




6 DE ENERO DE 2023

Developing a Scraper that publishes data through an API

DESARROLLO DE APLICACIONES PARA CIENCIA DE DATOS

GRADO EN CIENCIA E INGENIERÍA DE DATOS, 2º

Escuela de Ingeniería Informática, Universidad de Las Palmas de Gran Canaria



Resumen

Este trabajo consiste en usar los conocimientos adquiridos en clase relacionados con las API REST y con el conocimiento relacionado con el scrapping de información de una página web adquirido en la asignatura de “*Fundamentos del Marketing y Comportamiento del Consumidor*”. El objetivo de este proyecto era ser capaces de responder a las peticiones que nuestros clientes realizaran desde la API, y poder proporcionarle la información que desearan, tal como la localización, servicios del hotel, valoraciones, comentarios... Desde este proyecto se ha intentado que el código sea lo más legible y entendible posible, sin recaer en ningún momento en el uso de comentarios para la explicación de este. Ahondando en la metodología, de forma muy resumida, el programa adopta un comportamiento lazy, ya que cuando arranca no inicializa datos de hoteles, sino que los va tomando a medida que se lo van pidiendo. De igual manera, cuando se nos pide, por ejemplo, los servicios de un hotel, el programa toma todos los datos del hotel, ya que basándonos en la “localidad temporal”, que establece que si nos piden las reviews de un hotel, es muy probable que en un futuro nos pida otra información relacionada con ese hotel.

La aplicación no se puede decir que sea rápida, ni eficiente, ni que esté al 100% optimizada, aunque he intentado que, con mis conocimientos actuales, ésta no incurra en acciones innecesarias o tenga código zombie; por ende, se puede decir que, aunque no será la más rápida, sí se puede decir que realiza bien su función.

Índice

Resumen.....	2
Recursos Utilizados	2
Diseño.....	3
Conclusiones	5
Líneas Futuras	5
Bibliografía	6

Recursos Utilizados

Para la realización de este proyecto se ha utilizado el entorno de programación “IntelliJ Idea Community Edition 2022.2.3”, así como las dependencias de gson para serializar la información y devolverla por la API; jsoup para hacer el scrapping de Booking; y spark-core para la implementación de la API. Para el control de versiones se ha usado Git, y para la realización de esta memoria se ha utilizado “Microsoft Word”.

Diseño

En el apartado de diseño, se explicarán las distintas clases existentes en el proyecto, así como su organización y utilidad, y a su vez se nombrarán las interfaces implementadas. Finalmente, explicaré de forma más específica el funcionamiento de la aplicación.

En primer lugar, el modelo se compone de las siguientes clases:

- Review: que contiene los atributos de nota, comentario positivo, y comentario negativo.
- HotelInfo: que contiene información del hotel como la url de Booking, el nombre, la localización, la nota general y la acomodación (Hotel, Apartamento, Bungalows...).
- Hotel: que contiene una instancia de HotelInfo, a parte de el nombre que le da Booking (Ej: bungalows-cordial-biarritz) de forma que sea más fácil buscar para futuras peticiones, un id, un mapa de las categorías del hotel (clave categoría, valor nota), un mapa de servicios (clave servicio principal, valor lista de subservicios) y una lista de Review.
- Resource: tiene dos atributos: nombre del hotel y localización del recurso en la API (url).

A continuación, se procederá a explicar las distintas clases funcionales que contiene el proyecto:

- Interfaz HotelScrapper: que establece los métodos que debe tener cualquier scrapper, independientemente de la implementación, tales como scrapName(), scrapLocation(), scrapMark(), scrapAccommodation(), scrapCategories(), scrapCategoriesMark(), scrapServices() y scrapReviews()
- BookingHotelScrapper: implementación de la interfaz definida en el anterior punto que tiene dos atributos (ulr de la página a scrappear y url de la página de reviews); sin embargo, al crear una instancia de esta clase, se le pasa el nombre que le da booking a ese hotel (Ej: bungalows-cordial-biarritz), y el constructor se encarga de crear las conexiones a ambas páginas. Aparte, esta clase tiene un método estático que parsea una String por letras mayúsculas mediante una expresión regular (`\\s(?=[A-Z])`), de gran utilidad para obtener los servicios.
- SparkWebService: clase que implementa la API y que da soporte a las siguientes peticiones: /v1/hotels que devuelve una lista de Resource, que muestra los hoteles que actualmente tenemos cargados en memoria; /v1/hotels/:name que proporciona información sobre el hotel pasado por parámetro; v1/hotels/:name/services; v1/hotels/:name/comments y v1/hotels/:name/reviews. A esta clase se le pasa por constructor una instancia de controlador, de forma que pueda comunicarse con él y pedirle la información necesaria.
- Controller: el controller tiene dos atributos, una lista de recursos y una lista de hoteles, que va manejando de forma paralela (si se añade un hotel a hoteles, también se añade a resources). Cada vez que la API le pide algo al controlador, lo primero que hace es mirar si ya lo tiene (que vendría a ser mirar en la lista de hoteles), y si no lo tiene, ejecuta la función scrap, que acepta como parámetro el nombre del hotel, y se encarga de crear la instancia de BookingHotelScrapper y conseguir la información de un hotel. Si el hotel no existe (es decir, a la hora de scrapear el nombre da null), se lanza la excepción (creada por mi y que hereda a Exception) HotelNotFoundException, que posteriormente la API atraparé y como consecuencia, hará halt de un 400.

Si es la primera vez que se pide información sobre el hotel, tardará unos segundos, pero la segunda vez será prácticamente automático, ya que solo será buscar y tardará pocas décimas.

Conclusiones

Igual que en otros proyectos, también me gustaría aprovechar esta ocasión para contar mi experiencia personal. Antes de nada, me gustaría decir que este proyecto en específico me ha resultado un poco pesado, pero no por el proyecto en sí (que me parece bastante interesante), sino por el hecho de que hicimos uno igual hace dos meses para la asignatura de FMCC. Y como dicho trabajo lo hice con Selenium y Python, hacerlo en este proyecto con Jsoup y Java se ha hecho un poco pesado porque, aunque puedan parecer muy similares, hay ciertas diferencias que realmente fueron un dolor de cabeza, y sentí como tuve que hacer todo dos veces y se me hizo un poco cansado y repetitivo. Sin ninguna duda, la mayor dificultad que tuve fue a la hora de implementar la clase `BookingHotelScraper` (obviamente ya que es la clase grande del proyecto) ya que había ciertas cosas que te hacían odiar a Booking (por ejemplo, había ciertas cosas que, aunque solo fueran mostradas una vez en la página web, en el html estaban repetidas, y tenías que seleccionar ciertas etiquetas para que a la hora de scrappear no te saliera repetido). Pero quitando esta parte, la implementación de la API fue bastante sencilla ya que habíamos hecho algunos ejemplos en clase, aunque tuve un debate interno sobre cómo debía la API pedirle la información al controlador. Finalmente, la mejor opción que encontré fue pasarle a la API una instancia del controlador, de forma que éste tuviese ciertos métodos para que la API pudiese pedirle la información; aunque eso supusiese que el controlador no pudiera crear la instancia de la API. Reconozco que esta última parte (y seguramente alguna otra parte) se podría haber hecho de otra manera, pero fue la única que se me ocurrió que fuese viable y que supiese implementar.

Pero, al margen de eso, el proyecto es muy interesante y agradezco que se haya hecho con la finalidad de subir nota (lo cual me viene bastante bien).

Líneas Futuras

Si miramos a las líneas futuras del proyecto, aunque actualmente la función que realiza puede ser de utilidad para una persona de a pie interesada en la información de un hotel, hay que tener en cuenta que toda la información la sacamos de booking, por lo que nuestra aplicación actual sería como Booking 2.0 pero sin la posibilidad de reservar; aunque esto se puede ver como una ventaja: en nuestra aplicación puedes ver la información de un hotel sin la necesidad de introducir fechas, ni número de personas... Ahorrando pasos innecesarios para gente que quiere acceder a la información que ofrece booking sobre un hotel (que sinceramente es bastante completa), pero que no busca hacer una reserva.

Ahora sí, mirando al futuro, implementar la posibilidad de reservar, o de calcular precios no sería muy útil ya que, en este caso, sí que sería una copia idéntica a Booking y no tendríamos ninguna ventaja competitiva sobre ellos. Sin embargo, una cosa que me parece muy interesante es una especie de Análisis Porter, pero para hoteles.

Un Análisis Porter es según Wikipedia una articulación de las 5 fuerzas que determinan la intensidad de competencia y rivalidad en una industria, y, por lo tanto, cuán atractiva es esta industria en cuanto a oportunidades de inversión y rentabilidad. Es decir, cada fuerza tiene ciertas variables a estudiar, donde se establece una importancia y su atractivo (adjunto imagen de un trabajo que hicimos el año pasado en La Empresa y Sus Procesos sobre el análisis Porter).

<i>5. Amenaza de entrantes potenciales</i>			
		(1=atractivo muy bajo; 3= atractivo medio; 5= atractivo muy alto)	
VARIABLES	IMPORTANCIA (%)	PERFIL DE ATRACTIVO (1-5)	ATRACTIVO
Existencia de economías de escala	15	3	45
Identidad de marca	40	4	160
Costes de cambio	Elegir un valor	Elegir un valor	0
Requerimientos de material	Elegir un valor	Elegir un valor	0
Acceso a canales	Elegir un valor	Elegir un valor	0
Ventajas en costes	Elegir un valor	Elegir un valor	0
Know-how	Elegir un valor	Elegir un valor	0
Curva de aprendizaje	25	3	75
Acceso a recursos	Elegir un valor	Elegir un valor	0
Políticas de proteccionismo	Elegir un valor	Elegir un valor	0
Barreras administrativas y legales	Elegir un valor	Elegir un valor	0
Inversión necesaria	20	5	100
Respuesta de empresas establecidas	Elegir un valor	Elegir un valor	0
IMPORTANCIA TOTAL: 100		ATRACTIVO TOTAL: 380	

En el caso de nuestra aplicación, le pediríamos al cliente tres cosas: las variables que quiere estudiar (precio económico, buena limpieza, calidad del wifi...), que establezca la importancia de dichas variables, y los hoteles de los que quiere realizar la comparación. Luego, internamente, nos encargaremos de hacer un scrapping de los distintos hoteles y calcularemos el atractivo de cada variable para cada hotel, y el atractivo total de cada hotel. De esta forma, el usuario podrá ver cuál es el mejor hotel para elegir basado en sus preferencias.

Siendo sinceros, esta idea me parece buena, ya que existen muchos comparadores de precios, lugares donde puedas ordenar por notas... Pero en ninguno que yo conozca puedes establecer un orden de prioridad o ir tan lejos. Además, pienso que por esta misma razón, estaríamos resolviendo uno de los principales rompecabezas de las familias (elegir un hotel para ir de vacaciones).

Y ya con esto llegamos al fin de la memoria, esperando que no se le haya hecho extremadamente larga y agrediéndole su tiempo.

Bibliografía

https://es.wikipedia.org/wiki/Cercan%C3%ADa_de_referencias

<https://chat.openai.com/chat> (para dudas, no ha generado código)

Datos que se pueden extraer de la página web, Otavio Roncal; José Juan Hernández, 2022

<https://www.booking.com>