


WEB SCRAPING

PAULO CÉSAR TUYA

ÍNDICE


- Módulo 1: Introducción a Web Scraping
- Módulo 2: Herramientas para el Análisis de una Página Web
- **Módulo 3: Web Scraping con Scrapy** 
 - ¿Qué es Scrapy?
 - Selectores CSS y XPATH
 - Arquitectura de un Proyecto en Scrapy
 - Callbacks
 - Pipelines
- Módulo 4: Scraping en páginas dinámicas
- Módulo 5: Despliegue de un Spider



¿QUÉ ES SCRAPY?

SCRAPY



- **Framework** para extracción de datos web
- Ventajas
 - Escalabilidad
 - Rendimiento
 - Soporte de la Comunidad
- Herramientas 
 - Selectores CSS y XPATH
 - Exporta a distintos formatos
 - Consola interactiva iPython

¿SCRAPY VS BEAUTIFULSOUP?

- Son fundamentalmente **herramientas distintas**
- BeautifulSoup: **librería** para **parsing** de Markup
- Scrapy: **Framework** para **extracción de datos**
- Es posible usar bs dentro de Scrapy



Scrapy

BeautifulSoup





¿SCRAPY VS BEAUTIFULSOUP?

BeautifulSoup	Scrapy
Muy fácil de aprender y amigable para principiantes	Requiere más esfuerzo puesto que consiste en un framework completo con distintos conceptos
No es muy utilizado y no tiene una comunidad grande	Gran cantidad de proyectos, ejemplos, guías, tutoriales, plugins y comunidades
Es útil para proyectos pequeños pero sufre problemas de escalabilidad	A medida que el proyecto crece, es fácil de mantener a través de middlewares o pipelines



SPIDERS

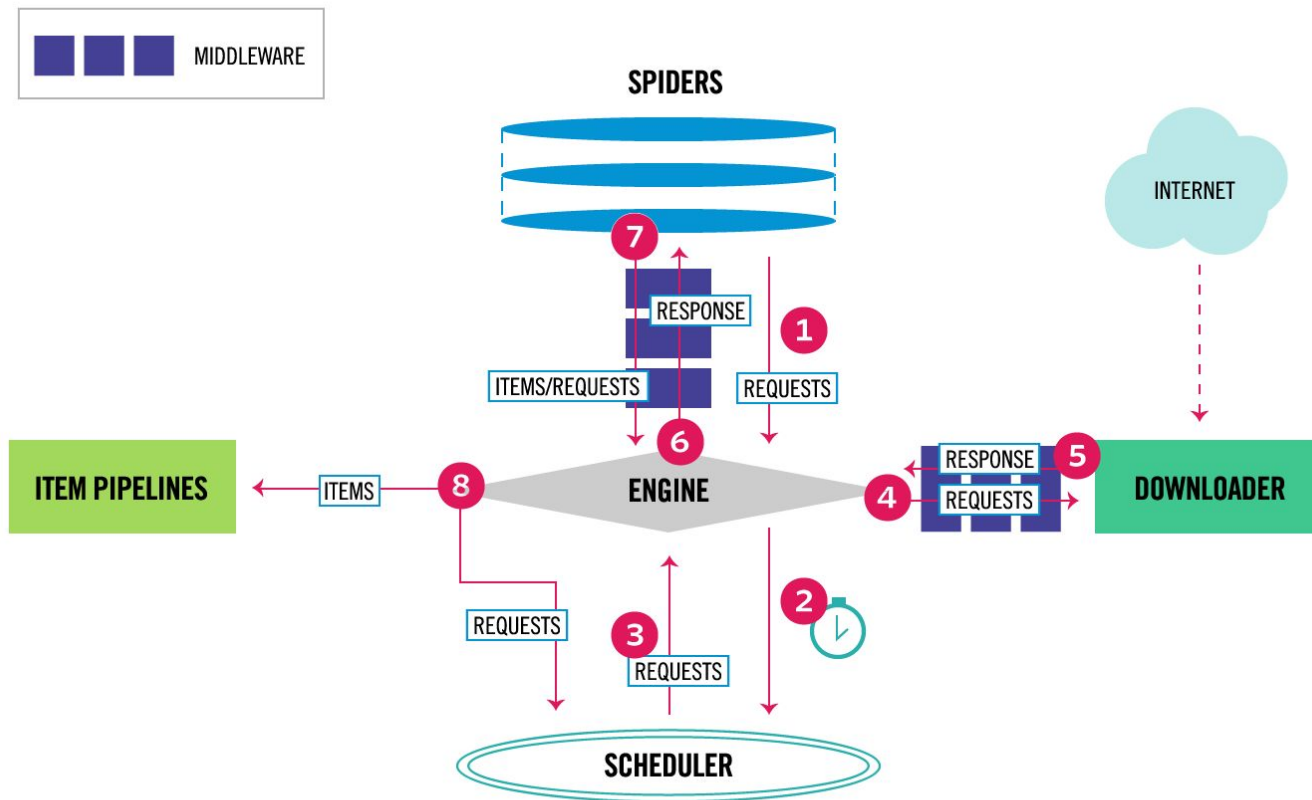


scrapy.Spider

```
class scrapy.spiders.Spider
```

- Pieza fundamental del framework **Scrapy**
- Dentro de Python son **clases** que deben definirse
- Definen cómo se hará el **scraping**
 - Cómo seguir los enlaces
 - Cómo extraer datos

FLUJO DE TRABAJO DE SCRAPY





SELECTORES CSS & XPATH



SELECTORES CSS

Ejemplo	Selector	Descripción
*	Universal	Devuelve todos los elementos del documento
nombre	de Tipo	Devuelve los elementos con la etiqueta nombre
.nombre	de Clase	Devuelve los elementos con la clase nombre
#nombre	de ID	Devuelve los elementos con el ID nombre
[atrib]	de atributo	Devuelve los elementos que contengan el atributo atrib
[atrib="valor"]	de atributo	Devuelve los elementos cuyo valor de atrib sea valor

<https://www.w3schools.com/cssref/trysel.asp>



LENGUAJE XPATH

- **XML Path** Language
- Lenguaje para construir expresiones que recorren y procesan un documento de **Markup**
- Aprovecha la **estructura jerárquica** de los lenguajes de Markup
- Pensado inicialmente para usarse en XML, también puede usarse en código **HTML**

SELECTORES XPATH

Ejemplo	Descripción
nombre	Devuelve todos los nodos con nombre “nombre”
/nombre	Selecciona desde el nodo raíz
//nombre	Selecciona nodos en cualquier parte del documento
.	Selecciona el nodo actual
..	Selecciona el nodo padre del actual
@	Selecciona los atributos



SELECTORES XPATH

Ejemplo	Descripción
nombre/nombre2	Selecciona nombre2 que sea hijo de nombre
nombre//nombre2	Selecciona nombre2 que sea descendiente de nombre
*	Selecciona todos los nodos
[]	Permite introducir predicados que filtran nodos
//nombre[@atr='azul']	Selecciona todos los nodos llamados nombre cuyo valor de atr sea azul
/nombre/iter[1]	Selecciona el primer nodo iter que sea hijo de nombre



RECURSOS EN LÍNEA

- <https://www.freeformatter.com/xpath-tester.html>
- <https://codebeautify.org/Xpath-Tester>
- <http://xpather.com/>
- <https://extendsclass.com/xpath-tester.html>
- https://www.w3schools.com/xml/xpath_syntax.asp

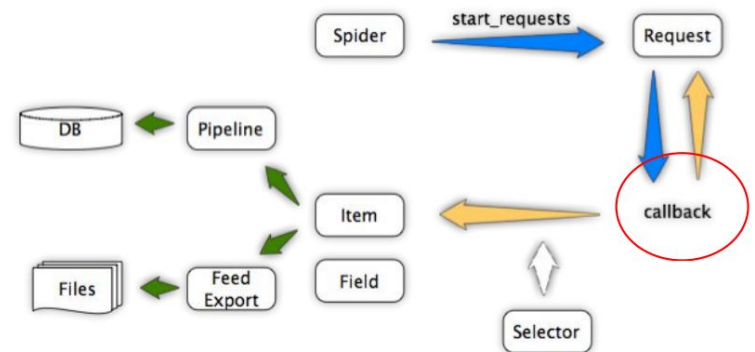


ARQUITECTURA DE UN PROYECTO EN SCRAPY

CALLBACKS

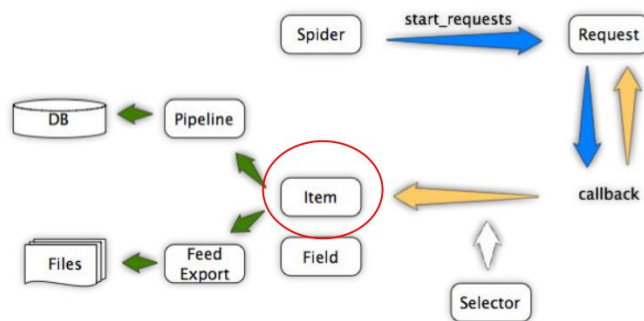


- Funciones que se llaman después de recibir la respuesta
- Se encargan de **parsear** el contenido y devolver la data deseada
- Pueden hacer return de:
 - Un request
 - Un item
 - Un iterable de estas categorías



ITEMS

- Contenedores usados por Scrapy
- Son análogos a los **diccionarios** de Python



```
import scrapy

class Product(scrapy.Item):
    name = scrapy.Field()
    price = scrapy.Field()
    stock = scrapy.Field()
    last_updated = scrapy.Field(serializer=str)
```

ITEM LOADERS

- Proporcionan un mecanismo para llenar los datos del item
- Debe definirse su comportamiento por el usuario

```
from scrapy.loader import ItemLoader
from myproject.items import Product

def parse(self, response):
    l = ItemLoader(item=Product(), response=response)
    l.add_xpath('name', '//div[@class="product_name"]')
    l.add_xpath('name', '//div[@class="product_title"]')
    l.add_xpath('price', '//p[@id="price"]')
    l.add_css('stock', 'p#stock')
    l.add_value('last_updated', 'today') # you can also use literal values
    return l.load_item()
```



PIPELINES

- Después de extraer un **item**, se le envía a una **pipeline**
- Las pipelines tienen **componentes** que se encargan de procesar la data extraída
- Ejemplos de uso de pipelines:
 - Limpieza de datos
 - Validación de datos
 - Revisión de duplicados
 - Almacenamiento en base de datos



ACTIVIDAD MÓDULO 3

**GRACIAS
POR SU ATENCIÓN**