

Influencing Scrapy Crawling



Eduardo Freitas

BUSINESS AUTOMATION & DATA CAPTURE SPECIALIST



Overview



Allow and deny rules

Processors

Item loaders

Implementing a scraping pipeline



Allow and Deny Rules



You can control what URLs a CrawlSpider
crawls by using
Allow and Deny rules.



```
allow=( 'catalogue/category/books' )
```

◀ Indicates the URL paths that will be crawled

```
deny=( 'erotica_50', 'crime_51' )
```

◀ Indicates the URL paths that won't be crawled



Processors



Pre and post-processing of raw scraped data to format and clean it.



Processors Types

Raw scraped data

Input

Output



Built-in Processors

Identity

Join

MapCompose

TakeFirst

Compose

SelectJmes



Input and Output Processors – items.py

```
import scrapy
from scrapy.loader.processors import Join, MapCompose, TakeFirst
from w3lib.html import remove_tags
```

```
def filter_price(value):
    if value.isdigit():
        return value
```

```
class Book(scrapy.Item):
    title = scrapy.Field(
        input_processor=MapCompose(remove_tags),
        output_processor=Join(),
    )
    price = scrapy.Field(
        input_processor=MapCompose(remove_tags, filter_price),
        output_processor=TakeFirst(),
    )
```



Item Loaders



Convenient mechanisms to populate
items.



Item Loaders

```
from scrapy.loader import ItemLoader
from myspider.items import Book
```

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

```
    book = ItemLoader(item=Book(), response=response)
```

```
    book.add_xpath('title', ' //div[@class="book_title"]')
    book.add_xpath('isbn', ' //div[@class="book_isbn"]')
    book.add_xpath('price', ' //p[@id="price"]')
    book.add_css('stock', 'p#stock')
    book.add_value('updated', 'today')
```

```
    return book.load_item()
```

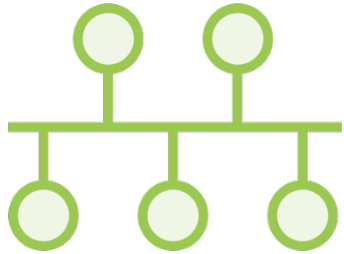
Item Pipelines



Easy way to chain transformations or
checks on scraped data.



Use Cases



Cleaning raw scraped HTML data

Validating scraped data

Checking that items contain values

Checking and removing duplicates

Storing items in a file or database

Item Pipeline

```
from scrapy.exceptions import DropItem
```

```
class PricePipeline(object):
```

```
    sales_tax = 1.15
```

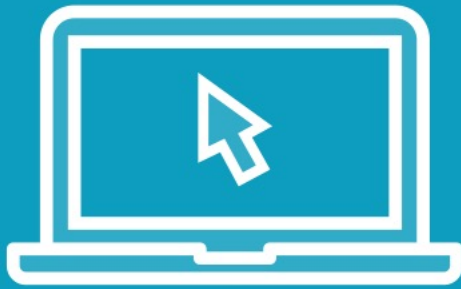
```
    def process_item(self, item, spider):
```

```
        if item.get('price'):
            if item.get('price_without_sales_tax'):
                item['price'] = item['price'] * self.sales_tax
            return item
```

```
        else:
```

```
            raise DropItem("Item without a price: %s" % item)
```

Demo



Implementing a scraping pipeline



Summary



Modifying crawling behavior

Input and output processors

Loaders and pipelines

Implementing a scraping pipeline

