Web scraping

Introducción

El web scraping (o raspado web) es una técnica utilizada para extraer información de sitios web. Consiste en utilizar programas o scripts para automatizar la navegación por un sitio web y recolectar información de manera automatizada. Los datos recolectados pueden ser utilizados para diversos fines, como la monitorización de precios, la recopilación de datos para análisis, la generación de contenido automatizado, entre otros. Algunos ejemplos de uso del web scraping incluyen el análisis de tendencias en redes sociales, la extracción de datos de sitios de comparación de precios, la recopilación de noticias y actualizaciones de precios en sitios de comercio electrónico.

Pupeteer y Mongoose

Puppeteer es una biblioteca de Node.js desarrollada por Google que permite automatizar tareas en un navegador Chrome o Chromium. Con Puppeteer, se pueden crear scripts para realizar tareas como navegar por sitios web, rellenar formularios, hacer clic en botones, tomar capturas de pantalla y extraer datos de páginas web. Puppeteer también proporciona una interfaz para interactuar con una página web utilizando JavaScript, lo que permite realizar tareas avanzadas como simular eventos del usuario, ejecutar JavaScript en una página y acceder a elementos de la página que no están disponibles para el usuario final. Es importante mencionar que Puppeteer es una herramienta de automatización de navegador y no una librería de web scraping, pero puede ser utilizada para realizar tareas de scraping debido a su capacidad para interactuar con la página web y extraer información.

Mongoose es una biblioteca de Node.js que proporciona una interfaz de alto nivel para interactuar con una base de datos MongoDB. Mongoose proporciona una serie de características para trabajar con MongoDB, como la definición de esquemas para los documentos, la validación de datos, la construcción de consultas y la conexión a una base de datos.

Instalación

Inicializar un paquete de npm

```
npm init -y
```

Instalación de los paquetes necesarios

```
npm i pupeteer mongoose
```

crear un fichero index.js

```
touch index.js
```

añadir el script para ejecutar dicho fichero en el package.json

```
"start": node index.js
```

Guía

En nuestro fichero index.js

- Se importan las bibliotecas de Puppeteer y Mongoose al principio del script.
- Se crea un modelo de datos llamado "Data" utilizando Mongoose, con dos campos: título y precio.
- Se define una función "connect" para conectarse a la base de datos MongoDB.
- Se define una función "scrapeProducts" para extraer los datos de la página web.

  a. Se llama a la función "connect" para conectarse a la base de datos.
  b. Se inicializa el navegador con Puppeteer y se abre una nueva pestaña.
  c. Se navega a la URL específica (Amazon en este caso) y se realiza una búsqueda en el sitio utilizando la palabra "star wars".
  d. Se espera a que la página se cargue y se hace un scroll para ver más resultados de búsqueda.
  e. Utilizando el método ".$$eval" de Puppeteer se extraen los títulos y precios de los productos de la página.
  f. Se crea un array con los datos extraídos y se convierte en un objeto JSON.
  g. Se guarda el objeto JSON en la base de datos MongoDB.
  h. Se cierra el navegador y se imprime un mensaje de éxito.
- Finalmente, se llama a la función "scrapeProducts" para iniciar el proceso de extracción y guardado de datos.

Code

```
const puppeteer = require('puppeteer')
const mongoose = require('mongoose')

const Data = mongoose.model('Data', new mongoose.Schema({
  title: String,
  price: String
}))

const connect = async () => {
  try {
    const URI = 'mongodb+srv://user:password@cluster0.qdiqj9n.mongodb.net/nameProject?retryWrites=true&w=majority'
    await mongoose.connect(URI, { useNewUrlParser: true, useUnifiedTopology: true })
    console.log('Connected to DB 🚀')
  } catch (error) {
    console.error('Not connected to DB ❌')
  }
}

const scrapeProducts = async () => {
  await connect()

  const url = 'https://www.amazon.es/'

  const browser = await puppeteer.launch({
    headless: false,
    defaultViewport: null,
    args: ['--start-maximized']
  })

  const page = await browser.newPage()

  await page.goto(url)

  await page.type('#twotabsearchtextbox', 'star wars')

  await page.click('#nav-search-submit-button')

  await page.waitForSelector('.s-pagination-next')

  const title = await page.$$eval('h2 span.a-color-base', (nodes) =>
    nodes.map((n) => n.innerText)
  )
```

```
  const price = await page.$$eval('span.a-price[data-a-color="base"] span.a-offscreen', (nodes) =>
    nodes.map((n) => n.innerText)
  )

  const amazonProduct = title.map((value, index) => {
    return {
      title: title[index],
      price: price[index]
    }
  })


  amazonProduct.map(async (data) => {
    const dataSchema = new Data(data)
    try {
      await dataSchema.save()
      console.log(`Succesfully saved ${dataSchema.title} to the database 🤘`)
    } catch (error) {
      console.error(`Failed to save ${dataSchema.title} to the database ❌`)
    }
  })

  await browser.close()
  console.log('All saved Succesfully - Live Rock 🤘🧙')



}

scrapeProducts()
```

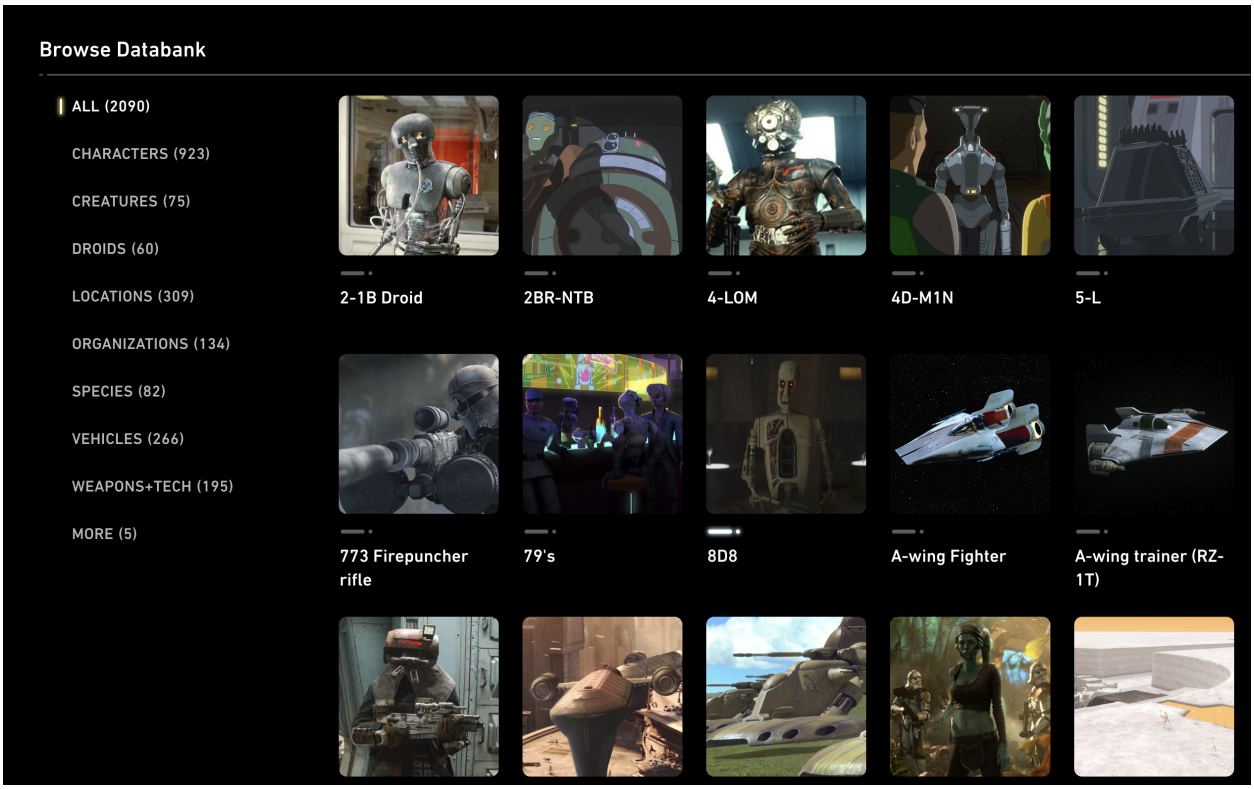
Star wars scraping

En este caso vamos a obtener una serie de personajes de la base de datos oficial de Star Wars y generar un json de objetos con los mismos.

Databank - StarWars.com

Learn about Star Wars characters, planets, ships, vehicles, droids, and more in the official Star Wars Databank at StarWars.com.

https://www.starwars.com/databank



Para ello crearemos un fichero index.js el cual podremos ejecutar con Node a través de un script de arranque simple.

```
const puppeteer = require("puppeteer"); //Importamos puppeteer

const fs = require("fs"); // Importamos fs

const url = "https://www.starwars.com/databank"; //Almacenamos la URL

const selector = ".building-block-wrapper"; //Almacenamos el selector de los productos que hemos descubierto insepeccionando la web

const scrapping = async () => {
  //Lanzamos puppeteer
  const browser = await puppeteer.launch();
  //Con headless false se ve fisicamente el navegador cuando arranca el script
  //Abrimos una nueva página en el navegador
  const page = await browser.newPage();
  //Vamos a la URL que hemos definido al principio
  await page.goto(url);
  //Vamos a clickar 6 veces en el botón .show_more para que nos muestre hasta cierto limite x personajes de la plantilla principal
  await page.click(".show_more");
  await page.click(".show_more");
  await page.click(".show_more");
  await page.click(".show_more");
  await page.click(".show_more");
  await page.click(".show_more");

  //Recuperamos con $$eval todos los nodos y nos quedaremos mediante un mapeo con los elementos name, image y description de cada uno de lo
  const data = await page.$$eval(selector, (nodes) => {
    return nodes.map((node) => ({
      name: node.querySelector(".title").innerText,
      image: node.querySelector(".display-check-link").firstElementChild
        .currentSrc,
      description: node.querySelector(".desc").innerText,
    }));
  });

  //Con FS escribiremos un nuevo fichero .json con los productos
  fs.writeFile("./data.json", JSON.stringify(data), (err) =>
    err ? console.log(err) : null
```

```
  );

  //Cerramos el navegador
  await browser.close();
};

scrapping();
```