

# Achieving Common Spider Behaviors Using Built-in Classes

---



**Eduardo Freitas**

BUSINESS AUTOMATION & DATA CAPTURE SPECIALIST



# Overview



Spiders overview

Types of Scrapy spiders

`scrapy.Spider`

Generic spiders

Implementing a `scrapy.Spider`

Implementing a `CrawlSpider`



# Spiders Overview

---



Spiders are classes where custom behaviors are defined for crawling and parsing pages.



# How Are Spiders Implemented

What can be crawled

How it can be crawled

How it can be parsed



# Types of Scrapy Spiders

---



# Scrapy Spider Types

`scrapy.Spider`

Generic spiders



# Generic Spiders



Scrapy has four different types of generic spiders



CrawlSpider - follows all links on a site based on certain rules



XMLFeedSpider - parses XML feeds by iterating through nodes



CSVFeedSpider - parses CSV feeds by iterating through rows



SitemapSpider - crawl a site by discovering the URLs using sitemaps





scrapy.Spider

---



name

allowed\_domains

start\_urls

start\_requests

parse

- ◀ Defines the name of the spider
- ◀ List of domains that the spider is allowed to crawl
- ◀ List of URLs where the spider will beginning to crawl from
- ◀ A method that must return an iterable object with the first requests to crawl for the spider
- ◀ Default method used by Scrapy to process downloaded responses, when their requests don't specify a callback



# Processing Multiple Requests

```
import scrapy
```

```
class CoolSpider(scrapy.Spider):
```

```
    name = 'website.com'  
    allowed_domains = ['website.com']  
    start_urls = [  
        'http://www.website.com/page1.html',  
        'http://www.website.com/page2.html',  
    ]
```

```
    def parse(self, response):
```

```
        for h1 in response.xpath('//h1').getall():  
            yield {"title": h1}
```

```
        for href in response.xpath('//a/@href').getall():  
            yield scrapy.Request(response.urljoin(href), self.parse)
```



# CrawlSpider

---



rules

rule

parse\_start\_url

link\_extractor

- ◀ List of one or more rule objects
- ◀ Defines a behavior for crawling a site
- ◀ This method is called for the start\_urls responses
- ◀ Defines how links will be extracted from each crawled page



# CrawlSpider with Rules

```
import scrapy
from scrapy.spiders import CrawlSpider, Rule
from scrapy.linkextractors import LinkExtractor
```

```
class CoolSpider(CrawlSpider):
```

```
    name = 'website.com'
    allowed_domains = ['website.com']
    start_urls = ['http://www.website.com']
```

```
    rules = (
        Rule(LinkExtractor(allow=('product\.php', )), callback='parse_item'),
    )
```

```
    def parse_item(self, response):
```

```
        item = scrapy.Item()
        item['name'] = response.xpath('//td[@id="prod_name"]/text()').get()
        item['link_text'] = response.meta['link_text']
        return item
```



# XMLFeedSpider

---



iterator

iternodes

html/xml

itertag

◀ Defines the type of iterator to use, which defaults to iternodes

◀ Fast iterator based on regular expressions

◀ Uses DOM parsing. Load all DOM in memory

◀ This method is called for the start\_urls responses





# XMLFeedSpider

```
import scrapy
from scrapy.spiders import XMLFeedSpider
from project.items import ProductItem
```

```
class CoolSpider(XMLFeedSpider):
```

```
    name = 'website.com'
    allowed_domains = ['website.com']
    start_urls = ['http://www.website.com/products.xml']
    itertag = 'product'
```

```
    def parse_node(self, response, node):
```

```
        item = ProductItem()
        item['product'] = node.xpath('product').get()
        return item
```

# CSVFeedSpider

---



delimiter

quotechar

headers

◀ Represents the separator char for each field in the CSV.  
Defaults to ,

◀ Represents the enclosure char for each field in the CSV.  
Defaults to “

◀ Column names within the CSV



# CSVFeedSpider

```
import scrapy
from scrapy.spiders import CSVFeedSpider
from project.items import ProductItem
```

```
class CoolSpider(CSVFeedSpider):
```

```
    name = 'website.com'
    allowed_domains = ['website.com']
    start_urls = ['http://www.website.com/products.csv']
    delimiter = ';'
    quotechar = "'"
    headers = ['product', 'price']
```

```
    def parse_row(self, response, row):
```

```
        item = ProductItem()
        item['product'] = row['product']
        item['price'] = row['price']
        return item
```



# SitemapSpider

---



sitemap\_urls

sitemap\_rules

sitemap\_follow

sitemap\_alternate\_links

sitemap\_filter

- ◀ List of URLs pointing to the sitemap
- ◀ `[('/product/', 'parse_product')]`
- ◀ List of sitemap regular expressions that should be followed
- ◀ Alternate links for a specific URL
- ◀ Filter to select sitemap entries based on attributes



# SitemapSpider

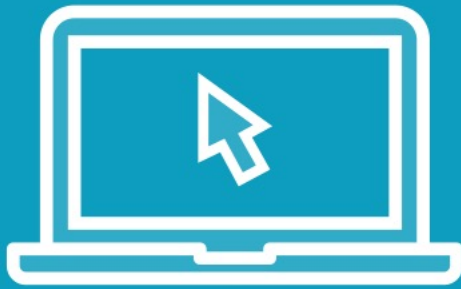
```
import scrapy
from scrapy.spiders import SitemapSpider
```

```
class CoolSpider(SitemapSpider):
    sitemap_urls = ['http://www.website.com/sitemap.xml']
    sitemap_rules = [
        ('/product/', 'parse_product'),
        ('/prodcategory/', 'parse_prod_category'),
    ]
```

```
def parse_product(self, response):
    # scrape each product

def parse_prod_category(self, response):
    # scrape each product category
```

# Demo

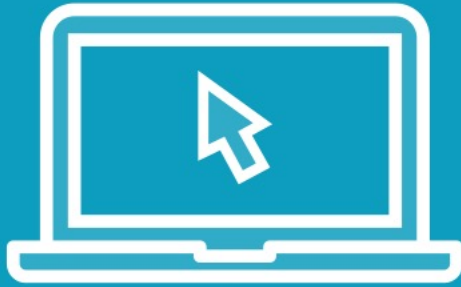


## Implementing a scrapy.Spider





# Demo



## Implementing a CrawlSpider



# Summary



Overview of spiders types

XMLFeedSpider

CSVFeedSpider

SitemapSpider

Implemented a scrapy.Spider

Implemented a CrawlSpider