Scrapy Python web crawling

CONCEPTS

> Develop selectors in scrapy shell first, then paste them into the spider's parse().

Spider start requests
↓ `Request`
Downloader fetches page
↓ `Response`
parse() callback
↓ `yield item`
Item pipeline / feed clean, store

PROJECT SETUP

`scrapy startproject <name> [dir]`
Create a project skeleton

`scrapy genspider <name> <domain>`
Create a spider from a template

`scrapy genspider -t crawl <name> <domain>`
Spider with link-following rules

`scrapy list`
List spiders in this project

RUNNING SPIDERS

`scrapy crawl <spider>`
Run a project spider

`scrapy runspider <file.py>`
Run a standalone spider file

`scrapy crawl <spider> -a <name>=<value>`
Pass a spider argument

`scrapy crawl <spider> -s <NAME>=<value>`
Override a setting once

OUTPUT (FEEDS)

`scrapy crawl <spider> -o <file>`
Append items to a file

`scrapy crawl <spider> -O <file>`
Overwrite the file with items

OUTPUT (FEEDS) (cont.)

`scrapy crawl <spider> -O out.jsonl`
One JSON object per line

`scrapy crawl <spider> -O out.csv`
Write CSV

SHELL AND INSPECTION

`scrapy shell "<url>"`
Open the console on a page

`scrapy fetch "<url>"`
Print the body as Scrapy sees it

`scrapy view "<url>"`
Open the page as Scrapy got it

`scrapy parse --spider=<name> "<url>"`
Run the spider's callback on one URL

SHELL OBJECTS AND SELECTORS

`response.css("<sel>")`
CSS selector on the page

`response.css("<sel>::text").getall()`
All matching text nodes

`response.css("<sel>::attr(href)").get()`
First attribute value or None

`response.xpath("//<tag>")`
XPath selector on the page

`fetch("<url>")`
Load another page into the shell

SPIDER CODE

`yield {"<key>": <value>}`
Emit an item from parse()

`yield response.follow(<href>, callback=self.<fn>)`
Follow a link to another callback

`response.follow_all(css="<sel>", callback=self.<fn>)`
Follow every matching link