SCRAPPER 2022-2023

13 enero

Universidad de Las Palmas de Gran Canaria

Escuela de Ingeniería Informática

Grado en Ciencia e Ingeniería de Datos

ULPGC
Universidad de
Las Palmas de
Gran Canaria

Autor:

Óscar Rico Rodríguez

Versiones:

0.0 [15/12/2022]

Creación de la interfaz HotelScrapper, implementación en la clase HotelBookingScrapper y creación de las clases POJO (Comment, Location, Rating y Service).

1.0 [20/12/2023]

Sustitución de clases POJO por records y creación de la clase WebService

2.0 [03/01/2023]

Creación de una interfaz para los WebService e implementación en la clase SparkWebService (antes WebService)

3.0 [09/01/2023] FINAL

Mejoras en las llamadas API e unificación de los getter en un mismo método implementando un path común y una excepción si la url es desconocida.

Resumen

Scrapper

Para este proyecto se nos pedía realizar un programa que realizara un scrapping sobre una página web dedicada a la oferta de estancias vacacionales y devolver sus datos mediante un webservice. Para el desarrollo de mi programa he decidido centrarme en la página web de booking y para el webservice he realizado una API REST mediante Java Spark.

Contenido

Mi programa consta de tres clases, dos interfaces y cuatro records. Divididos en tres paquetes y un subpaquete :

- runnable. Contiene la clase Main desde la que se inicializa la programa.
- scrapper. Contiene la clase HotelBookingScrapper y la interfaz HotelScrapper.
    - records. Contiene cuatro records Comment, Location, Rating y Service.
- api. Contiene la clase SparkWebService y la interfaz WebService.

Funcionamiento

Al ejecutar el programa la clase main inicializa la clase SparkWebService.

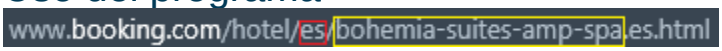
SparkWebService se encarga de poner en funcionamiento las llamadas api que los usuarios usaran para interactuar con la aplicación e inicializa la clase HotelBookingScrapper que será la encargada de recolectar los datos del hotel insertado por el usuario y dar respuesta a sus peticiones. Una vez recibidos los datos, SparkWebService los pasará a formato JSON (ya que se trata de una API REST) y se los mostrará al cliente en el cuerpo de la petición siempre y cuando los datos introducidos sean correctos.

Anotaciones

Para el desarrollo de esta aplicación he tenido en cuenta los siguientes aspectos:

He preferido centrarme en booking, ya que las distintas páginas usan sus propios formatos para identificar los hoteles, y el de booking me pareció que se asemejaba más al lenguaje natural (sus identificados son muy parecidos al nombre del hotel y no difusos como *82710ab2* para referirse a un hotel). Además el hecho de implementar varias páginas no permite que la búsqueda sea al cien por cien libre sin la necesidad de tener una base de datos que guarde los identificadores de los diversos hoteles en cada una de la página, y para mi gusto la libertad a la hora de poder buscar cualquier hotel sin necesidad de que tenga que estar en una base de datos de la propia aplicación (lo que limitaría el número de hoteles de los que sacar información con respecto al número de hoteles en booking y otras páginas) tiene más valor que poder sacar información de un hotel en diversas páginas pero el número de hoteles sea limitado.

Uso del programa



Para usar el programa es necesario usar como identificador el nombre del hotel en booking (recuadro amarillo) e indicar en que país se encuentra (recuadro rojo). Esto último es para permitir que se puedan scrapper hoteles de cualquier país encontrados en booking.

Ejemplos de busquedas:

http://localhost:4567/es/hotels/bohemia-suites-amp-spa (Idéntico para localización, servicios y ratings)

Para los comentarios decidí añadir un query param (page) que te permite avanzar por las páginas de comentarios. Se muestran 25 comentarios por cada página (siempre y cuando haya). En caso de no haber devuelve una lista vacía.

http://localhost:4567/es/hotels/bohemia-suites-amp-spa/comments?page=1

Índice

Recursos utilizados

Entorno de desarrollo

- IntelliJ

Controlador de Versiones

- Git

Editor de texto

- Word

Alojamiento del Proyecto

- GitHub

Diseño

Patrón y principio de diseño usado

Como patrón de diseño para mi proyecto decidí usar el Movel View Controller (MVC) ya que es un patrón sencillo, escalable y mantenible. Por lo que no lo vi mala opción para un primer proyecto.

El MVC es un modelo divido en 3 capas donde cada cual cumple una función espéficica.

- View. Parte de la aplicación donde se contempla la interfaz gráfica. En esta capa se obtiene información sobre lo que quiere el usuario (Eventos) y se muestra la información del Modelo.
- Model (Lógica de negocio). Parte de la aplicación donde se relacionan los datos con los que se van a trabajar. Es el encargado de mapear las actividades del mundo real a la forma en la que se va a modificar la información.
- Controller. Parte de la aplicación que responde a los eventos del usuario y decide que view mostrar al usuario dependiendo de la solicitud recibida (Se puede ver como la conexión entre el model y el view).

En cuanto al principio de diseño, he seguido la guía de Clean Code que se nos proporcionó en el Campus Virtual.

Clean Code es una filosofía de desarrollo de software que consiste en aplicar técnicas simples que facilitan la escritura y lectura de un código.

Algunas de las principales prácticas para tener un código limpio son las siguientes:

1. Los nombres son importantes
    - Debe ser preciso, sin preocupación sobre el tamaño del nombre, es preferible claridad antes que brevedad.
2. Regla del boy scout
    - Esta regla indica que el código debe quedar más limpio de lo que estaba antes de ser editado
3. Métodos breves y descriptivos
    - Debemos evitar el anidamiento excesivo (complejidad ciclomática). Esto se consigue asignando a cada método una única funición.
4. DRY (Don´t Repeat Yourself)
    - Evitar el código duplicado. El código repetido entre diferentes clases, se pasa a clases abstractas.
5. Comentar solamente lo necesario
    - El código bien escrito se explica solo, sin necesidad de comentarios.
6. Tratamiento de errores
    - Para un código limpio, tratar las excepciones de forma correcta es un gran paso.
7. Tests limpios
    - Realizar tests es una etapa muy importante en la programación, un código solo se considera limpio después de haber sido válidada con varias pruebas.

Diagrama de clases

Imagen en el repositorio [class_diagram.png]

Conclusiones

Durante el desarrollo de esta actividad he afianzado conocimientos con la creación de servicios web siguiendo el modelo de una API rest, con la utilización de los getters, paths, parametros tanto de query como en el propio path y el lanzamiento de excepciones de tipo halt. Además, he ampliado conocimientos sobre Java Spark, ya que fue el modulo que use para el manejo del webservice y también sobre Jsoup, que fue el modulo que use para realizar el scrapping a booking.

En cuanto a experiencias en el proyecto me ha gustado bastante, ya que gracias a él he aprendido cosas sobre como trabajar con html y como funcionan los records en java. Por otro lado, si empezara de nuevo el proyecto, comenzaría con el uso de Git desde el inicio del proyecto, para tener un control de las versiones y evitar problemas como pérdidas de código.

Líneas Futuras

Bajo mi punto de vista si, quisiéramos evolucionar este producto hasta convertirlo en un producto comercializable, debería tener en cuenta los siguientes aspectos:

- Posibilidad de buscar cualquier hotel en diversas páginas con un mismo identificador.
- Posibilidad de scrappear hoteles por zonas.

Bibliografía

MVC

¿Qué es patron MVC y cómo funciona?

MVC Explained in 4 Minutos

Clean Code

Clean Code: Código limpio, ¿qué es?

Guía de estilo en Java y buenas prácticas para un código limpio

Java

Spark Documentation

Selector-syntax jsoup