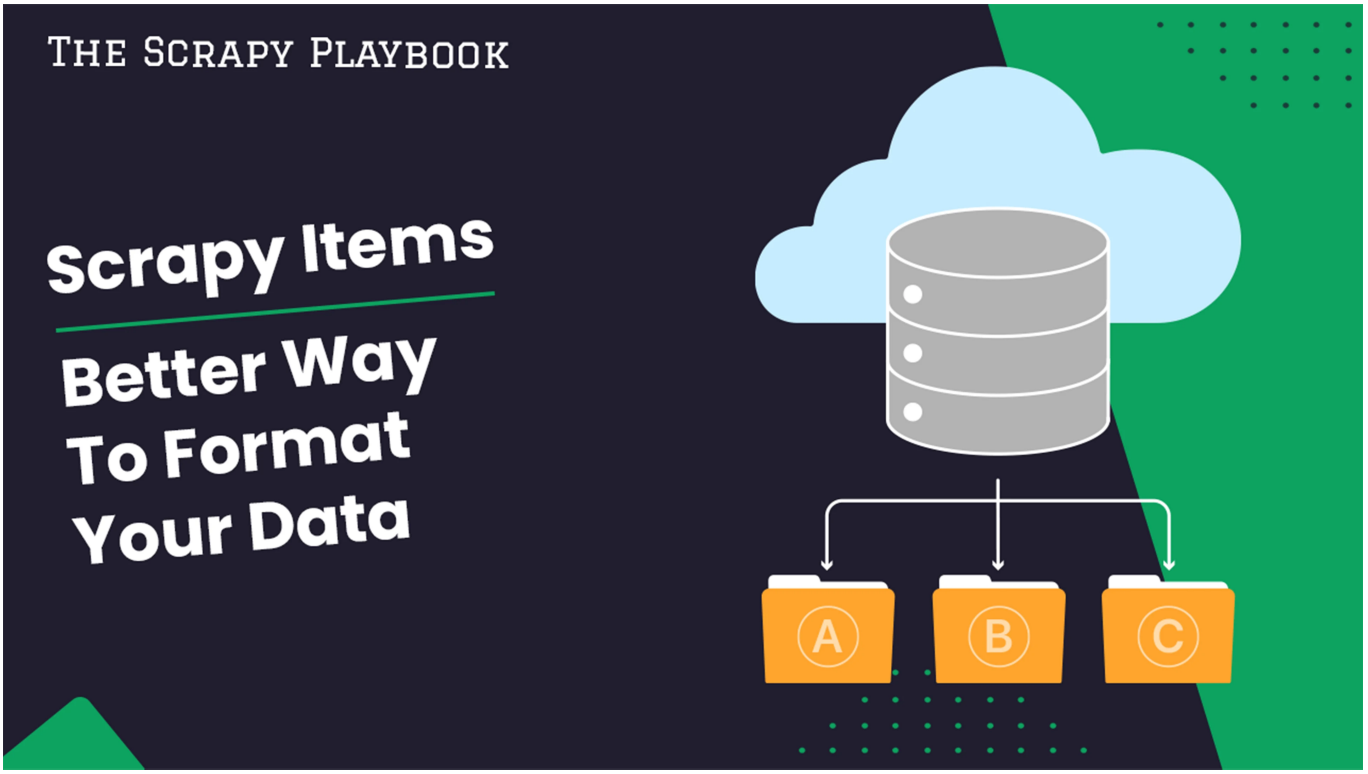

Scrapy Items: The Better Way To Format Your Data

The goal of web scraping is to turn unstructured HTML data into clean and structured data that can be used in our applications and data pipelines.

Oftentimes, developers just yield their scraped data in the form of a dictionary when starting out with Scrapy, however there is a better way which is called **Scrapy Items**.

In this guide, we're going to walk through:

- What Are Scrapy Items & Why Should We Use Them?
- How To Integrate Items Into Your Spiders

Need help scraping the web?

Then check out **ScrapeOps**, the complete toolkit for web scraping.

Proxy Manager **Scraper Monitoring** **Job Scheduling**

What Are Scrapy Items & Why Should We Use Them?

Scrapy Items are a predefined data structure that holds your data.

Instead of yielding your scraped data in the form of a dictionary for example, you define a Item schema beforehand in your `items.py` file and use this schema when scraping data.

This enables you to quickly and easily check what structured data you are collecting in your project, it will raise exceptions if you try and create incorrect data with your Item.

Because of this, using Scrapy Items have a number of advantages:

- Structures your data and gives it a clear schema.
- Enables you to easily clean and process your scraped data.
- Enables you to validate, deduplicate and monitor your data feeds.
- Enables you to easily store and export your data with Scrapy Feed Exports.
- Makes using Scrapy Item Pipelines & Item Loaders.

Scrapy supports multiple types of data formats that are automatically converted into Scrapy Items when yielded:

- Dictionaries
- Dataclass Objects
- Attrs Objects

However, defining your own **Item object** in your `items.py` file is normally the best option.

How To Integrate Items Into Your Spiders

Creating an **Item** in Scrapy is very straight forward. Simply open your `items.py` file and define the data you would like to scrape by inheriting from the Scrapy Item class.

```
# items.py

from scrapy.item import Item, Field

class QuoteItem(Item):
    text = Field()
    tags = Field()
    author = Field()
```

Then inside in your spider, instead of yielding a dictionary you would create a new Item with the scraped data before yielding it.

```
# items.py

import scrapy
from items_demo.items import QuoteItem


class QuotesSpider(scrapy.Spider):
        name = 'quotes'

        def start_requests(self):
                url = 'https://quotes.toscrape.com/'
                yield scrapy.Request(url, callback=self.parse)

        def parse(self, response):
                quote_item = QuoteItem()
                for quote in response.css('div.quote'):
                        quote_item['text'] = quote.css('span.text::text').get()
                        quote_item['author'] = quote.css('small.author::text').get()
                        quote_item['tags'] = quote.css('div.tags a.tag::text').getall()
                        yield quote_item
```

Now, all your scraped data will be contained in the structured `QuoteItem` we created which can then be sent through any active **Item Pipelines** to clean, validate and store your data.

More Scrapy Tutorials

This was a short guide, but that is because using Scrapy Items is very simple.

However, as a fundamental building block of Scrapy, it is vital that you learn what **Items** are and how to use them as they allow you to more effively use **Item Loaders** and **Item Pipelines**.

If you would like to learn more about Scrapy in general, then be sure to check out The Scrapy Playbook.