Documentación

21/10/2021

Alfonso González Pascual
2º DAM SAFA REYES

Metodología

Tras mucha investigación decidí utilizar la librería Scrapy de Python para realizar el web scraping, ya que es una de las librerías más completas y actualizadas para realizar esta tarea con Python. Por otra parte decidí almacenar los datos en una base de datos SQLite3 ya que las librerías para manejarla vienen integrados con el propio Python y tienen una fácil utilización, de todas formas he sacado parte del resultado en un archivo tipo Json que adjunto con el proyecto, ya que bases de datos no relacionales para las que almacenar la información en archivos Json es mucho mejor.

Clases

Primero creé varias spiders dentro de la carpeta Spiders, con las cuales fuí iterando de manera modular el desarrollo del proyecto, añado estas clases al proyecto en el GitHub por fueran de su interés, una vez que todas estas clases funcionaron de manera independiente procedí a su unificación el la spider anunciosmulti_spider.py. Por otra parte en el archivo Settings.py habilité la posibilidad de utilizar un User_agent distinto, poniendo en mi caso el compatible con Mozilla.

```
USER_AGENT = 'Mozilla/5.0 (compatible; Googlebot/2.1;
+http://www.google.com/bot.html)'
```

Por otra parte dentro de este mismo archivo hice que mi programa no obedeciera las directivas de robots.txt.

```
ROBOTSTXT_OBEY = False
```

En el archivo de Items.py declaré los tres items que me ayudarían a mandar la información a mí base de datos, y finalmente en pipelines.py configuré la base de datos y la entrada y salida de información de la misma.

Dificultades

I. Conexión con la página

La primera dificultad que tuve con el proyecto fué a la hora de conectarme con la página de mil anuncios, ya que el archivo robots.txt de esta página prohibía el scrapping en la misma, para solucionar esto hice lo ya mencionado de ignorar este archivo.

Por otra parte me di cuenta de que la página estaba detectando que le estaba haciendo scrapping de todas formas, y me cortaba la conexión tras haber scrapeado dos o tres entradas, por esto cambié mi user agent al ya mencionado y dejé de tener problemas

II. Campos dinámicos

Tuve problemas tanto a la hora de acceder a la información del botón llamar como para cambiar a la siguiente página con el botón, ya que ambos dependían de una función de JS y no de un atributo href.

Para solucionar el caso del botón llamada busqué dentro del inspector la llamada que hacía el JS al servidor y lo que le devolvía, encontrando así que le enviaba un archivo json y solicitando yo este archivo desde el mismo código de la spider.

Para solucionar el problema del cambio de página se me ocurrió buscar la solicitud de JS en el inspector y enviar el token a la página mediante un request, pero fuí incapaz de encontrarlo, por lo que hice una solución hardcodeada que me recorre las 200 primeras páginas.