# 🚀 Day 16 of #30DaysofWebScraping: Building Crawlers with Scrapy

I am continuously working on exciting projects related to develop my skills in web crawling using Scrapy. Here is the task where we scrape movie detailed information (title, year, genre, ratings, etc) from a structured website **(imdb.com)**. The project gave me the chance to learn the in and out of user agent management, link-following, and pagination handling.

## What I Did Today

1. **Built Crawlers with Scrapy**

I dug into Scrapy "**CrawlSpider**" class which makes it easier to crawl and follow links across different pages. Traversing the website was straightforward, allowing us to scrape detailed movie data.

2. **User-Agent Handling**

I used custom User-Agent headers to mimic a real browser and pass detection. I made all requests contain the correct header with the "**process_request**" method to limit the number of times that the website flagged me.

3. **Multi-Level Crawling**

Using the "**LinkExtractor**" in the Scrapy framework, allowed me to automatically extract and follow relevant links, like movie details pages and pagination links. This made it easy for the spider to collect data from multiple pages.

4. **Extracted Movie Details**

For each movie, I scraped:

- **Title**: The name of the movie.
- **Year**: The release year.
- **Duration**: The runtime of the movie.
- **Genre**: The movie's primary genre.
- **Rating**: The IMDb user rating.
- **URL**: The movie's unique webpage link.

## Key Takeaways

1. **CrawlSpider Simplifies Navigation**: Scrapy's **CrawlSpider** made navigating and scraping a multi-page site much easier.
2. **User-Agent Rotation for Anti-Scraping:** Customizing headers proved crucial for bypassing detection.
3. **Efficient Link Extraction**: Leveraging **LinkExtractor** automated the process of identifying and following important links on the website.

## Challenges Faced

1. **Handling Dynamic Pagination**: To make sure the spider didn't miss any of the pages, pagination links required careful inspection.
2. **Data Validation**: Ensuring no duplicate entries were included in the output took some extra debugging.

## Reflections & What's Next

Day 16 reinforced the power of Scrapy in handling structured websites efficiently and extract meaningful data. Implementing this simple crawler helped solidify my understanding the potential of Scrapy while providing a pragmatic lesson on the importance of mindfulness when designing resilient spiders.

Next, I'll explore Splash, a tool for rendering JavaScript on dynamic websites. This will unlock the ability to scrape content from even the most challenging, JS-heavy web pages. Let's keep pushing the boundaries of what's possible! 💻 ✨