Manuel Technique : Bibliothèque de Web Scraping

Introduction

Cette bibliothèque est un outil modulaire conçu pour effectuer un web scraping efficace et structuré. Elle permet d'extraire des liens, de télécharger des fichiers spécifiques (PDF/EPUB), et d'explorer des pages web selon des critères définis.

Fonctionnalités principales

- Téléchargement de fichiers PDF et EPUB depuis des pages web.

- Exploration récursive de pages web jusqu'à une profondeur maximale.

- Gestion des liens relatifs et des URL visitées.

- Architecture modulaire et réutilisable.

Structure de la Bibliothèque

1. File Downloader : Responsable du téléchargement des fichiers.

Méthodes :

- __init__(self, download_folder ='downloads') : Initialise le répertoire de téléchargement.

- download_file(self, file_url) : Télécharge un fichier depuis l'URL donnée.

Manuel Technique : Bibliothèque de Web Scraping

2. LinkExtractor : Responsable de l'extraction et du filtrage des liens.

Méthodes :

- extract_links(self, html_content, base_url) : Extrait tous les liens (absolus) d'une page HTML.

- filter_links(self, links, extensions=None) : Filtre les liens selon les extensions de fichiers. Exemple :

extractor = LinkExtractor() html = "PDF<a

href='/next-page'>Next" links = extractor.extract_links(html,

'https://example.com') filtered = extractor.filter_links(links, ['.pdf'])

3. WebScraper

Orchestre le processus de scraping en combinant les téléchargements et l'exploration des liens.

Méthodes :

- __init__(self, downloader, extractor) : Initialise le scraper avec les composants FileDownloader et LinkExtractor.

- scrape(self, start_url, max_depth, max_files) : Démarre le processus de scraping.

Exemple :

Downloader = FileDownloader()

Manuel Technique : Bibliothèque de Web Scraping

extractor = LinkExtractor() scraper =

WebScraper(downloader, extractor)

scraper.scrape('https://example.com', max_depth=2, max_files=10)

Utilisation

Etape 1 : Initialisation

Creez les instances des composants :

downloader = FileDownloader() extractor

= LinkExtractor()

scraper = WebScraper(downloader, extractor)

Etape 2 : Définition des paramètres Spécifiez :

start_url = 'https://example.com'

max_depth = 2 max_files = 10

Etape 3 : Lancement du scraping Démarrez le

scraping avec :

scraper.scrape(start_url, max_depth, max_files)

Bonnes pratiques

Manuel Technique : Bibliothèque de Web Scraping

1. Respectez les politiques des sites web :

- Vérifiez le fichier robots.txt avant de scraper un site.

- Ajoutez des délais (time.sleep) entre les requêtes pour réduire la charge serveur.

2. Limiter les profondeurs :

- Utilisez une faible profondeur pour éviter de scraper inutilement un trop grand nombre de pages.

3. Log des erreurs :

- Ajoutez un mécanisme de journalisation pour suivre les échecs de téléchargement ou d'accès aux pages.

Extensions possibles

1. Ajout de types de fichiers :

- Modifiez filter_links dans LinkExtractor pour supporter dautres extensions.

2. Gestion avancée des erreurs :

- Implémentez un journal détaillé ou relancez les échecs dans une prochaine session.

3. Interface utilisateur :

- Construisez une interface CLI ou GUI pour simplifier l'utilisation.

Conclusion

Cette bibliothèque fournit une base robuste pour des taches de web scraping ciblées.

Grace a son architecture modulaire, elle est extensible et facilement adaptable a des besoins spécifiques.