ماشین

کیفیت مدل‌های یادگیری ماشین، بستگی به کیفیت داده‌های تمرینی استفاده شده دارد؛ بنابراین زمانی که داده‌ها به آسانی در دسترس نیست، می توانیم از وب اسکرپینگ استفاده کنیم؛ تا اطلاعات سطح وب را برای مدل ما استخراج کند.

از وب اسکرپینگ میتوان در موارد زیر در زمینه یادگیری ماشین استفاده کرد.

- تست مدل‌های یادگیری ماشین
- ارائه تعدادی از باکیفیت‌ترین داده‌ها برای طبقه‌بندی و آموزش الگوریتم‌های پیش‌بینی

۶. بررسی سئو

از وب اسکرپرها برای اسکرپ کردن گوگل و دیگر موتورهای جستجو میتوان اسفاده کرد ؛ تا ببینیم کدام صفحات با کدام کلمات کلیدی رتبه گرفته‌اند.

۷. مقایسه قیمت‌ها

وب‌سایت‌هایی مانند alibaba.ir ، flightio.comو mrbilit.com از وب اسکرپرها(همراه باAPI ها) برای مقایسه قیمت انواع بلیط‌ها استفاده می‌کنند؛ بنابراین، با استفاده از وب اسکرپینگ،نیازی به مقایسه ۲۰ وب‌سایت مختلف برای پیدا کردن بهترین بلیط نداریم.

۳.۳.۱. انواع وب اسکرپرها

وب اسکرپر ها با توجه به طراحیشان به انواع مختلفی تقسیم میشوند.

۱. وب اسکرپرهای خودساخته

وب اسکرپرهای خودساخته را با استفاده از فریمورک‌هایی مانند Scrapy و کتابخانه‌هایی مانند Beautiful Soup و Selenium و Requests می‌سازیم؛ که با استفاده از این فریمورک ها طراحی و پیاده سازی وب اسکرپر برای ما راحت میشود.

۲. وب اسکرپرهای پیش‌ساخته

اگر دانش ساخت و پیاده سازی وب اسکرپر را نداریم میتوانیم از وب اسکرپر های پیش ساخته استفاده کنیم.

برخی از این وب اسکرپرهای پیش ساخته امکاناتی مثل برنامه ریزی زمان وب اسکرپینگ و خروجی ها json یا اکسل را دارند که خود این وب اسکرپر های پیش ساخته به دو نوع افزونه های مرورگر و نرم افزار های کامپیوتر تقسیم میشوند.

۱. **افزونه‌های مرورگر**

برنامه‌هایی هستند که به مرورگرها، مانند کروم و فایرفاکس، اضافه می‌شوند. مزیت وب اسکرپرهای افزونه‌ای این است که ساده هستند و به آسانی می‌توانیم از آن‌ها استفاده کنیم.

۲. نرم افزارهای وب اسکریپینگ

در طرف دیگر، نرم افزارهای وب اسکریپینگ وجود دارند که می‌توانند دانلود شده و روی سیستم کامپیوتر ما نصب شوند. در حالی که استفاده از آن‌ها کمی سخت‌تر از افزونه‌ها است، اما آن‌ها به دلیل ویژگی‌های پیشرفته‌تری که دارند، مورد استفاده قرار می‌گیرند.

۴.۳.۱. آیا وب اسکریپینگ قانونی است؟

اگر شما یکسری داده و اطلاعات حقیقی که در دسترس عموم هستند را اسکریپ کنید، این کار قانونی است. ولی همواره شرایط استفاده وبسایت موردنظر را دنبال کنید و فایل robots.txt مربوط به آن را بخوانید.

۵.۳.۱. آیا اسکرپینگ و کراولینگ یکی هستند؟

اگرچه این دو بسیار شبیه هم به نظر می آیند، اما یکی نیستند. وب کراولینگ راهی است برای دریافت اطلاعات و سازماندهی کردن آن ها، در حالی که وب اسکریپینگ می تواند داده های خاص موردنظر ما را دریافت و برای استفاده های آتی، ذخیره کند.

۶.۳.۱. محافظت از وب سایت ها در برابر وب اسکرپینگ

به دلیل استفاده از الگوریتم های جدید در ربات های وب اسکرپی، اکثر مکانیزم های امنیتی قادر به شناسایی آن ها نیستند.

برای شناسایی ربات ها باید تمامی ترافیک ورودی و خروجی تجزیه و تحلیل شود. این تضمین می کند که تمام ترافیک ورودی و خروجی سایت شما، انسان است یا ربات.

عوامل زیر در برسی ترافیک موثر هستند:

HTML fingerprint : روند مشاهده ربات ها از هدرهای HTML آغاز می شود. این می تواند سرنخ هایی را در مورد ربات یا انسان بودن بازدید کننده برای ما بدهد.

IP reputation : جمع آوری اطلاعات از IP تمامی بازدید کنندگان وب سایت ما. از این طریق میتوان IP هایی را که تارخچه خوبی ندارند و قبلا هم از طرق آنها حملاتی را مشاهده کرده ایم را بشناسیم.

Behavior analysis : بررسی الگوهای رفتاری کاربران، مانند میزان درخواست های مشکوک و الگوهای بازدید غیر منطقی که به ما کمک می کند تا ربات ها را شناسایی کنیم.

Progressive challenges : استفاده از مجموعه ای از چالش ها، مانند پشتیبانی از کوکی ها و استفاده از جاوا اسکریپت، برای فیلتر کردن ربات ها. به عنوان آخرین راه حل، یک چالش CAPTCHA می تواند از ربات هایی که تلاش می کنند خودشان را به عنوان انسان جا بزنند، جلو گیری کند.

فصل دوم

۱.۲. بک – اند (Back – end)

۱.۱.۲. پایتون

پایتون یک زبان برنامه‌نویسی شیءگرا، تفسیری، سطح بالا، و همه منظوره است، که خیدو فان روسوم آن را طراحی کرده است، و اولین بار در سال ۱۹۹۱ منتشر شده‌است. فلسفه اصلی طراحی پایتون «خوانایی بالای کد» است و نویسه‌های فاصله خالی در آن معنادار هستند و مکرر استفاده می‌شوند. ساختار زبانی و دیدگاه شیءگرا در پایتون به گونه‌ای طراحی شده‌است که به برنامه‌نویس امکان نوشتن کد منطقی و واضح (بدون ابهام) را برای پروژه‌های کوچک و بزرگ می‌دهد.

کلمات کلیدی و اصلی این زبان به صورت حداقلی تهیه شده‌اند و در مقابل کتابخانه‌هایی که در اختیار کاربر است بسیار وسیع هستند.

بر خلاف برخی زبان‌های برنامه‌نویسی رایج دیگر که بلاک‌های کد در آکولاد تعریف می‌شوند (به‌ویژه زبان‌هایی که از نحو زبان سی پیروی می‌کنند) در زبان پایتون از نویسه فاصله و جلو بردن متن برنامه برای مشخص کردن بلاک‌های کد استفاده می‌شود. به این معنی که تعدادی یکسان از نویسه فاصله در ابتدای سطرهای هر بلاک قرار می‌گیرند و این تعداد در بلاک‌های کد درونی‌تر افزایش می‌یابد. بدین ترتیب بلاک‌های کد به صورت خودکار ظاهری مرتب دارند.

در پایتون مدل‌های مختلف برنامه‌نویسی (از جمله شی‌گرا و برنامه‌نویسی دستوری و تابع محور) را پشتیبانی می‌شود و برای مشخص کردن نوع متغیرها از یک سامانه پویا استفاده می‌شود.

این زبان از زبان‌های برنامه‌نویسی مفسر بوده و به صورت کامل یک زبان شیءگرا است که در ویژگی‌ها با زبان‌های تفسیری پرل، روبی، اسکیم، اسمال‌تاک و تی‌سی‌ال مشابهت دارد و از مدیریت خودکار حافظه استفاده می‌کند.

۲.۱.۲. کتابخانه های مورد نیاز برای وب اسکرپینگ

Requests

اولین پیش نیاز برای طراحی، ارتباط با اینترنت و دریافت اطلاعات از صفحات وب است در پایتون کتابخانه requsts این امکان را به ما میدهد که درخواست هایمان را به سمت سرور بفرستیم و با برسی نتیجه شروع به جستجو در المان های صفحات کنیم

Beautiful Soup

یک کتابخانه (Library) متن‌باز (Open-Source) پایتون است که برای وب اسکرپینگ فایل‌های HTML و XML طراحی شده‌است. Beautiful Soup بهترین تجزیه‌کننده (Parser) پایتون است، که به طور گسترده‌ای استفاده می‌شود. برای ساخت یک خزشگر شما نیاز به یک ابزار یا یک کتابخونه دارید که با استفاده از اون بتوانید المان های موجود داخل صفحات را به صورت مرتب شده و دسته بندی شده بخوانید و راحت تر بتونید المان های خودتان را پیدا کنید تا وقت کمتری از شما گرفته بشود.
کتابخانه**Beautifulsoup** این کمک را به شما میکند که داخل المان های موجود در صفحات وب جستجو کنید و راحتتر و سریعتر به نتیجه برسید.

Pymongo

یک توزیع پایتون است که شامل ابزار هایی برای کار با Mongodb است . در این پروژه از پایگاه داده Mongodb استفاده کرده ایم

Flask

Flask یک فریم ورک وب سبک می باشد که با زبان پایتون طراحی و نوشته شده است. البته به عبارت دقیق تر فلسک یک میکرو فریم ورک است چون بسیاری از ابزارها و کتابخانه های رایج سایر فریم ورک ها را ندارد. مثلا فلسک به طور پیش فرض نمی تواند با دیتابیس کار کند با فرم ها را اعتبارسنجی کند و برای این کار باید سراغ کتابخانه ها و دیتابیس های موجود بروید. فلسک به طور رسمی از سال ۲۰۱۰ تحت لایسنس BSD شروع به کار کرد و توسط آقای Armin Ronacher نوشته شده است.

Scrapy

Scrapy یک فریمورک اپن‌سورس و توسعه‌یافته با زبان برنامه‌نویسی پایتون است که به منظور اِسکرپ کردن صفحات وب و استخراج دیتا از آن‌ها در قالبی ساختاریافته به کار گرفته می‌شود که دیتای جمع‌آوری‌شده را برای اهداف مختلفی همچون داده کاوی، پردازش اطلاعات، یادگیری ماشینی و مواردی از این دست می‌توان مورد استفاده قرار داد

برترین ویژگی های این فریمورک

- قابلیت استخراج دیتا از صفحات وب به فرمت HTML و XML
- امکان دسترسی به دیتای مد نظر با استفاده ازXPath
- برخورداری از کنسول شِل تعاملی با امکان دیباگ کردن سورس‌کد به منظور مشاهدهٔ عملکرد ابزار اِسکرپینگ در جهت استخراج دیتای مربوط به تگ‌های مد نظر
- امکان استخراج دیتا به فرمت‌های مختلف از جمله JSON ، CSVوXML
- قابلیت ذخیرهٔ دیتای جمع‌آوری‌شده در فایل‌سیستم لوکال و...
- امکان اِنکُدینگ دیتای استخراج‌شده و برخورداری از قابلیت تشخیص خودکار نوع اِنکُدینگ فایل‌ها در جهت شناسایی دیتای آسیب‌دیده و غیر استاندارد
- برخورداری از قابلیت توسعه و به‌کارگیری یکسری API جهت بهبود عملکرد ابزار اِسکرپینگ
- برخورداری از یکسری به اصطلاح Middleware و Extention به منظور هندل کردن کوکی وب‌سایت‌ها برای ریکوئست‌های مختلف در هنگام اِسکرپینگ، ارسال ریکوئست‌های اچ‌تی‌تی‌پی و محدود کردن عمق کِراولینگ صفحات
- برخورداری از افزونهٔ Telnet Console که مشابه محیط شِل پایتون است با امکان هندل کردن تَسک‌هایی همچون کنترل وضعیت اِسکرپینگ وب‌سایت، توقف یا انجام مجدد فرآیند اِسکرپینگ
- امکان استفادهٔ مجدد از سورس‌کد اپلیکیشن به منظور کرول کردن صفحات از روی سایت‌مَپ یا سورس پیج مد نظر و کَش کردن فایل‌ها و تصاویر دانلودشده و جلوگیری از دانلود آن‌ها در اسکرپینگ‌های مجدد

۳.۱.۲. پایگاه داده

۱.۳.۱.۲. مفهوم پایگاه داده

پایگاه داده به ذخیره و دسته بندی اطلاعات گفته می شود به نحوی که دسترسی، مدیریت و به روزرسانی این اطلاعات به راحتی باشد.

۲.۳.۱.۲. انواع پایگاه داده ها

درپایگاه گذشته پایگاه داده ها به شکل سلسله مراتبی و یا شبکه ای وجود داشتند. اما امروزه پایگاه داده های رابطه ای (Relational)،SQL و NoSQL بیشتر مورد استفاده قرار می گیرند. شیوه دسترسی به اطلاعات موجود در این پایگاه داده ها بر اساس نوع آن ها متفاوت است

۳.۳.۱.۲. پایگاه داده NoSQL

NoSQL انواعی از پایگاه داده ها هستند که در سطحی وسیع تر از پایگاه داده های SQL کار می کنند و با مدل های مختلف داده ها مانند کلید – مقدار (Key-Value)، داده های گرافی، مبتنی بر مستند و غیره سر و کار دارند. این پایگاه داده ها با مجموعه های عظیمی از داده های توزیع شده کار می کنند و جایگزینی برای دیتابیس های رابطه ای هستند که داده را در جدول ذخیره می کردند.

۴.۳.۱.۲. Mongo db چیست

مونگو دیبی (**M**ongo **DB**) یکی از معروف ترین پایگاه داده های**No SQL** است که ساختار منعطفی دارد و بیشتر در پروژه هایی با حجم بالای داده استفاده می شود. این پایگاه داده پلتفرمی متن باز و رایگان است و با مدل داده های مستند گرا (**D**ocument - **O**riented) کار می کند و در ویندوز، مکینتاش و لینوکس قابل استفاده است. مقادیر داده ای ذخیره شده در مونگو دیبی، با دو کلید اولیه (Primary Key) و ثانویه (Secondary Key) مورد استفاده قرار می گیرند.

مونگو دیبی شامل مجموعه ای از مقادیر است. این مقادیر به صورت سندهایی (**D**ocument) هستند که با اندازه های مختلف، انواع مختلفی از داده ها را در خود جای داده اند. این مسئله باعث شده که مونگو دیبی بتواند داده هایی با ساختار پیچیده مانند داده های سلسله مراتبی و یا آرایه ای را در خود ذخیره کند

۱.۴.۳.۱.۲. ویژگی های استفاده از Mongodb

۱. مونگو دیبی به علت مستند گرا بودن مدل ذخیره داده ها در مقایسه با دیتابیس های رابطه ای بسیار منعطف تر و مقیاس پذیر تر است و بسیاری از نیازمندی های کسب و کارها را برطرف می کند.
۲. این پایگاه داده برای تقسیم داده ها و مدیریت بهتر سیستم از شاردینگ (**S**harding) استفاده می کند. شاردینگ به معنی تکه تکه کردن است و در لود بالای شبکه انجام می شود. به گونه ای که دیتابیس به چند زیربخش تقسیم می شود تا روند پاسخ دهی به درخواست هایی که از سمت سرور می آید، راحت تر شود.
۳. داده ها با دو کلید اولیه و ثانویه قابل دسترسی هستند و هر فیلدی قابلیت کلید شدن را دارد. این امر زمان دسترسی و پردازش داده را بسیار سریع می کند.
٤. همانند سازی (**R**eplication) یکی دیگر از خصوصیات مهم مونگو دیبی است. در این تکنیک از یک داده به عنوان داده اصلی کپی هایی تهیه شده و بخش های دیگری از سیستم پایگاه داده ذخیره می شود. در صورت از بین رفتن و یا مخدوش شدن این داده، داده های کپی شده به عنوان داده اصلی و جایگزین مورد استفاده قرار می گیرند.

۲.۴.۳.۱.۲. مزایا و معایب استفاده ازmongo db

دیتابیس های رابطه ای دارای اسکیما (**S**chema) هستند. یعنی ساختار خاصی برای داده هادر نظر گرفته و مدل های محدودی را ذخیره می کنند. اما مونگو دیبی و به طور کلی دیتابیس های **NoSQL**در برابر پذیرش داده هایی با توع مختلف بسیار منعطف هستند و این مزیت مهمی برای برنامه نویسان محسوب می شود. مقیاس پذیری این پایگاه داده باعث استفاده از آن در پروژه هایی می شود که با کلان داده ها (**B**ig **D**ata) سروکار دارند.

علاوه بر مزایای گفته شده مشکلاتی نیز در مونگو دیبی وجود دارد که ممکن است دردسرساز شود. این دیتابیس در استفاده از کلید خارجی (**F**oreign **K**ey) برای داده ها ضعف دارد و ممکن است پایداری داده ها و یکپارچگی سیستم را به هم بریزد. همچنین در خوشه بندی داده های موجود در این پایگاه داده، تنها می توان یک گره (**N**ode) را به عنوان گره اصلی (**M**aster) انتخاب کرد که اگر از بین برود، ممکن است مرتب سازی زیرگره های آن از بین برود. این مشکل در پایگاه داده کاساندرا (**C**assandra)برطرف شده است.

۴.۱.۲. جمع بندی

معمولا مونگو دیبی در سطوح بالاتر توسعه نرم افزار و برنامه نویسی استفاده می شود. این پایگاه داده بسیار قدرتمند است و ساختار ذخیره داده آن، باعث تمایز آن از سایر پایگاه داده ها شده است.
همچنین کوئری های ساده ای دارد و کار با آن، برای کسانی که تجربه حرفه ای در انجام پروژه های مختلف برنامه نویسی دارند کار چندان سختی نیست.

۲.۲. فرانت – اند (Front - End)

HTML

زبان نشانه‌گذاری ابرمتنی در کنار سی‌اس‌اس هسته فناوری ساخت صفحه‌های وب هستند. اچ‌تی‌ام‌ال زبان توصیف ساختار صفحه‌های وب است.زبانی‌است برای نشانه‌گذاری ابرمتن (فرامتن) که برای تدوین قالب و طراحی صفحه‌های وب به کار می‌برند. دستورالعمل‌های این زبان، برچسب (Tag) نام دارند که محتوای یک صفحه وب با آن‌ها، نشانه‌گذاری شده و بدین‌ترتیب، نحوه نمایش آن صفحه برای مرورگرهای وب، توصیف می‌شود. تاکنون ۵ نسخه از اچ‌تی‌ام‌ال عرضه شده‌است.

هر یک از برچسب‌های اچ‌تی‌ام‌ال، معنا و مفهوم خاصی دارند و تأثیر مشخصی بر محتوا می‌گذارند؛ مثلاً برچسب‌هایی برای تغییر شکل ظاهری متن، نظیر درشت و ضخیم کردن یک کلمه یا برقراری پیوند به صفحات دیگر در اچ‌تی‌ام‌ال تعریف شده‌اند.

یک سند اچ‌تی‌ام‌ال، یک پرونده‌ی مبتنی بر متن (Text–based) است که معمولاً با پسوند htm. یا html.نام‌گذاری شده و محتویات آن از برچسب‌های اچ‌تی‌ام‌ال تشکیل می‌شود. مرورگرهای وب، که قادر به درک و تفسیر برچسب‌های اچ‌تی‌ام‌ال هستند، تک‌تک آن‌ها را از داخل سند اچ‌تی‌ام‌ال خوانده و سپس محتوای آن صفحه را نمایان‌سازی (Render) می‌کنند.

اچ‌تی‌ام‌ال زبان برنامه‌نویسی نیست، بلکه زبانی برای نشانه‌گذاری ابرمتن است و اساساً برای ساخت‌مند کردن اطلاعات و جدایش اجزای منطقی یک نوشتار (نظیر عناوین، تصاویر، فهرست‌ها، بندها و جداول) به کار می‌رود. از سوی دیگر، اچ‌تی‌ام‌ال را نباید به عنوان زبانی برای صفحه‌آرایی یا نقاشی صفحات وب به کار بُرد؛ این وظیفه اکنون بر دوش فناوری‌های دیگری همچون سی‌اس‌اس است.

گفتنی است اچ‌تی‌ام‌ال شکلی از زبان دیگری به نام اس‌جی‌ام‌ال است و کنسرسیوم وب آن را به عنوان استانداردی برای نشانه‌گذاری مستندات ابرمتنی برای عرضه در وب، تدوین کرده‌است.

فصل سوم

۱.۳. پلتفرم و ابزار ها

برای پیاده سازی هر دو قسمت پروژه، خزشگرها و وب سایت، از زبان برنامه نویسی پایتون استفاده شده است. بخش های مربوط به خزش، عموما کد های شی گرا، ساده و مفهومی هستند. برای ارسال و دریافت درخواست های HTTP از پکیج requests استفاده شده است. همچنین برای سهولت در پردازش سورس صفحات از کتابخانه قدرتمند و محبوب BeautifulSoup استفاده شد.

۱.۱.۳. اشنایی با نحوه کار وب اسکرپینگ

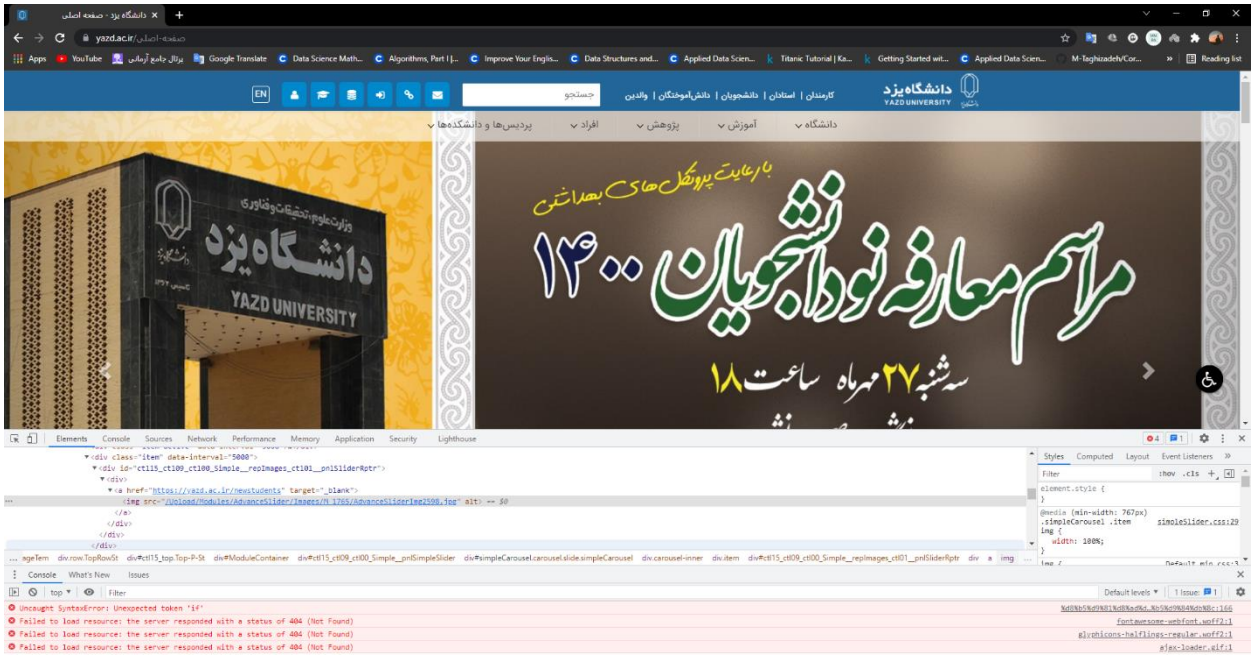
برای اینکه با سازوکار وب اسکرپینگ و نحوه عملکرد ان اشنا بشویم لازم است در ابتدا با عملکرد صفحات وب اشنا شویم

ابتدا مرورگر شما-URL ی که وارد کرده یا کلیک کرده‌اید را دریافت می‌کند و یک درخواست به سرور مربوطه ارسال می‌کند. سرور درخواست را پردازش کرده و یک پاسخ بازگشت می‌دهد.

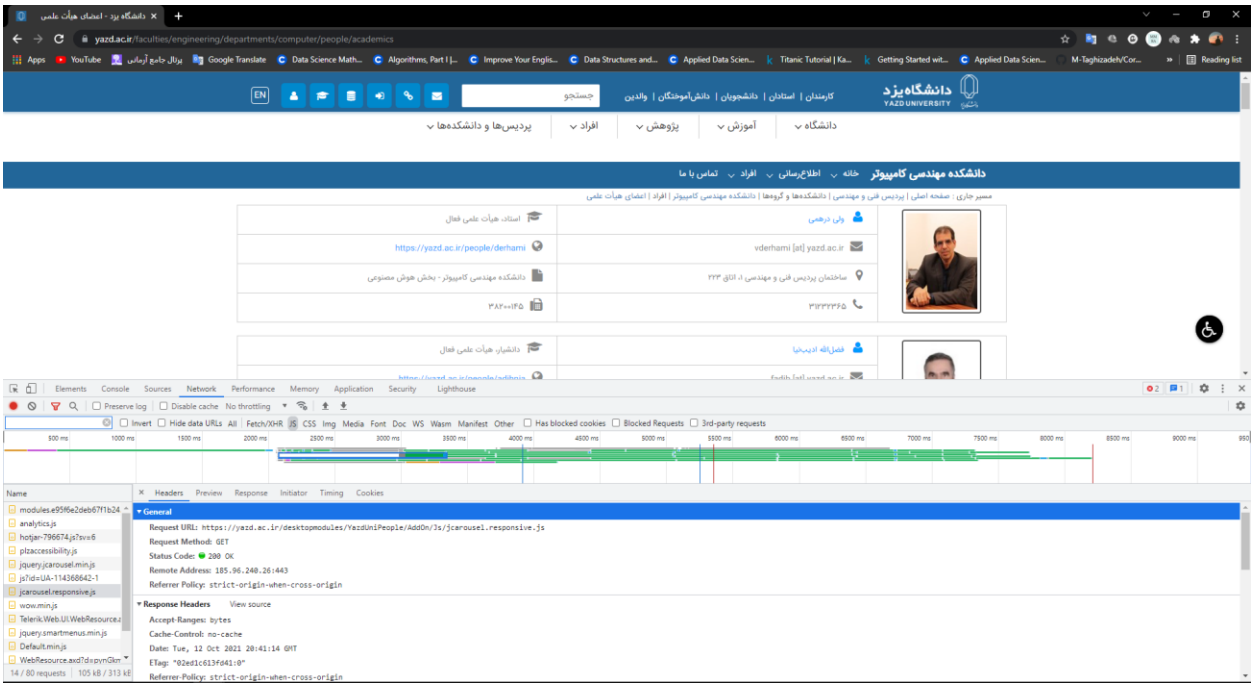
پاسخ سرور شامل کدHTML ، جاوا اسکریپت، CSS، JSONو داده‌های دیگر مورد نیاز است که به مرورگر امکان می‌دهد تا یک صفحه وب را برای بازدید شما ایجاد کند.

۱.۱.۱.۳. بررسی عناصر وب

در بعضی از مرورگر ها با زدن کلید f12 یا زدن کلیک راست و انتخاب گزینه Inspect پنجره شبیه تصویر زیر باز میشود.



یک فهرست از گزینه‌ها در نوار فوقانی پنجره مشاهده می‌شود. برگه‌ای که با ان کار داریم Network نام دارد. این برگه اطلاعاتی در مورد ترافیک HTTP به صورت زیر نمایش می‌دهد:



در بخش راست-پایین این برگه، اطلاعاتی در مورد درخواست HTTP مشاهده می‌شود. URL آن چیزی است که به دنبالش هستیم و method یک درخواست GET در HTTP است. «کد وضعیت» (Status Code) پاسخ

بازگشتی برابر با ۲۰۰ است که به این معنی است سرور درخواست را معتبر شمرده است. زیر کد وضعیت، آدرس ریموت را ملاحظه می‌کنید که آدرس IP عمومی سرور دانشگاه یزد است.

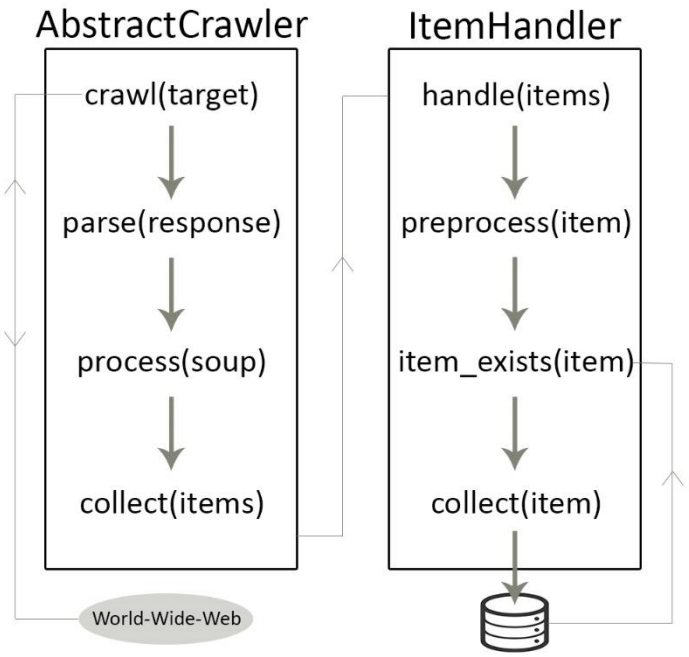
در لایه داده برنامه هم، پایگاه داده MongoDB، به جهت سادگی و انعطاف، انتخاب و استفاده شد. ارتباط و تعامل با این پایگاه داده با استفاده از پکیج pymongo صورت گرفته است.

از میکرو-فریم ورک Flask هم برای پیاده سازی یک سرویس تحت وب مینیمال، بر پایه دیتابیس جمع آوری شده، استفاده شد.

۲.۱.۳ خزشگر

برای پیاده سازی خزشگرها، ابتدا نیاز های پروژه بررسی، و ساختاری شی گرا طراحی و ارائه شد. فرآیند خزش، از خواندن صفحات شروع شده، و با استخراج و جمع آوری داده های هدف اتمام می یابد. در ارائه ساختار شی گرا خزشگر سعی شده در عین سادگی، از تکرار کد ها جلوگیری شود. دو کلاس Abstract، به عنوان کلاس پایه(Base Class) پیاده شده اند. کلاس AbstractCrawler و ItemHandler؛ در ادامه پیاده سازی، وابسته به نیاز ها و شرایط، کلاس های مشتق شده(Sub Class) گوناگونی بر اساس این دو کلاس پیاده سازی شده اند.

دو کلاس ارائه شده، با همکاری یکدیگر، یک ساختار منظم و هدفمند، برای پیاده سازی خزشگر ها ارائه می کنند. این بدین معناست که استفاده از این دو کلاس به عنوان بستر برنامه، یک Program Flow و یک Data Flow مشخص و پیوسته را باعث می شوند؛ چنانکه مراحل مختلف خزش، پردازش، و ذخیره سازی، بصورت ساماندهی شده و پیوسته برای هر کراولر، از پیش پیاده سازی شده و اجرا می شوند؛ و نیاز به تکرار حجم زیادی از کد نیست.



در ادامه با یک مثال، پروسه پیاده سازی یک نمونه خزشگر برای یک وب سایت مشخص فرضی را بررسی می کنیم. تصور کنید قصد داریم صفحات اساتید دانشگاه یزد را خزش کنیم. این صفحات عموما با یک قالب مشخص اساتید را نمایش می دهند. برای خزش این صفحات نیاز به یک کلاس مشتق شده از AbstractCrawler داریم. تنها متد Abstract و پیاده سازی نشده در کلاس AbstractCrawler، متد process می باشد؛ علت این موضوع، تفاوت فرآیند پردازش صفحات است. منطق مورد استفاده در پردازش صفحات سایت های مختلف، متفاوت اند. پس تنها متد تعریف نشده در کلاس پایه، متد مربوط به پردازش و استخراج اطلاعات، از سورس HTML می باشد.و اما در ساختار شی گرا برنامه، هر Object از Crawler، به یک Object از ItemHandler نیاز دارد. علت این موضوع در تصویر ارائه شده در بالا، کاملا مشهود است. ما به سادگی، هر کراولر را به یک ItemHandler مجهز و متصل می کنیم؛ تا پردازش های ثانویه روی داده های جمع آوری شده، و ذخیره سازی آنها، در جایی مستقل از خزشگر، و توسط فرآیندی ایزوله از آن انجام گیرد. کلاس EchoItemHandler، یک نمونه مفهومی، مینیمال و ساده از پیاده سازی ItemHandler است، که صرفا داده های جمع آوری شده را در کنسول پایتون چاپ می کند.

در ادامه، کلاس ها و کد های اصلی مربوط به قسمت خزش برنامه را به دقت بررسی خواهم کرد.

```
class AbstractCrawler(ABC)
```

همانطور پیش تر گفته شد، واضح است که این کلاس، اصلی ترین قسمت از بستر خزشگر ها را پوشش می دهد. در خط بالا شماهده می شود که برای پیاده سازی آن، از کلاسی به اسم ABC ارث بری شده است؛ کلاس ABC یک کلاس پیش ساخته یا از پیش تعریف شده در زبان پایتون است؛ که حروف آن مخفف عبارت Abstract Base Class می باشند. این کلاس ابزاری برای دستیابی به مفاهیم ارث بری و چندریختی می باشد.

```
def __init__(self, item_handler: ItemHandler):
    self.item_handler = item_handler
```

هر خزشگر، یک فرم مشتق شده (Sub Classed Object) از ItemHandler را به عنوان پارامتر ورودی سازنده اش می پذیرد و آن را به عنوان یک صفت نگهداری می کند. علت این موضوع پیش تر گفته شد.

```
# if you need to send a custom non-Raw GET request to a
target,
# ... just do fetch=False, and pass the response object
instead of url.
def crawl(self, target, fetch=True):
    if fetch:
        response = requests.get(target)
    else:
        response = target
    # performing the crawl event chain
    self.collect(
        self.process(
            self.parse(response)
        )
    )
```

در تصویر ارائه شده از ساختار کلاس ها که پیش تر ارائه شد، می توان دید که فرآیند خزش از این متد، و با فراخوانی ما، شروع می شود. پارامتر اول این متد، در حالتی که fetch=True باشد، یک URL است؛ و در حالتی که fetch=False باشد، یک سورس HTML می باشد. به عبارتی دیگر، fetch=False، مرحله ارسال

و دریافت درخواست HTTP را به شما واگذار می کند. در انتهای متد، فراخوانی تو در تو سه متد parse، process و collect را مشاهده می کنیم؛ که همان پروسه زنجیره ای خزش را شکل می دهد.

```
def parse(self, response):
    return BeautifulSoup(response.text, 'html.parser')
```

در اینجا سورس HTML به یک ساختار قابل پیمایش و جستجو BeautifulSoup یا اصطلاحا soup object تبدیل می شود که در ادامه برای استخراج اطلاعات از آن استفاده می کنیم.

```
@abstractmethod
def process(self, soup: BeautifulSoup):
    # should yield none, one, or multiple items.
    pass
```

متد process در بستر برنامه و در یک حالت کلی قابل تعریف نبوده است؛ علت آن را هم گفتیم؛ پردازش صفحات وابسته به وب سایت است. پس به یک تعریف Abstract و خالی اکتفا می کنیم. دکوریتور (Decorator) @abstractmethod به همین منظور استفاده شده است. این دکوریتور، به مفسر پایتون اجبار می کند، که کلاس های فرزند من، باید این متد را پیاده سازی کنند. اما نکته قابل توجهی در پیاده سازی این متد نهفته است؛ زیرا که ما از تعداد خروجی این متد مطلع نیستیم؛ نتیجه پردازش یک صفحه، می تواند هیچ، یک، یا چند آیتم مستقل باشند(برای مثال ۱۰ استاد). پس به سراغ Generator ها رفتیم. این ابزار در زبان پایتون، به همراه کلمه کلیدی yield، کمک می کنند تا هر زمان که در هنگام پردازش صفحه، آیتمی پیدا شد، آن را به متد collect منتقل کنیم.

```
def collect(self, items):
    # process() didn't yield any items:
    if items is None:
        return

    for item in items:
        self.item_handler.handle(item)
```

مقصد نهایی در کلاسCrawler . پارامتر items حاوی آیتم های yield شده از متد process است. منطق این متد هم بسیار واضح است. اگر آیتمی وجود داشت، به ItemHandler منتقل کن.

```
class ItemHandler(ABC):
```

در ادامه به بررسی کلاس ItemHandler می پردازیم. تعریف Abstract این کلاس، فاقد متد سازنده است.

```
def handle(self, item):
    self.preprocess_item(item)
    if self.item_exists(item):
        return
    self.collect_item(item)
```

مشابه کلاس AbstractCrawler، زنجیره متد های abstract این کلاس، در متد handle(مشابه متد crawl) فراخوانی می شوند. ابتدا متد preprocess_item فراخوانی می شود. سپس متد item_exists که یک متد boolean است، در قالب یک شرط، برای جلوگیری از ذخیره سازی داده های تکراری پیاده سازی شده است. در نهایت فرخوانی تابع collect که ذخیره سازی نهایی را عهده دار است، انجام شده است.

```
# in case any pre-processing is needed; e.g. setting static
fields
@abstractmethod
def preprocess_item(self, item):
    pass
```

همانطور که پیش تر گفته شد، این متد جایگاه مناسبی برای انجام پردازش های ثانویه بر روی داده ها، قبل از ذخیره سازی، فراهم می کند. برای مثال، در کلاس MongoDBHandler که نوعی ItemHandler است، ما در این متد، فیلد هایی ثابت در مقیاس صفحه را، به آیتم ها اضافه می کنیم؛ مثل فیلد دانشگاه و دانشکده.

```
# a boolean function, for avoiding duplicate data
@abstractmethod
def item_exists(self, item):
    return True
```

تعریفی abstract از متد item_exists؛ گفته شد که این متد مکانیزمی برای جلوگیری از ذخیره سازی داده های تکراری فراهم می کند. ماهیت این تابع boolean است و باید یکی از مقادیر True یا False را برگرداند. این متد در کلاس MongoDBHandler به صورت یک کوئری در دیتابیس Mongo برای بررسی وجود آیتم(استاد) پیاده سازی شده است.

```
# final part of the data-flow
# this is where the storing process happens (like passing
the item to database)
```

```
@abstractmethod
def collect_item(self, item):
    pass
```

اخرین مقصد در Data Flow برنامه

این متد هم همانند دو متد قبلی، abstract است، و بصورت ثانویه در Sub Class ها تعریف خواهد شد.

MongoDBHandler

```
class MongoDBHandler(ItemHandler):
```

پیش تر به این کلاس و نقش آن اشاره کردیم. در این قسمت از کد مشاهده می شود که ItemHandler به عنوان Base Class برای MongoDBHandler مورد استفاده قراره گرفته است. این فرآیند ارث بری، پیاده سازی اصولی فرم ساختاری کلاسItemHandler را حاصل شده است. به عبارتی دیگر، نوعی ItemHandler، بر روی بستر MongoDB، و بر اساس اصول ارائه شده، طراحی کردیم؛ که در ادامه از آن در کنار خزشگر های مان استفاده خواهیم کرد.

```
def __init__(self, mongo_uri="mongodb://localhost:27017/",
mongo_db="professors_db", mongo_collection="professors"):
    self.mongo_uri = mongo_uri
    self.mongo_db = mongo_db
    self.collection_name = mongo_collection
    self.client = pymongo.MongoClient(self.mongo_uri)
    self.db = self.client[self.mongo_db]
    self.static_fields = {}
```

برای ایجاد و نگه داری اتصال به پایگاه داده، در کلاسMongoDBHandler ، یک سازنده نوشته شد. این سازنده با استفاده از پارامتر های اتصال، یک Client پایگاه داده ایجاد کرده، و Database و Collection مورد استفاده را آماده سازی و در Attribute های کلاس ذخیره می کند. لازم به ذکر است که این فرآیند با استفاده از API پکیج pymongo انجام شده است. در آخرین خط از سازنده، یک دیکشنری خالی، در صفتی به نام static_fields ذخیره کردیم؛ این دیکشنری مربوط به ساختاری است که قبل تر درباره آن کمی صحبت شد. این دیکشنری به همراه متد preprocess_item، مکانیزمی را برای چسباندن زوج هایی از ("فیلد"، "مقدار") به آیتم‌ها(استاد ها) فراهم می کند. در ادامه جزئیات بیشتری حول این ساختار ارائه خواهد شد.

```
def goodbye(self):
    self.client.close()
```

این متد برای قطع اتصال به پایگاه داده در پایان برنامه نوشته شده است.

```
def preprocess_item(self, item: dict):
    item.update(self.static_fields)
```

در زمان تعریف static_fields، حول این متد صحبت شد. حال با یک مثال، کاربرد آن را با دقت بیشتری بررسی می کنیم. تصور کنید قصد داریم صفحه مربوط به اساتید دانشکده کامپیوتر دانشگاه یزد را خزش کنیم. با فرض اینکه، ما، پنج فیلد (نام، رشته، ایمیل، دانشکده و دانشگاه) را در پایگاه داده ذخیره می کنیم، فیلد های دانشکده و دانشگاه مقادیر ثابت ("دانشکده کامپیوتر" و "دانشگاه یزد") را دارا هستند. پس آن هارا به فرم زیر در دیکشنری static_feilds ذخیره می کنیم:

“Department”: “دانشکده کامپیوتر{“ ,”College”: “دانشگاه یزد}”

حال اگر یک آیتم(استاد) فرضی، در جریان خزش تا ذخیره سازی برنامه حرکت کند، با رسیدن به متد preprocess_item، دو فیلد تعریف شده در static_fields، به آن اضافه خواهد شد؛ مشهود است که این متد برای هر یک از آیتم های جمع آوری شده از صفحات انجام خواهد شد.

```
# accepts a dict too
# checks the existence based on Name
def item_exists(self, item):
    return self.db[self.collection_name].find({'Name':
item['Name'], 'College': item['College']}).count() > 0
```

درباره نقش متد item_exists، در قسمت تعریف abstract آن، صحبت شد. حال در اینجا، کوئری ای برای بررسی وجود آیتم(استاد) در پایگاه داده نوشته شده است.

```
# accepts a dict
def collect_item(self, item: dict):
    self.db[self.collection_name].insert_one(item)
```

درباره نقش متد collect_item، در قسمت تعریف abstract آن، صحبت شد. حال در اینجا، برای افزودن آیتم(استاد) به پایگاه داده، متد insert از Client پایگاه داده تحت pymongo را فراخوانی کردیم.

۳.۱.۳. پیاده سازی یک خزشگر کامل

حال می خواهیم بر روی بستر فراهم شده، یک خزشگر کامل برای وب سایت دانشگاه یزد را پیاده سازی کرده؛ و لیستی از استاید را در پایگاه داده جمع آوری کنیم.

```
class YazdCrawler(AbstractCrawler):
```

ارث بری از AbstractCrawler؛ متد process را در ادامه تعریف خواهیم کرد.

```
def process(self, soup: BeautifulSoup):
    for personDiv in soup.find(attrs={'class':
'YazdUniPeople'}).find_all(attrs={'class': 'row
personDiv'}):
        cols = personDiv.div.find_all('div',
recursive=False)[1].find_all('div', recursive=False)
        r_col = cols[1]
        l_col = cols[2]
        aoe, department =
self.extract_aoe(l_col.div.find_all('div',
recursive=False)[2].get_text())
        yield {
            'Name': r_col.div.div.a['title'],
            'eMail': r_col.div.find_all('div',
recursive=False)[1].get_text(),
            'AOE': aoe,
            'Department': department
        }
```

درباره نقش و چهارچوب پیاده سازی این متد در بخش معرفی AbstractCrawler، به صورت مفصل صحبت شد. در اینجا، ابتدا المنت های HTML حاوی اطلاعات اساتید، به دقت بررسی و شناسایی شد. سپس با استفاده از توانایی BeautifulSoup، فیلد های موردنظر استخراج می شوند؛ و در نهایت یک دیکشنری به ازای هر

استاد حاوی پنج فیلد، yield می شود. این دیکشنری ها پس از yield شدن به عنوان آیتم؛ به ItemHandler منتقل می شوند.

```
def extract_aoe(self, aoe_str):
    aoe_split = aoe_str.split(" - ")
    if len(aoe_split) > 1:
        return aoe_split[1], aoe_split[0]
    else:
        return None, aoe_split[0]
```

این متد یک متد فرعی، برای پردازش بخشی از متن استخراج شده از المان های HTML است؛ علت جداسازی این متد از متد process، یک تصمیم سلیقه ای بوده است.

۴.۱.۳. به کارگیری خزشگر دانشگاه یزد

برای استفاده از کلاس YazdCrawler، به یک instance از هر یک از Sub Class های ItemHandler، و یک instance از YazdCrawler نیاز داریم.

```
mongo_handler = MongoDBHandler()
mongo_handler.static_fields['College'] = "یزد دانشگاه"
yazd = YazdCrawler(mongo_handler)
```

به تعریف static_fields توجه کنید.

حال کافیست متد crawl را به همراه یک url استفاده کنیم:

```
yazd.crawl("https://yazd.ac.ir/faculties/science/departments
/physics/people/academics")
```

پس از خزش مجموعا ۱۳ صفحه از دانشگاه یزد، که تقریبا ۲۴ ثانیه به طول انجامید، اطلاعات ۲۲۷ استاد در پایگاه داده جمع آوری شد.

۵.۱.۳. پیاده سازی وب سایت

حال می خواهیم، پایگاه داده ای که از خزشگر حاصل شد را به کار گرفته، و وب سایتی به روی آن بسازیم.یک طرح حداقلی، به صورت یک وب گاه تک صفحه ای، شامل یک فرم جستجو و قسمتی برای نمایش نتایج در نظر گرفته شد. برای Front-end این وب گاه، یک Template با استفاده از HTML و CSS طراحی شد؛ که در

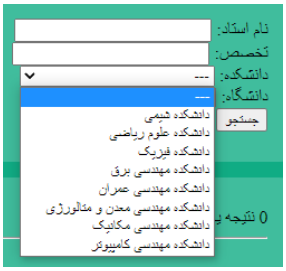
مسیر web/templates/index.html قرار گرفته است. Back-end این وب گاه نیز، با استفاده از Flask و در قالب یک route و یک متد خلاصه، در فایل pdb.py نوشته شد. در ادامه کد آن را بررسی خواهیم کرد.

```
mongo_client =
pymongo.MongoClient("mongodb://localhost:27017/")
mongo_db = mongo_client['professors_db']
mongo_collection = mongo_db['professors']
```

برای پیاده سازی Back-end وب سایت، اتصال به پایگاه داده، اصلی ترین پیش نیاز موجود است. در ابتدای برنامه، Client پایگاه داده راه ایجاد و آماده کردیم.

```
# distinct values for department and college
departments = mongo_collection.distinct('Department')
colleges = mongo_collection.distinct('College')
```

در اینجا، تمام مقادیر یکتا یا distinct موجود در فیلد Department و College را از پایگاه داده استخراج کردیم، این کار با هدف ارائه ساختاری برای محدود کردن نتایج جستجو توسط کاربر انجام شده است. مقادیر یکتای هر کدام از فیلد ها، در سمت رابط کاربری گرافیکی صفحه، در المان های Select قرار می گیرند؛ که به کاربر توانایی انتخاب می دهد.




```
@app.route('/', methods=["GET", "POST"])
def index():
```

همانگونه که قبل تر گفته شد، وب سایت ما فقط دارای یک Route یا مسیر می باشد. این مسیر، دو متد POST و GET را می پذیرد. ما از متد GET برای نمایش صفحه بدون انجام فرآیند جستجو، و از متد POST برای انتقال پارامتر های جستجو به سرور استفاده می کنیم.

```
if request.method == "GET":
    professors = []
```

در اینجا با بررسی متد Request وارد شده به سرور، در صورت برابری با GET، متغیر professors را با یک لیست تهی، مقدار دهی می کنیم. این متغیر(professors)، بعدا در مرحله render صفحه، به Template تحت HTML منتقل خواهد شد.

```
match = {}
if request.form['department'] != "any":
    match['Department'] = request.form['department']
```

```
if request.form['college'] != "any":
    match['College'] = request.form['college']
if request.form['name']:
    match['Name'] = {'$regex': request.form['name']}
if request.form['aoe']:
    match['AOE'] = {'$regex': request.form['aoe']}
search_results = mongo_collection.aggregate([
    {
        "$match": match
    }
])
```

در این قسمت از کد، که اصلی ترین پروسه موجود در Back-end محسوب می شود، پارامتر های مختلف جستجو، که از سمت کاربر ارسال شده اند را می خوانیم، و مقادیر آن ها را در کوئری های مجزایی قرار می دهیم؛ نهایتا چند کوئری با استفاده از عملگر aggregate ترکیب می شوند و نتیجه آن ها از پایگاه داده خوانده می شود.

```
professors = list(search_results)
```

پس از فراهم شدن نتایج جستجو، رکورد های به دست آمده را به لیست Cast می کنیم و آن را در متغیر professors قرار می دهیم. در ادامه، و در قسمت نهایی کد، این متغیر، و دیگر متغیر های فراهم شده برای Front-end از طریق تابع render_template، به سمت Client منتقل خواهند شد.

```
return render_template('index.html', professors=professors,
departments=departments, colleges=colleges)
```

۵.۱.۳. نتیجه گیری

راه مختلفی برای تسریع کارها و مسائل روزمره وجود دارد . در این پروژه سعی شده است بستری فراهم شود تا متقاضیان تحصیلات تکمیلی و دانشجویانی که قصد مهاجرت دارند به سرعت اساتیدی که حوزه مرتبط با انها را دارند پیدا کنند و برای مکاتبه و تبادل اطلاعات با هم ارتباط گیرند

مراجع

[۱] https://blog.faradars.org/what-is-web-scraping/#%DA%86%D8%B1%D8%A7_%D9%88%D8%A8_%D8%A7%D8%B3%DA%A9%D8%B1%D9%BE%DB%8C%D9%86%DA%AF_%D9%85%D9%81%DB%8C%D8%AF_%D8%A7%D8%B3%D8%AA%D8%9F

[۲] https://7learn.com/blog/what-is-mongo-db

[۳] https://kaliboys.com/web-scraping/

[۴] https://pymongo.readthedocs.io/en/stable/

[۵] https://virgool.io/@o.h.hadidy/%D9%86%D9%82%D8%B7%D9%87-%D8%B5%D9%81%D8%B1-%D8%B7%D8%B1%D8%A7%D8%AD%DB%8C-%D8%AE%D8%B2%D8%B4%DA%AF%D8%B1-%D9%88%D8%A8-%D8%A8%D8%A7-%D9%BE%D8%A7%DB%8C%D8%AA%D9%88%D9%86-aqncevxjkjyz