**[스크래피 2.7 문서](https://docs.scrapy.org/en/latest/index.html)**

**도움을 받다**

**첫 번째 단계**

**기본 컨셉**

**기본 제공 서비스**

**특정 문제 해결**

**스크래피 확장**

**나머지는 모두**

[**한 눈에 스크랩**](https://docs.scrapy.org/en/latest/intro/overview.html)

**import** **scrapy**

**class** **QuotesSpider**(scrapy.Spider):

name = 'quotes'

start\_urls = [

'https://quotes.toscrape.com/tag/humor/',

]

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

**for** quote **in** response.css('div.quote'):

**yield** {

'author': quote.xpath('span/small/text()').get(),

'text': quote.css('span.text::text').get(),

}

next\_page = response.css('li.next a::attr("href")').get()

**if** next\_page **is** **not** **None**:

**yield** response.follow(next\_page, self.parse)

scrapy runspider quotes\_spider.py -o quotes.jsonl

{"author": "Jane Austen", "text": "**\u201c**The person, be it gentleman or lady, who has not pleasure in a good novel, must be intolerably stupid.**\u201d**"}

{"author": "Steve Martin", "text": "**\u201c**A day without sunshine is like, you know, night.**\u201d**"}

{"author": "Garrison Keillor", "text": "**\u201c**Anyone who thinks sitting in church can make you a Christian must also think that sitting in a garage can make you a car.**\u201d**"}

...

**예제 거미의 연습**

**import** **scrapy**

**class** **QuotesSpider**(scrapy.Spider):

name = 'quotes'

start\_urls = [

'https://quotes.toscrape.com/tag/humor/',

]

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

**for** quote **in** response.css('div.quote'):

**yield** {

'author': quote.xpath('span/small/text()').get(),

'text': quote.css('span.text::text').get(),

}

next\_page = response.css('li.next a::attr("href")').get()

**if** next\_page **is** **not** **None**:

**yield** response.follow(next\_page, self.parse)

scrapy runspider quotes\_spider.py -o quotes.jsonl

{"author": "Jane Austen", "text": "**\u201c**The person, be it gentleman or lady, who has not pleasure in a good novel, must be intolerably stupid.**\u201d**"}

{"author": "Steve Martin", "text": "**\u201c**A day without sunshine is like, you know, night.**\u201d**"}

{"author": "Garrison Keillor", "text": "**\u201c**Anyone who thinks sitting in church can make you a Christian must also think that sitting in a garage can make you a car.**\u201d**"}

...

**방금 무슨 일이?**

**또 뭐?**

**무엇 향후 계획?**

[**설치 안내서**](https://docs.scrapy.org/en/latest/intro/install.html)

conda install -c conda-forge scrapy

pip install Scrapy

conda install -c conda-forge scrapy

sudo apt-get install python3 python3-dev python3-pip libxml2-dev libxslt1-dev zlib1g-dev libffi-dev libssl-dev

pip install scrapy

xcode-select --install

echo "export PATH=/usr/local/bin:/usr/local/sbin:$PATH" >> ~/.bashrc

source ~/.bashrc

brew install python

brew update; brew upgrade python

pip install Scrapy

[…]

File "[…]/site-packages/twisted/protocols/tls.py", line 63, in <module>

from twisted.internet.\_sslverify import \_setAcceptableProtocols

File "[…]/site-packages/twisted/internet/\_sslverify.py", line 38, in <module>

TLSVersion.TLSv1\_1: SSL.OP\_NO\_TLSv1\_1,

AttributeError: 'module' object has no attribute 'OP\_NO\_TLSv1\_1'

pip install twisted[tls]

**지원되는 Python 버전**

**스크래피 설치**

conda install -c conda-forge scrapy

pip install Scrapy

**알아두면 좋은 것들**

**가상 환경 사용(권장)**

**플랫폼별 설치 참고 사항**

conda install -c conda-forge scrapy

sudo apt-get install python3 python3-dev python3-pip libxml2-dev libxslt1-dev zlib1g-dev libffi-dev libssl-dev

pip install scrapy

xcode-select --install

echo "export PATH=/usr/local/bin:/usr/local/sbin:$PATH" >> ~/.bashrc

source ~/.bashrc

brew install python

brew update; brew upgrade python

pip install Scrapy

**창**

conda install -c conda-forge scrapy

**우분투 14.04 이상**

sudo apt-get install python3 python3-dev python3-pip libxml2-dev libxslt1-dev zlib1g-dev libffi-dev libssl-dev

pip install scrapy

**맥 OS**

xcode-select --install

echo "export PATH=/usr/local/bin:/usr/local/sbin:$PATH" >> ~/.bashrc

source ~/.bashrc

brew install python

brew update; brew upgrade python

pip install Scrapy

**파이파이**

**문제 해결**

[…]

File "[…]/site-packages/twisted/protocols/tls.py", line 63, in <module>

from twisted.internet.\_sslverify import \_setAcceptableProtocols

File "[…]/site-packages/twisted/internet/\_sslverify.py", line 38, in <module>

TLSVersion.TLSv1\_1: SSL.OP\_NO\_TLSv1\_1,

AttributeError: 'module' object has no attribute 'OP\_NO\_TLSv1\_1'

pip install twisted[tls]

**AttributeError: '모듈' 개체에 'OP\_NO\_TLSv1\_1' 속성이 없습니다.**

[…]

File "[…]/site-packages/twisted/protocols/tls.py", line 63, in <module>

from twisted.internet.\_sslverify import \_setAcceptableProtocols

File "[…]/site-packages/twisted/internet/\_sslverify.py", line 38, in <module>

TLSVersion.TLSv1\_1: SSL.OP\_NO\_TLSv1\_1,

AttributeError: 'module' object has no attribute 'OP\_NO\_TLSv1\_1'

pip install twisted[tls]

[**스크랩 튜토리얼**](https://docs.scrapy.org/en/latest/intro/tutorial.html)

scrapy startproject tutorial

tutorial/

scrapy.cfg *# deploy configuration file*

tutorial/ *# project's Python module, you'll import your code from here*

\_\_init\_\_.py

items.py *# project items definition file*

middlewares.py *# project middlewares file*

pipelines.py *# project pipelines file*

settings.py *# project settings file*

spiders/ *# a directory where you'll later put your spiders*

\_\_init\_\_.py

**import** **scrapy**

**class** **QuotesSpider**(scrapy.Spider):

name = "quotes"

**def** start\_requests(self):

urls = [

'https://quotes.toscrape.com/page/1/',

'https://quotes.toscrape.com/page/2/',

]

**for** url **in** urls:

**yield** scrapy.Request(url=url, callback=self.parse)

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

page = response.url.split("/")[-2]

filename = f'quotes-*{*page*}*.html'

**with** open(filename, 'wb') **as** f:

f.write(response.body)

self.log(f'Saved file *{*filename*}*')

scrapy crawl quotes

... (omitted **for** brevity)

2016-12-16 21:24:05 [scrapy.core.engine] INFO: Spider opened

2016-12-16 21:24:05 [scrapy.extensions.logstats] INFO: Crawled 0 pages (at 0 pages/min), scraped 0 items (at 0 items/min)

2016-12-16 21:24:05 [scrapy.extensions.telnet] DEBUG: Telnet console listening on 127.0.0.1:6023

2016-12-16 21:24:05 [scrapy.core.engine] DEBUG: Crawled (404) <GET https://quotes.toscrape.com/robots.txt> (referer: **None**)

2016-12-16 21:24:05 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://quotes.toscrape.com/page/1/> (referer: **None**)

2016-12-16 21:24:05 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://quotes.toscrape.com/page/2/> (referer: **None**)

2016-12-16 21:24:05 [quotes] DEBUG: Saved file quotes-1.html

2016-12-16 21:24:05 [quotes] DEBUG: Saved file quotes-2.html

2016-12-16 21:24:05 [scrapy.core.engine] INFO: Closing spider (finished)

...

**import** **scrapy**

**class** **QuotesSpider**(scrapy.Spider):

name = "quotes"

start\_urls = [

'https://quotes.toscrape.com/page/1/',

'https://quotes.toscrape.com/page/2/',

]

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

page = response.url.split("/")[-2]

filename = f'quotes-*{*page*}*.html'

**with** open(filename, 'wb') **as** f:

f.write(response.body)

scrapy shell 'https://quotes.toscrape.com/page/1/'

scrapy shell "https://quotes.toscrape.com/page/1/"

[ ... Scrapy log here ... ]

2016-09-19 12:09:27 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://quotes.toscrape.com/page/1/> (referer: **None**)

[s] Available Scrapy objects:

[s] scrapy scrapy module (contains scrapy.Request, scrapy.Selector, etc)

[s] crawler <scrapy.crawler.Crawler object at 0x7fa91d888c90>

[s] item {}

[s] request <GET https://quotes.toscrape.com/page/1/>

[s] response <200 https://quotes.toscrape.com/page/1/>

[s] settings <scrapy.settings.Settings object at 0x7fa91d888c10>

[s] spider <DefaultSpider 'default' at 0x7fa91c8af990>

[s] Useful shortcuts:

[s] shelp() Shell help (print this help)

[s] fetch(req\_or\_url) Fetch request (**or** URL) **and** update local objects

[s] view(response) View response **in** a browser

**>>>** response.css('noelement')[0].get()

Traceback (most recent call last):

*...*

IndexError: list index out of range

**>>>** response.css("noelement").get()

scrapy shell 'https://quotes.toscrape.com'

**import** **scrapy**

**class** **QuotesSpider**(scrapy.Spider):

name = "quotes"

start\_urls = [

'https://quotes.toscrape.com/page/1/',

'https://quotes.toscrape.com/page/2/',

]

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

**for** quote **in** response.css('div.quote'):

**yield** {

'text': quote.css('span.text::text').get(),

'author': quote.css('small.author::text').get(),

'tags': quote.css('div.tags a.tag::text').getall(),

}

2016-09-19 18:57:19 [scrapy.core.scraper] DEBUG: Scraped **from** <200 https://quotes.toscrape.com/page/1/>

{'tags': ['life', 'love'], 'author': 'André Gide', 'text': '“It is better to be hated for what you are than to be loved for what you are not.”'}

2016-09-19 18:57:19 [scrapy.core.scraper] DEBUG: Scraped **from** <200 https://quotes.toscrape.com/page/1/>

{'tags': ['edison', 'failure', 'inspirational', 'paraphrased'], 'author': 'Thomas A. Edison', 'text': "“I have not failed. I've just found 10,000 ways that won't work.”"}

scrapy crawl quotes -O quotes.json

scrapy crawl quotes -o quotes.jsonl

**import** **scrapy**

**class** **QuotesSpider**(scrapy.Spider):

name = "quotes"

start\_urls = [

'https://quotes.toscrape.com/page/1/',

]

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

**for** quote **in** response.css('div.quote'):

**yield** {

'text': quote.css('span.text::text').get(),

'author': quote.css('small.author::text').get(),

'tags': quote.css('div.tags a.tag::text').getall(),

}

next\_page = response.css('li.next a::attr(href)').get()

**if** next\_page **is** **not** **None**:

next\_page = response.urljoin(next\_page)

**yield** scrapy.Request(next\_page, callback=self.parse)

**import** **scrapy**

**class** **QuotesSpider**(scrapy.Spider):

name = "quotes"

start\_urls = [

'https://quotes.toscrape.com/page/1/',

]

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

**for** quote **in** response.css('div.quote'):

**yield** {

'text': quote.css('span.text::text').get(),

'author': quote.css('span small::text').get(),

'tags': quote.css('div.tags a.tag::text').getall(),

}

next\_page = response.css('li.next a::attr(href)').get()

**if** next\_page **is** **not** **None**:

**yield** response.follow(next\_page, callback=self.parse)

**for** href **in** response.css('ul.pager a::attr(href)'):

**yield** response.follow(href, callback=self.parse)

**for** a **in** response.css('ul.pager a'):

**yield** response.follow(a, callback=self.parse)

anchors = response.css('ul.pager a')

**yield from** response.follow\_all(anchors, callback=self.parse)

**yield from** response.follow\_all(css='ul.pager a', callback=self.parse)

**import** **scrapy**

**class** **AuthorSpider**(scrapy.Spider):

name = 'author'

start\_urls = ['https://quotes.toscrape.com/']

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

author\_page\_links = response.css('.author + a')

**yield from** response.follow\_all(author\_page\_links, self.parse\_author)

pagination\_links = response.css('li.next a')

**yield from** response.follow\_all(pagination\_links, self.parse)

**def** parse\_author(self, response):

**def** extract\_with\_css(query):

**return** response.css(query).get(default='').strip()

**yield** {

'name': extract\_with\_css('h3.author-title::text'),

'birthdate': extract\_with\_css('.author-born-date::text'),

'bio': extract\_with\_css('.author-description::text'),

}

scrapy crawl quotes -O quotes-humor.json -a tag=humor

**import** **scrapy**

**class** **QuotesSpider**(scrapy.Spider):

name = "quotes"

**def** start\_requests(self):

url = 'https://quotes.toscrape.com/'

tag = getattr(self, 'tag', **None**)

**if** tag **is** **not** **None**:

url = url + 'tag/' + tag

**yield** scrapy.Request(url, self.parse)

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

**for** quote **in** response.css('div.quote'):

**yield** {

'text': quote.css('span.text::text').get(),

'author': quote.css('small.author::text').get(),

}

next\_page = response.css('li.next a::attr(href)').get()

**if** next\_page **is** **not** **None**:

**yield** response.follow(next\_page, self.parse)

**프로젝트 만들기**

scrapy startproject tutorial

tutorial/

scrapy.cfg *# deploy configuration file*

tutorial/ *# project's Python module, you'll import your code from here*

\_\_init\_\_.py

items.py *# project items definition file*

middlewares.py *# project middlewares file*

pipelines.py *# project pipelines file*

settings.py *# project settings file*

spiders/ *# a directory where you'll later put your spiders*

\_\_init\_\_.py

**우리의 첫 번째 거미**

**import** **scrapy**

**class** **QuotesSpider**(scrapy.Spider):

name = "quotes"

**def** start\_requests(self):

urls = [

'https://quotes.toscrape.com/page/1/',

'https://quotes.toscrape.com/page/2/',

]

**for** url **in** urls:

**yield** scrapy.Request(url=url, callback=self.parse)

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

page = response.url.split("/")[-2]

filename = f'quotes-*{*page*}*.html'

**with** open(filename, 'wb') **as** f:

f.write(response.body)

self.log(f'Saved file *{*filename*}*')

scrapy crawl quotes

... (omitted **for** brevity)

2016-12-16 21:24:05 [scrapy.core.engine] INFO: Spider opened

2016-12-16 21:24:05 [scrapy.extensions.logstats] INFO: Crawled 0 pages (at 0 pages/min), scraped 0 items (at 0 items/min)

2016-12-16 21:24:05 [scrapy.extensions.telnet] DEBUG: Telnet console listening on 127.0.0.1:6023

2016-12-16 21:24:05 [scrapy.core.engine] DEBUG: Crawled (404) <GET https://quotes.toscrape.com/robots.txt> (referer: **None**)

2016-12-16 21:24:05 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://quotes.toscrape.com/page/1/> (referer: **None**)

2016-12-16 21:24:05 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://quotes.toscrape.com/page/2/> (referer: **None**)

2016-12-16 21:24:05 [quotes] DEBUG: Saved file quotes-1.html

2016-12-16 21:24:05 [quotes] DEBUG: Saved file quotes-2.html

2016-12-16 21:24:05 [scrapy.core.engine] INFO: Closing spider (finished)

...

**import** **scrapy**

**class** **QuotesSpider**(scrapy.Spider):

name = "quotes"

start\_urls = [

'https://quotes.toscrape.com/page/1/',

'https://quotes.toscrape.com/page/2/',

]

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

page = response.url.split("/")[-2]

filename = f'quotes-*{*page*}*.html'

**with** open(filename, 'wb') **as** f:

f.write(response.body)

scrapy shell 'https://quotes.toscrape.com/page/1/'

scrapy shell "https://quotes.toscrape.com/page/1/"

[ ... Scrapy log here ... ]

2016-09-19 12:09:27 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://quotes.toscrape.com/page/1/> (referer: **None**)

[s] Available Scrapy objects:

[s] scrapy scrapy module (contains scrapy.Request, scrapy.Selector, etc)

[s] crawler <scrapy.crawler.Crawler object at 0x7fa91d888c90>

[s] item {}

[s] request <GET https://quotes.toscrape.com/page/1/>

[s] response <200 https://quotes.toscrape.com/page/1/>

[s] settings <scrapy.settings.Settings object at 0x7fa91d888c10>

[s] spider <DefaultSpider 'default' at 0x7fa91c8af990>

[s] Useful shortcuts:

[s] shelp() Shell help (print this help)

[s] fetch(req\_or\_url) Fetch request (**or** URL) **and** update local objects

[s] view(response) View response **in** a browser

**>>>** response.css('noelement')[0].get()

Traceback (most recent call last):

*...*

IndexError: list index out of range

**>>>** response.css("noelement").get()

scrapy shell 'https://quotes.toscrape.com'

**import** **scrapy**

**class** **QuotesSpider**(scrapy.Spider):

name = "quotes"

start\_urls = [

'https://quotes.toscrape.com/page/1/',

'https://quotes.toscrape.com/page/2/',

]

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

**for** quote **in** response.css('div.quote'):

**yield** {

'text': quote.css('span.text::text').get(),

'author': quote.css('small.author::text').get(),

'tags': quote.css('div.tags a.tag::text').getall(),

}

2016-09-19 18:57:19 [scrapy.core.scraper] DEBUG: Scraped **from** <200 https://quotes.toscrape.com/page/1/>

{'tags': ['life', 'love'], 'author': 'André Gide', 'text': '“It is better to be hated for what you are than to be loved for what you are not.”'}

2016-09-19 18:57:19 [scrapy.core.scraper] DEBUG: Scraped **from** <200 https://quotes.toscrape.com/page/1/>

{'tags': ['edison', 'failure', 'inspirational', 'paraphrased'], 'author': 'Thomas A. Edison', 'text': "“I have not failed. I've just found 10,000 ways that won't work.”"}

**거미를 실행하는 방법**

scrapy crawl quotes

... (omitted **for** brevity)

2016-12-16 21:24:05 [scrapy.core.engine] INFO: Spider opened

2016-12-16 21:24:05 [scrapy.extensions.logstats] INFO: Crawled 0 pages (at 0 pages/min), scraped 0 items (at 0 items/min)

2016-12-16 21:24:05 [scrapy.extensions.telnet] DEBUG: Telnet console listening on 127.0.0.1:6023

2016-12-16 21:24:05 [scrapy.core.engine] DEBUG: Crawled (404) <GET https://quotes.toscrape.com/robots.txt> (referer: **None**)

2016-12-16 21:24:05 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://quotes.toscrape.com/page/1/> (referer: **None**)

2016-12-16 21:24:05 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://quotes.toscrape.com/page/2/> (referer: **None**)

2016-12-16 21:24:05 [quotes] DEBUG: Saved file quotes-1.html

2016-12-16 21:24:05 [quotes] DEBUG: Saved file quotes-2.html

2016-12-16 21:24:05 [scrapy.core.engine] INFO: Closing spider (finished)

...

**후드 아래에서 무슨 일이?**

**start\_requests 메서드에 대한 바로 가기**

**import** **scrapy**

**class** **QuotesSpider**(scrapy.Spider):

name = "quotes"

start\_urls = [

'https://quotes.toscrape.com/page/1/',

'https://quotes.toscrape.com/page/2/',

]

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

page = response.url.split("/")[-2]

filename = f'quotes-*{*page*}*.html'

**with** open(filename, 'wb') **as** f:

f.write(response.body)

**데이터 추출**

scrapy shell 'https://quotes.toscrape.com/page/1/'

scrapy shell "https://quotes.toscrape.com/page/1/"

[ ... Scrapy log here ... ]

2016-09-19 12:09:27 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://quotes.toscrape.com/page/1/> (referer: **None**)

[s] Available Scrapy objects:

[s] scrapy scrapy module (contains scrapy.Request, scrapy.Selector, etc)

[s] crawler <scrapy.crawler.Crawler object at 0x7fa91d888c90>

[s] item {}

[s] request <GET https://quotes.toscrape.com/page/1/>

[s] response <200 https://quotes.toscrape.com/page/1/>

[s] settings <scrapy.settings.Settings object at 0x7fa91d888c10>

[s] spider <DefaultSpider 'default' at 0x7fa91c8af990>

[s] Useful shortcuts:

[s] shelp() Shell help (print this help)

[s] fetch(req\_or\_url) Fetch request (**or** URL) **and** update local objects

[s] view(response) View response **in** a browser

**>>>** response.css('noelement')[0].get()

Traceback (most recent call last):

*...*

IndexError: list index out of range

**>>>** response.css("noelement").get()

scrapy shell 'https://quotes.toscrape.com'

**XPath: 간단한 소개**

**인용문 및 저자 추출**

scrapy shell 'https://quotes.toscrape.com'

**스파이더에서 데이터 추출**

**import** **scrapy**

**class** **QuotesSpider**(scrapy.Spider):

name = "quotes"

start\_urls = [

'https://quotes.toscrape.com/page/1/',

'https://quotes.toscrape.com/page/2/',

]

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

**for** quote **in** response.css('div.quote'):

**yield** {

'text': quote.css('span.text::text').get(),

'author': quote.css('small.author::text').get(),

'tags': quote.css('div.tags a.tag::text').getall(),

}

2016-09-19 18:57:19 [scrapy.core.scraper] DEBUG: Scraped **from** <200 https://quotes.toscrape.com/page/1/>

{'tags': ['life', 'love'], 'author': 'André Gide', 'text': '“It is better to be hated for what you are than to be loved for what you are not.”'}

2016-09-19 18:57:19 [scrapy.core.scraper] DEBUG: Scraped **from** <200 https://quotes.toscrape.com/page/1/>

{'tags': ['edison', 'failure', 'inspirational', 'paraphrased'], 'author': 'Thomas A. Edison', 'text': "“I have not failed. I've just found 10,000 ways that won't work.”"}

**스크랩한 데이터 저장**

scrapy crawl quotes -O quotes.json

scrapy crawl quotes -o quotes.jsonl

**다음 링크**

**import** **scrapy**

**class** **QuotesSpider**(scrapy.Spider):

name = "quotes"

start\_urls = [

'https://quotes.toscrape.com/page/1/',

]

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

**for** quote **in** response.css('div.quote'):

**yield** {

'text': quote.css('span.text::text').get(),

'author': quote.css('small.author::text').get(),

'tags': quote.css('div.tags a.tag::text').getall(),

}

next\_page = response.css('li.next a::attr(href)').get()

**if** next\_page **is** **not** **None**:

next\_page = response.urljoin(next\_page)

**yield** scrapy.Request(next\_page, callback=self.parse)

**import** **scrapy**

**class** **QuotesSpider**(scrapy.Spider):

name = "quotes"

start\_urls = [

'https://quotes.toscrape.com/page/1/',

]

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

**for** quote **in** response.css('div.quote'):

**yield** {

'text': quote.css('span.text::text').get(),

'author': quote.css('span small::text').get(),

'tags': quote.css('div.tags a.tag::text').getall(),

}

next\_page = response.css('li.next a::attr(href)').get()

**if** next\_page **is** **not** **None**:

**yield** response.follow(next\_page, callback=self.parse)

**for** href **in** response.css('ul.pager a::attr(href)'):

**yield** response.follow(href, callback=self.parse)

**for** a **in** response.css('ul.pager a'):

**yield** response.follow(a, callback=self.parse)

anchors = response.css('ul.pager a')

**yield from** response.follow\_all(anchors, callback=self.parse)

**yield from** response.follow\_all(css='ul.pager a', callback=self.parse)

**import** **scrapy**

**class** **AuthorSpider**(scrapy.Spider):

name = 'author'

start\_urls = ['https://quotes.toscrape.com/']

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

author\_page\_links = response.css('.author + a')

**yield from** response.follow\_all(author\_page\_links, self.parse\_author)

pagination\_links = response.css('li.next a')

**yield from** response.follow\_all(pagination\_links, self.parse)

**def** parse\_author(self, response):

**def** extract\_with\_css(query):

**return** response.css(query).get(default='').strip()

**yield** {

'name': extract\_with\_css('h3.author-title::text'),

'birthdate': extract\_with\_css('.author-born-date::text'),

'bio': extract\_with\_css('.author-description::text'),

}

**요청 생성을 위한 바로 가기**

**import** **scrapy**

**class** **QuotesSpider**(scrapy.Spider):

name = "quotes"

start\_urls = [

'https://quotes.toscrape.com/page/1/',

]

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

**for** quote **in** response.css('div.quote'):

**yield** {

'text': quote.css('span.text::text').get(),

'author': quote.css('span small::text').get(),

'tags': quote.css('div.tags a.tag::text').getall(),

}

next\_page = response.css('li.next a::attr(href)').get()

**if** next\_page **is** **not** **None**:

**yield** response.follow(next\_page, callback=self.parse)

**for** href **in** response.css('ul.pager a::attr(href)'):

**yield** response.follow(href, callback=self.parse)

**for** a **in** response.css('ul.pager a'):

**yield** response.follow(a, callback=self.parse)

anchors = response.css('ul.pager a')

**yield from** response.follow\_all(anchors, callback=self.parse)

**yield from** response.follow\_all(css='ul.pager a', callback=self.parse)

**더 많은 예제와 패턴**

**import** **scrapy**

**class** **AuthorSpider**(scrapy.Spider):

name = 'author'

start\_urls = ['https://quotes.toscrape.com/']

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

author\_page\_links = response.css('.author + a')

**yield from** response.follow\_all(author\_page\_links, self.parse\_author)

pagination\_links = response.css('li.next a')

**yield from** response.follow\_all(pagination\_links, self.parse)

**def** parse\_author(self, response):

**def** extract\_with\_css(query):

**return** response.css(query).get(default='').strip()

**yield** {

'name': extract\_with\_css('h3.author-title::text'),

'birthdate': extract\_with\_css('.author-born-date::text'),

'bio': extract\_with\_css('.author-description::text'),

}

**스파이더 인수 사용**

scrapy crawl quotes -O quotes-humor.json -a tag=humor

**import** **scrapy**

**class** **QuotesSpider**(scrapy.Spider):

name = "quotes"

**def** start\_requests(self):

url = 'https://quotes.toscrape.com/'

tag = getattr(self, 'tag', **None**)

**if** tag **is** **not** **None**:

url = url + 'tag/' + tag

**yield** scrapy.Request(url, self.parse)

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

**for** quote **in** response.css('div.quote'):

**yield** {

'text': quote.css('span.text::text').get(),

'author': quote.css('small.author::text').get(),

}

next\_page = response.css('li.next a::attr(href)').get()

**if** next\_page **is** **not** **None**:

**yield** response.follow(next\_page, self.parse)

**다음 단계**

[**예**](https://docs.scrapy.org/en/latest/intro/examples.html)

[**명령줄 도구**](https://docs.scrapy.org/en/latest/topics/commands.html)

**구성 설정**

**Scrapy 프로젝트의 기본 구조**

**프로젝트 간 루트 디렉토리 공유**

**스크랩 도구 사용**

**프로젝트 만들기**

**프로젝트 제어**

**사용 가능한 도구 명령**

**시작 프로젝트**

**젠스파이더**

**기다**

**확인하다**

**목록**

**편집하다**

**술책**

**보다**

**껍데기**

**분석하다**

**설정**

**런스파이더**

**버전**

**벤치**

**사용자 정의 프로젝트 명령**

**COMMANDS\_MODULE**

**setup.py 진입점을 통해 명령 등록**

[**거미**](https://docs.scrapy.org/en/latest/topics/spiders.html)

**class** **MySpider**(scrapy.Spider):

name = 'myspider'

**def** start\_requests(self):

**return** [scrapy.FormRequest("http://www.example.com/login",

formdata={'user': 'john', 'pass': 'secret'},

callback=self.logged\_in)]

**def** logged\_in(self, response):

*# here you would extract links to follow and return Requests for*

*# each of them, with another callback*

**pass**

**import** **scrapy**

**class** **MySpider**(scrapy.Spider):

name = 'example.com'

allowed\_domains = ['example.com']

start\_urls = [

'http://www.example.com/1.html',

'http://www.example.com/2.html',

'http://www.example.com/3.html',

]

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

self.logger.info('A response from *%s* just arrived!', response.url)

**import** **scrapy**

**class** **MySpider**(scrapy.Spider):

name = 'example.com'

allowed\_domains = ['example.com']

start\_urls = [

'http://www.example.com/1.html',

'http://www.example.com/2.html',

'http://www.example.com/3.html',

]

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

**for** h3 **in** response.xpath('//h3').getall():

**yield** {"title": h3}

**for** href **in** response.xpath('//a/@href').getall():

**yield** scrapy.Request(response.urljoin(href), self.parse)

**import** **scrapy**

**from** **myproject.items** **import** MyItem

**class** **MySpider**(scrapy.Spider):

name = 'example.com'

allowed\_domains = ['example.com']

**def** start\_requests(self):

**yield** scrapy.Request('http://www.example.com/1.html', self.parse)

**yield** scrapy.Request('http://www.example.com/2.html', self.parse)

**yield** scrapy.Request('http://www.example.com/3.html', self.parse)

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

**for** h3 **in** response.xpath('//h3').getall():

**yield** MyItem(title=h3)

**for** href **in** response.xpath('//a/@href').getall():

**yield** scrapy.Request(response.urljoin(href), self.parse)

scrapy crawl myspider -a category=electronics

**import** **scrapy**

**class** **MySpider**(scrapy.Spider):

name = 'myspider'

**def** \_\_init\_\_(self, category=**None**, \*args, \*\*kwargs):

super(MySpider, self).\_\_init\_\_(\*args, \*\*kwargs)

self.start\_urls = [f'http://www.example.com/categories/*{*category*}*']

*# ...*

**import** **scrapy**

**class** **MySpider**(scrapy.Spider):

name = 'myspider'

**def** start\_requests(self):

**yield** scrapy.Request(f'http://www.example.com/categories/*{*self.category*}*')

process = CrawlerProcess()

process.crawl(MySpider, category="electronics")

scrapy crawl myspider -a http\_user=myuser -a http\_pass=mypassword -a user\_agent=mybot

**import** **scrapy**

**class** **TestItem**(scrapy.Item):

id = scrapy.Field()

name = scrapy.Field()

description = scrapy.Field()

**import** **scrapy**

**from** **scrapy.spiders** **import** CrawlSpider, Rule

**from** **scrapy.linkextractors** **import** LinkExtractor

**class** **MySpider**(CrawlSpider):

name = 'example.com'

allowed\_domains = ['example.com']

start\_urls = ['http://www.example.com']

rules = (

*# Extract links matching 'category.php' (but not matching 'subsection.php')*

*# and follow links from them (since no callback means follow=True by default).*

Rule(LinkExtractor(allow=('category\.php', ), deny=('subsection\.php', ))),

*# Extract links matching 'item.php' and parse them with the spider's method parse\_item*

Rule(LinkExtractor(allow=('item\.php', )), callback='parse\_item'),

)

**def** parse\_item(self, response):

self.logger.info('Hi, this is an item page! *%s*', response.url)

item = scrapy.Item()

item['id'] = response.xpath('//td[@id="item\_id"]/text()').re(r'ID: (\d+)')

item['name'] = response.xpath('//td[@id="item\_name"]/text()').get()

item['description'] = response.xpath('//td[@id="item\_description"]/text()').get()

item['link\_text'] = response.meta['link\_text']

url = response.xpath('//td[@id="additional\_data"]/@href').get()

**return** response.follow(url, self.parse\_additional\_page, cb\_kwargs=dict(item=item))

**def** parse\_additional\_page(self, response, item):

item['additional\_data'] = response.xpath('//p[@id="additional\_data"]/text()').get()

**return** item

itertag = 'product'

**class** **YourSpider**(XMLFeedSpider):

namespaces = [('n', 'http://www.sitemaps.org/schemas/sitemap/0.9')]

itertag = 'n:url'

*# ...*

**from** **scrapy.spiders** **import** XMLFeedSpider

**from** **myproject.items** **import** TestItem

**class** **MySpider**(XMLFeedSpider):

name = 'example.com'

allowed\_domains = ['example.com']

start\_urls = ['http://www.example.com/feed.xml']

iterator = 'iternodes' *# This is actually unnecessary, since it's the default value*

itertag = 'item'

**def** parse\_node(self, response, node):

self.logger.info('Hi, this is a <*%s*> node!: *%s*', self.itertag, ''.join(node.getall()))

item = TestItem()

item['id'] = node.xpath('@id').get()

item['name'] = node.xpath('name').get()

item['description'] = node.xpath('description').get()

**return** item

**from** **scrapy.spiders** **import** CSVFeedSpider

**from** **myproject.items** **import** TestItem

**class** **MySpider**(CSVFeedSpider):

name = 'example.com'

allowed\_domains = ['example.com']

start\_urls = ['http://www.example.com/feed.csv']

delimiter = ';'

quotechar = "'"

headers = ['id', 'name', 'description']

**def** parse\_row(self, response, row):

self.logger.info('Hi, this is a row!: *%r*', row)

item = TestItem()

item['id'] = row['id']

item['name'] = row['name']

item['description'] = row['description']

**return** item

sitemap\_rules = [('/product/', 'parse\_product')]

<url>

<loc>http://example.com/</loc>

<xhtml:link rel="alternate" hreflang="de" href="http://example.com/de"/>

</url>

<url>

<loc>http://example.com/</loc>

<lastmod>2005-01-01</lastmod>

</url>

**from** **datetime** **import** datetime

**from** **scrapy.spiders** **import** SitemapSpider

**class** **FilteredSitemapSpider**(SitemapSpider):

name = 'filtered\_sitemap\_spider'

allowed\_domains = ['example.com']

sitemap\_urls = ['http://example.com/sitemap.xml']

**def** sitemap\_filter(self, entries):

**for** entry **in** entries:

date\_time = datetime.strptime(entry['lastmod'], '%Y-%m-*%d*')

**if** date\_time.year >= 2005:

**yield** entry

**from** **scrapy.spiders** **import** SitemapSpider

**class** **MySpider**(SitemapSpider):

sitemap\_urls = ['http://www.example.com/sitemap.xml']

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

**pass** *# ... scrape item here ...*

**from** **scrapy.spiders** **import** SitemapSpider

**class** **MySpider**(SitemapSpider):

sitemap\_urls = ['http://www.example.com/sitemap.xml']

sitemap\_rules = [

('/product/', 'parse\_product'),

('/category/', 'parse\_category'),

]

**def** parse\_product(self, response):

**pass** *# ... scrape product ...*

**def** parse\_category(self, response):

**pass** *# ... scrape category ...*

**from** **scrapy.spiders** **import** SitemapSpider

**class** **MySpider**(SitemapSpider):

sitemap\_urls = ['http://www.example.com/robots.txt']

sitemap\_rules = [

('/shop/', 'parse\_shop'),

]

sitemap\_follow = ['/sitemap\_shops']

**def** parse\_shop(self, response):

**pass** *# ... scrape shop here ...*

**from** **scrapy.spiders** **import** SitemapSpider

**class** **MySpider**(SitemapSpider):

sitemap\_urls = ['http://www.example.com/robots.txt']

sitemap\_rules = [

('/shop/', 'parse\_shop'),

]

other\_urls = ['http://www.example.com/about']

**def** start\_requests(self):

requests = list(super(MySpider, self).start\_requests())

requests += [scrapy.Request(x, self.parse\_other) **for** x **in** self.other\_urls]

**return** requests

**def** parse\_shop(self, response):

**pass** *# ... scrape shop here ...*

**def** parse\_other(self, response):

**pass** *# ... scrape other here ...*

**스크래피.거미**

**class** **MySpider**(scrapy.Spider):

name = 'myspider'

**def** start\_requests(self):

**return** [scrapy.FormRequest("http://www.example.com/login",

formdata={'user': 'john', 'pass': 'secret'},

callback=self.logged\_in)]

**def** logged\_in(self, response):

*# here you would extract links to follow and return Requests for*

*# each of them, with another callback*

**pass**

**import** **scrapy**

**class** **MySpider**(scrapy.Spider):

name = 'example.com'

allowed\_domains = ['example.com']

start\_urls = [

'http://www.example.com/1.html',

'http://www.example.com/2.html',

'http://www.example.com/3.html',

]

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

self.logger.info('A response from *%s* just arrived!', response.url)

**import** **scrapy**

**class** **MySpider**(scrapy.Spider):

name = 'example.com'

allowed\_domains = ['example.com']

start\_urls = [

'http://www.example.com/1.html',

'http://www.example.com/2.html',

'http://www.example.com/3.html',

]

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

**for** h3 **in** response.xpath('//h3').getall():

**yield** {"title": h3}

**for** href **in** response.xpath('//a/@href').getall():

**yield** scrapy.Request(response.urljoin(href), self.parse)

**import** **scrapy**

**from** **myproject.items** **import** MyItem

**class** **MySpider**(scrapy.Spider):

name = 'example.com'

allowed\_domains = ['example.com']

**def** start\_requests(self):

**yield** scrapy.Request('http://www.example.com/1.html', self.parse)

**yield** scrapy.Request('http://www.example.com/2.html', self.parse)

**yield** scrapy.Request('http://www.example.com/3.html', self.parse)

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

**for** h3 **in** response.xpath('//h3').getall():

**yield** MyItem(title=h3)

**for** href **in** response.xpath('//a/@href').getall():

**yield** scrapy.Request(response.urljoin(href), self.parse)

**스파이더 인수**

scrapy crawl myspider -a category=electronics

**import** **scrapy**

**class** **MySpider**(scrapy.Spider):

name = 'myspider'

**def** \_\_init\_\_(self, category=**None**, \*args, \*\*kwargs):

super(MySpider, self).\_\_init\_\_(\*args, \*\*kwargs)

self.start\_urls = [f'http://www.example.com/categories/*{*category*}*']

*# ...*

**import** **scrapy**

**class** **MySpider**(scrapy.Spider):

name = 'myspider'

**def** start\_requests(self):

**yield** scrapy.Request(f'http://www.example.com/categories/*{*self.category*}*')

process = CrawlerProcess()

process.crawl(MySpider, category="electronics")

scrapy crawl myspider -a http\_user=myuser -a http\_pass=mypassword -a user\_agent=mybot

**일반 거미**

**import** **scrapy**

**class** **TestItem**(scrapy.Item):

id = scrapy.Field()

name = scrapy.Field()

description = scrapy.Field()

**import** **scrapy**

**from** **scrapy.spiders** **import** CrawlSpider, Rule

**from** **scrapy.linkextractors** **import** LinkExtractor

**class** **MySpider**(CrawlSpider):

name = 'example.com'

allowed\_domains = ['example.com']

start\_urls = ['http://www.example.com']

rules = (

*# Extract links matching 'category.php' (but not matching 'subsection.php')*

*# and follow links from them (since no callback means follow=True by default).*

Rule(LinkExtractor(allow=('category\.php', ), deny=('subsection\.php', ))),

*# Extract links matching 'item.php' and parse them with the spider's method parse\_item*

Rule(LinkExtractor(allow=('item\.php', )), callback='parse\_item'),

)

**def** parse\_item(self, response):

self.logger.info('Hi, this is an item page! *%s*', response.url)

item = scrapy.Item()

item['id'] = response.xpath('//td[@id="item\_id"]/text()').re(r'ID: (\d+)')

item['name'] = response.xpath('//td[@id="item\_name"]/text()').get()

item['description'] = response.xpath('//td[@id="item\_description"]/text()').get()

item['link\_text'] = response.meta['link\_text']

url = response.xpath('//td[@id="additional\_data"]/@href').get()

**return** response.follow(url, self.parse\_additional\_page, cb\_kwargs=dict(item=item))

**def** parse\_additional\_page(self, response, item):

item['additional\_data'] = response.xpath('//p[@id="additional\_data"]/text()').get()

**return** item

itertag = 'product'

**class** **YourSpider**(XMLFeedSpider):

namespaces = [('n', 'http://www.sitemaps.org/schemas/sitemap/0.9')]

itertag = 'n:url'

*# ...*

**from** **scrapy.spiders** **import** XMLFeedSpider

**from** **myproject.items** **import** TestItem

**class** **MySpider**(XMLFeedSpider):

name = 'example.com'

allowed\_domains = ['example.com']

start\_urls = ['http://www.example.com/feed.xml']

iterator = 'iternodes' *# This is actually unnecessary, since it's the default value*

itertag = 'item'

**def** parse\_node(self, response, node):

self.logger.info('Hi, this is a <*%s*> node!: *%s*', self.itertag, ''.join(node.getall()))

item = TestItem()

item['id'] = node.xpath('@id').get()

item['name'] = node.xpath('name').get()

item['description'] = node.xpath('description').get()

**return** item

**from** **scrapy.spiders** **import** CSVFeedSpider

**from** **myproject.items** **import** TestItem

**class** **MySpider**(CSVFeedSpider):

name = 'example.com'

allowed\_domains = ['example.com']

start\_urls = ['http://www.example.com/feed.csv']

delimiter = ';'

quotechar = "'"

headers = ['id', 'name', 'description']

**def** parse\_row(self, response, row):

self.logger.info('Hi, this is a row!: *%r*', row)

item = TestItem()

item['id'] = row['id']

item['name'] = row['name']

item['description'] = row['description']

**return** item

sitemap\_rules = [('/product/', 'parse\_product')]

<url>

<loc>http://example.com/</loc>

<xhtml:link rel="alternate" hreflang="de" href="http://example.com/de"/>

</url>

<url>

<loc>http://example.com/</loc>

<lastmod>2005-01-01</lastmod>

</url>

**from** **datetime** **import** datetime

**from** **scrapy.spiders** **import** SitemapSpider

**class** **FilteredSitemapSpider**(SitemapSpider):

name = 'filtered\_sitemap\_spider'

allowed\_domains = ['example.com']

sitemap\_urls = ['http://example.com/sitemap.xml']

**def** sitemap\_filter(self, entries):

**for** entry **in** entries:

date\_time = datetime.strptime(entry['lastmod'], '%Y-%m-*%d*')

**if** date\_time.year >= 2005:

**yield** entry

**from** **scrapy.spiders** **import** SitemapSpider

**class** **MySpider**(SitemapSpider):

sitemap\_urls = ['http://www.example.com/sitemap.xml']

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

**pass** *# ... scrape item here ...*

**from** **scrapy.spiders** **import** SitemapSpider

**class** **MySpider**(SitemapSpider):

sitemap\_urls = ['http://www.example.com/sitemap.xml']

sitemap\_rules = [

('/product/', 'parse\_product'),

('/category/', 'parse\_category'),

]

**def** parse\_product(self, response):

**pass** *# ... scrape product ...*

**def** parse\_category(self, response):

**pass** *# ... scrape category ...*

**from** **scrapy.spiders** **import** SitemapSpider

**class** **MySpider**(SitemapSpider):

sitemap\_urls = ['http://www.example.com/robots.txt']

sitemap\_rules = [

('/shop/', 'parse\_shop'),

]

sitemap\_follow = ['/sitemap\_shops']

**def** parse\_shop(self, response):

**pass** *# ... scrape shop here ...*

**from** **scrapy.spiders** **import** SitemapSpider

**class** **MySpider**(SitemapSpider):

sitemap\_urls = ['http://www.example.com/robots.txt']

sitemap\_rules = [

('/shop/', 'parse\_shop'),

]

other\_urls = ['http://www.example.com/about']

**def** start\_requests(self):

requests = list(super(MySpider, self).start\_requests())

requests += [scrapy.Request(x, self.parse\_other) **for** x **in** self.other\_urls]

**return** requests

**def** parse\_shop(self, response):

**pass** *# ... scrape shop here ...*

**def** parse\_other(self, response):

**pass** *# ... scrape other here ...*

**크롤스파이더**

**import** **scrapy**

**from** **scrapy.spiders** **import** CrawlSpider, Rule

**from** **scrapy.linkextractors** **import** LinkExtractor

**class** **MySpider**(CrawlSpider):

name = 'example.com'

allowed\_domains = ['example.com']

start\_urls = ['http://www.example.com']

rules = (

*# Extract links matching 'category.php' (but not matching 'subsection.php')*

*# and follow links from them (since no callback means follow=True by default).*

Rule(LinkExtractor(allow=('category\.php', ), deny=('subsection\.php', ))),

*# Extract links matching 'item.php' and parse them with the spider's method parse\_item*

Rule(LinkExtractor(allow=('item\.php', )), callback='parse\_item'),

)

**def** parse\_item(self, response):

self.logger.info('Hi, this is an item page! *%s*', response.url)

item = scrapy.Item()

item['id'] = response.xpath('//td[@id="item\_id"]/text()').re(r'ID: (\d+)')

item['name'] = response.xpath('//td[@id="item\_name"]/text()').get()

item['description'] = response.xpath('//td[@id="item\_description"]/text()').get()

item['link\_text'] = response.meta['link\_text']

url = response.xpath('//td[@id="additional\_data"]/@href').get()

**return** response.follow(url, self.parse\_additional\_page, cb\_kwargs=dict(item=item))

**def** parse\_additional\_page(self, response, item):

item['additional\_data'] = response.xpath('//p[@id="additional\_data"]/text()').get()

**return** item

**크롤링 규칙**

**CrawlSpider 예**

**import** **scrapy**

**from** **scrapy.spiders** **import** CrawlSpider, Rule

**from** **scrapy.linkextractors** **import** LinkExtractor

**class** **MySpider**(CrawlSpider):

name = 'example.com'

allowed\_domains = ['example.com']

start\_urls = ['http://www.example.com']

rules = (

*# Extract links matching 'category.php' (but not matching 'subsection.php')*

*# and follow links from them (since no callback means follow=True by default).*

Rule(LinkExtractor(allow=('category\.php', ), deny=('subsection\.php', ))),

*# Extract links matching 'item.php' and parse them with the spider's method parse\_item*

Rule(LinkExtractor(allow=('item\.php', )), callback='parse\_item'),

)

**def** parse\_item(self, response):

self.logger.info('Hi, this is an item page! *%s*', response.url)

item = scrapy.Item()

item['id'] = response.xpath('//td[@id="item\_id"]/text()').re(r'ID: (\d+)')

item['name'] = response.xpath('//td[@id="item\_name"]/text()').get()

item['description'] = response.xpath('//td[@id="item\_description"]/text()').get()

item['link\_text'] = response.meta['link\_text']

url = response.xpath('//td[@id="additional\_data"]/@href').get()

**return** response.follow(url, self.parse\_additional\_page, cb\_kwargs=dict(item=item))

**def** parse\_additional\_page(self, response, item):

item['additional\_data'] = response.xpath('//p[@id="additional\_data"]/text()').get()

**return** item

**XMLFeedSpider**

itertag = 'product'

**class** **YourSpider**(XMLFeedSpider):

namespaces = [('n', 'http://www.sitemaps.org/schemas/sitemap/0.9')]

itertag = 'n:url'

*# ...*

**from** **scrapy.spiders** **import** XMLFeedSpider

**from** **myproject.items** **import** TestItem

**class** **MySpider**(XMLFeedSpider):

name = 'example.com'

allowed\_domains = ['example.com']

start\_urls = ['http://www.example.com/feed.xml']

iterator = 'iternodes' *# This is actually unnecessary, since it's the default value*

itertag = 'item'

**def** parse\_node(self, response, node):

self.logger.info('Hi, this is a <*%s*> node!: *%s*', self.itertag, ''.join(node.getall()))

item = TestItem()

item['id'] = node.xpath('@id').get()

item['name'] = node.xpath('name').get()

item['description'] = node.xpath('description').get()

**return** item

**XMLFeedSpider 예**

**from** **scrapy.spiders** **import** XMLFeedSpider

**from** **myproject.items** **import** TestItem

**class** **MySpider**(XMLFeedSpider):

name = 'example.com'

allowed\_domains = ['example.com']

start\_urls = ['http://www.example.com/feed.xml']

iterator = 'iternodes' *# This is actually unnecessary, since it's the default value*

itertag = 'item'

**def** parse\_node(self, response, node):

self.logger.info('Hi, this is a <*%s*> node!: *%s*', self.itertag, ''.join(node.getall()))

item = TestItem()

item['id'] = node.xpath('@id').get()

item['name'] = node.xpath('name').get()

item['description'] = node.xpath('description').get()

**return** item

**CSVFeedSpider**

**from** **scrapy.spiders** **import** CSVFeedSpider

**from** **myproject.items** **import** TestItem

**class** **MySpider**(CSVFeedSpider):

name = 'example.com'

allowed\_domains = ['example.com']

start\_urls = ['http://www.example.com/feed.csv']

delimiter = ';'

quotechar = "'"

headers = ['id', 'name', 'description']

**def** parse\_row(self, response, row):

self.logger.info('Hi, this is a row!: *%r*', row)

item = TestItem()

item['id'] = row['id']

item['name'] = row['name']

item['description'] = row['description']

**return** item

**CSVFeedSpider 예**

**from** **scrapy.spiders** **import** CSVFeedSpider

**from** **myproject.items** **import** TestItem

**class** **MySpider**(CSVFeedSpider):

name = 'example.com'

allowed\_domains = ['example.com']

start\_urls = ['http://www.example.com/feed.csv']

delimiter = ';'

quotechar = "'"

headers = ['id', 'name', 'description']

**def** parse\_row(self, response, row):

self.logger.info('Hi, this is a row!: *%r*', row)

item = TestItem()

item['id'] = row['id']

item['name'] = row['name']

item['description'] = row['description']

**return** item

**사이트맵스파이더**

sitemap\_rules = [('/product/', 'parse\_product')]

<url>

<loc>http://example.com/</loc>

<xhtml:link rel="alternate" hreflang="de" href="http://example.com/de"/>

</url>

<url>

<loc>http://example.com/</loc>

<lastmod>2005-01-01</lastmod>

</url>

**from** **datetime** **import** datetime

**from** **scrapy.spiders** **import** SitemapSpider

**class** **FilteredSitemapSpider**(SitemapSpider):

name = 'filtered\_sitemap\_spider'

allowed\_domains = ['example.com']

sitemap\_urls = ['http://example.com/sitemap.xml']

**def** sitemap\_filter(self, entries):

**for** entry **in** entries:

date\_time = datetime.strptime(entry['lastmod'], '%Y-%m-*%d*')

**if** date\_time.year >= 2005:

**yield** entry

**from** **scrapy.spiders** **import** SitemapSpider

**class** **MySpider**(SitemapSpider):

sitemap\_urls = ['http://www.example.com/sitemap.xml']

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

**pass** *# ... scrape item here ...*

**from** **scrapy.spiders** **import** SitemapSpider

**class** **MySpider**(SitemapSpider):

sitemap\_urls = ['http://www.example.com/sitemap.xml']

sitemap\_rules = [

('/product/', 'parse\_product'),

('/category/', 'parse\_category'),

]

**def** parse\_product(self, response):

**pass** *# ... scrape product ...*

**def** parse\_category(self, response):

**pass** *# ... scrape category ...*

**from** **scrapy.spiders** **import** SitemapSpider

**class** **MySpider**(SitemapSpider):

sitemap\_urls = ['http://www.example.com/robots.txt']

sitemap\_rules = [

('/shop/', 'parse\_shop'),

]

sitemap\_follow = ['/sitemap\_shops']

**def** parse\_shop(self, response):

**pass** *# ... scrape shop here ...*

**from** **scrapy.spiders** **import** SitemapSpider

**class** **MySpider**(SitemapSpider):

sitemap\_urls = ['http://www.example.com/robots.txt']

sitemap\_rules = [

('/shop/', 'parse\_shop'),

]

other\_urls = ['http://www.example.com/about']

**def** start\_requests(self):

requests = list(super(MySpider, self).start\_requests())

requests += [scrapy.Request(x, self.parse\_other) **for** x **in** self.other\_urls]

**return** requests

**def** parse\_shop(self, response):

**pass** *# ... scrape shop here ...*

**def** parse\_other(self, response):

**pass** *# ... scrape other here ...*

**SitemapSpider 예**

**from** **scrapy.spiders** **import** SitemapSpider

**class** **MySpider**(SitemapSpider):

sitemap\_urls = ['http://www.example.com/sitemap.xml']

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

**pass** *# ... scrape item here ...*

**from** **scrapy.spiders** **import** SitemapSpider

**class** **MySpider**(SitemapSpider):

sitemap\_urls = ['http://www.example.com/sitemap.xml']

sitemap\_rules = [

('/product/', 'parse\_product'),

('/category/', 'parse\_category'),

]

**def** parse\_product(self, response):

**pass** *# ... scrape product ...*

**def** parse\_category(self, response):

**pass** *# ... scrape category ...*

**from** **scrapy.spiders** **import** SitemapSpider

**class** **MySpider**(SitemapSpider):

sitemap\_urls = ['http://www.example.com/robots.txt']

sitemap\_rules = [

('/shop/', 'parse\_shop'),

]

sitemap\_follow = ['/sitemap\_shops']

**def** parse\_shop(self, response):

**pass** *# ... scrape shop here ...*

**from** **scrapy.spiders** **import** SitemapSpider

**class** **MySpider**(SitemapSpider):

sitemap\_urls = ['http://www.example.com/robots.txt']

sitemap\_rules = [

('/shop/', 'parse\_shop'),

]

other\_urls = ['http://www.example.com/about']

**def** start\_requests(self):

requests = list(super(MySpider, self).start\_requests())

requests += [scrapy.Request(x, self.parse\_other) **for** x **in** self.other\_urls]

**return** requests

**def** parse\_shop(self, response):

**pass** *# ... scrape shop here ...*

**def** parse\_other(self, response):

**pass** *# ... scrape other here ...*

[**선택기**](https://docs.scrapy.org/en/latest/topics/selectors.html)

**선택기 사용**

**선택자 생성**

**선택기 사용**

**CSS 선택기 확장**

**중첩 선택기**

**요소 속성 선택**

**정규 표현식과 함께 선택기 사용**

**extract() 및 extract\_first()**

**XPath 작업**

**상대 XPath 작업**

**클래스별로 쿼리할 때 CSS 사용 고려**

**//node[1]과 (//node)[1]의 차이점에 주의하세요.**

**조건에서 텍스트 노드 사용**

**XPath 표현식의 변수**

**네임스페이스 제거**

**EXSLT 확장 사용**

**정규식**

**작업 설정**

**기타 XPath 확장**

**내장 선택기 참조**

**선택기 개체**

**SelectorList 객체**

**예**

**HTML 응답에 대한 선택기 예제**

**XML 응답에 대한 선택기 예제**

[**아이템**](https://docs.scrapy.org/en/latest/topics/items.html)

**from** **scrapy.item** **import** Item, Field

**class** **CustomItem**(Item):

one\_field = Field()

another\_field = Field()

**from** **dataclasses** **import** dataclass

**@dataclass**

**class** **CustomItem**:

one\_field: str

another\_field: int

**import** **attr**

**@attr**.s

**class** **CustomItem**:

one\_field = attr.ib()

another\_field = attr.ib()

**import** **scrapy**

**class** **Product**(scrapy.Item):

name = scrapy.Field()

price = scrapy.Field()

stock = scrapy.Field()

tags = scrapy.Field()

last\_updated = scrapy.Field(serializer=str)

**class** **DiscountedProduct**(Product):

discount\_percent = scrapy.Field(serializer=str)

discount\_expiration\_date = scrapy.Field()

**class** **SpecificProduct**(Product):

name = scrapy.Field(Product.fields['name'], serializer=my\_serializer)

**품목 유형**

**from** **scrapy.item** **import** Item, Field

**class** **CustomItem**(Item):

one\_field = Field()

another\_field = Field()

**from** **dataclasses** **import** dataclass

**@dataclass**

**class** **CustomItem**:

one\_field: str

another\_field: int

**import** **attr**

**@attr**.s

**class** **CustomItem**:

one\_field = attr.ib()

another\_field = attr.ib()

**사전**

**항목 개체**

**from** **scrapy.item** **import** Item, Field

**class** **CustomItem**(Item):

one\_field = Field()

another\_field = Field()

**데이터 클래스 객체**

**from** **dataclasses** **import** dataclass

**@dataclass**

**class** **CustomItem**:

one\_field: str

another\_field: int

**속성 개체**

**import** **attr**

**@attr**.s

**class** **CustomItem**:

one\_field = attr.ib()

another\_field = attr.ib()

**항목 개체 작업**

**import** **scrapy**

**class** **Product**(scrapy.Item):

name = scrapy.Field()

price = scrapy.Field()

stock = scrapy.Field()

tags = scrapy.Field()

last\_updated = scrapy.Field(serializer=str)

**class** **DiscountedProduct**(Product):

discount\_percent = scrapy.Field(serializer=str)

discount\_expiration\_date = scrapy.Field()

**class** **SpecificProduct**(Product):

name = scrapy.Field(Product.fields['name'], serializer=my\_serializer)

**아이템 서브클래스 선언하기**

**import** **scrapy**

**class** **Product**(scrapy.Item):

name = scrapy.Field()

price = scrapy.Field()

stock = scrapy.Field()

tags = scrapy.Field()

last\_updated = scrapy.Field(serializer=str)

**필드 선언**

**항목 개체 작업**

**아이템 생성**

**필드 값 가져오기**

**필드 값 설정**

**채워진 모든 값에 액세스**

**항목 복사**

**기타 일반적인 작업**

**아이템 서브클래스 확장**

**class** **DiscountedProduct**(Product):

discount\_percent = scrapy.Field(serializer=str)

discount\_expiration\_date = scrapy.Field()

**class** **SpecificProduct**(Product):

name = scrapy.Field(Product.fields['name'], serializer=my\_serializer)

**모든 항목 유형 지원**

**아이템과 관련된 다른 클래스**

[**아이템 로더**](https://docs.scrapy.org/en/latest/topics/loaders.html)

**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()

**from** **dataclasses** **import** dataclass, field

**from** **typing** **import** Optional

**@dataclass**

**class** **InventoryItem**:

name: Optional[str] = field(default=**None**)

price: Optional[float] = field(default=**None**)

stock: Optional[int] = field(default=**None**)

l = ItemLoader(Product(), some\_selector)

l.add\_xpath('name', xpath1) *# (1)*

l.add\_xpath('name', xpath2) *# (2)*

l.add\_css('name', css) *# (3)*

l.add\_value('name', 'test') *# (4)*

**return** l.load\_item() *# (5)*

**from** **itemloaders.processors** **import** TakeFirst, MapCompose, Join

**from** **scrapy.loader** **import** ItemLoader

**class** **ProductLoader**(ItemLoader):

default\_output\_processor = TakeFirst()

name\_in = MapCompose(str.title)

name\_out = Join()

price\_in = MapCompose(str.strip)

*# ...*

**import** **scrapy**

**from** **itemloaders.processors** **import** Join, MapCompose, TakeFirst

**from** **w3lib.html** **import** remove\_tags

**def** filter\_price(value):

**if** value.isdigit():

**return** value

**class** **Product**(scrapy.Item):

name = scrapy.Field(

input\_processor=MapCompose(remove\_tags),

output\_processor=Join(),

)

price = scrapy.Field(

input\_processor=MapCompose(remove\_tags, filter\_price),

output\_processor=TakeFirst(),

)

**def** parse\_length(text, loader\_context):

unit = loader\_context.get('unit', 'm')

*# ... length parsing code goes here ...*

**return** parsed\_length

loader = ItemLoader(product)

loader.context['unit'] = 'cm'

loader = ItemLoader(product, unit='cm')

**class** **ProductLoader**(ItemLoader):

length\_out = MapCompose(parse\_length, unit='cm')

*# HTML snippet: <p class="product-name">Color TV</p>*

loader.add\_css('name', 'p.product-name')

*# HTML snippet: <p id="price">the price is $1200</p>*

loader.add\_css('price', 'p#price', re='the price is (.\*)')

loader.add\_value('name', 'Color TV')

loader.add\_value('colours', ['white', 'blue'])

loader.add\_value('length', '100')

loader.add\_value('name', 'name: foo', TakeFirst(), re='name: (.+)')

loader.add\_value(**None**, {'name': 'foo', 'sex': 'male'})

*# HTML snippet: <p class="product-name">Color TV</p>*

loader.add\_xpath('name', '//p[@class="product-name"]')

*# HTML snippet: <p id="price">the price is $1200</p>*

loader.add\_xpath('price', '//p[@id="price"]', re='the price is (.\*)')

*# HTML snippet: <p class="product-name">Color TV</p>*

loader.get\_css('p.product-name')

*# HTML snippet: <p id="price">the price is $1200</p>*

loader.get\_css('p#price', TakeFirst(), re='the price is (.\*)')

*# HTML snippet: <p class="product-name">Color TV</p>*

loader.get\_xpath('//p[@class="product-name"]')

*# HTML snippet: <p id="price">the price is $1200</p>*

loader.get\_xpath('//p[@id="price"]', TakeFirst(), re='the price is (.\*)')

<footer>

<a class="social" href="https://facebook.com/whatever">Like Us</a>

<a class="social" href="https://twitter.com/whatever">Follow Us</a>

<a class="email" href="mailto:whatever@example.com">Email Us</a>

</footer>

loader = ItemLoader(item=Item())

*# load stuff not in the footer*

loader.add\_xpath('social', '//footer/a[@class = "social"]/@href')

loader.add\_xpath('email', '//footer/a[@class = "email"]/@href')

loader.load\_item()

loader = ItemLoader(item=Item())

*# load stuff not in the footer*

footer\_loader = loader.nested\_xpath('//footer')

footer\_loader.add\_xpath('social', 'a[@class = "social"]/@href')

footer\_loader.add\_xpath('email', 'a[@class = "email"]/@href')

*# no need to call footer\_loader.load\_item()*

loader.load\_item()

**from** **itemloaders.processors** **import** MapCompose

**from** **myproject.ItemLoaders** **import** ProductLoader

**def** strip\_dashes(x):

**return** x.strip('-')

**class** **SiteSpecificLoader**(ProductLoader):

name\_in = MapCompose(strip\_dashes, ProductLoader.name\_in)

**from** **itemloaders.processors** **import** MapCompose

**from** **myproject.ItemLoaders** **import** ProductLoader

**from** **myproject.utils.xml** **import** remove\_cdata

**class** **XmlProductLoader**(ProductLoader):

name\_in = MapCompose(remove\_cdata, ProductLoader.name\_in)

**항목 로더를 사용하여 항목 채우기**

**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()

**데이터 클래스 항목 작업**

**from** **dataclasses** **import** dataclass, field

**from** **typing** **import** Optional

**@dataclass**

**class** **InventoryItem**:

name: Optional[str] = field(default=**None**)

price: Optional[float] = field(default=**None**)

stock: Optional[int] = field(default=**None**)

**입력 및 출력 프로세서**

l = ItemLoader(Product(), some\_selector)

l.add\_xpath('name', xpath1) *# (1)*

l.add\_xpath('name', xpath2) *# (2)*

l.add\_css('name', css) *# (3)*

l.add\_value('name', 'test') *# (4)*

**return** l.load\_item() *# (5)*

**아이템 로더 선언**

**from** **itemloaders.processors** **import** TakeFirst, MapCompose, Join

**from** **scrapy.loader** **import** ItemLoader

**class** **ProductLoader**(ItemLoader):

default\_output\_processor = TakeFirst()

name\_in = MapCompose(str.title)

name\_out = Join()

price\_in = MapCompose(str.strip)

*# ...*

**입력 및 출력 프로세서 선언**

**import** **scrapy**

**from** **itemloaders.processors** **import** Join, MapCompose, TakeFirst

**from** **w3lib.html** **import** remove\_tags

**def** filter\_price(value):

**if** value.isdigit():

**return** value

**class** **Product**(scrapy.Item):

name = scrapy.Field(

input\_processor=MapCompose(remove\_tags),

output\_processor=Join(),

)

price = scrapy.Field(

input\_processor=MapCompose(remove\_tags, filter\_price),

output\_processor=TakeFirst(),

)

**항목 로더 컨텍스트**

**def** parse\_length(text, loader\_context):

unit = loader\_context.get('unit', 'm')

*# ... length parsing code goes here ...*

**return** parsed\_length

loader = ItemLoader(product)

loader.context['unit'] = 'cm'

loader = ItemLoader(product, unit='cm')

**class** **ProductLoader**(ItemLoader):

length\_out = MapCompose(parse\_length, unit='cm')

**ItemLoader 객체**

*# HTML snippet: <p class="product-name">Color TV</p>*

loader.add\_css('name', 'p.product-name')

*# HTML snippet: <p id="price">the price is $1200</p>*

loader.add\_css('price', 'p#price', re='the price is (.\*)')

loader.add\_value('name', 'Color TV')

loader.add\_value('colours', ['white', 'blue'])

loader.add\_value('length', '100')

loader.add\_value('name', 'name: foo', TakeFirst(), re='name: (.+)')

loader.add\_value(**None**, {'name': 'foo', 'sex': 'male'})

*# HTML snippet: <p class="product-name">Color TV</p>*

loader.add\_xpath('name', '//p[@class="product-name"]')

*# HTML snippet: <p id="price">the price is $1200</p>*

loader.add\_xpath('price', '//p[@id="price"]', re='the price is (.\*)')

*# HTML snippet: <p class="product-name">Color TV</p>*

loader.get\_css('p.product-name')

*# HTML snippet: <p id="price">the price is $1200</p>*

loader.get\_css('p#price', TakeFirst(), re='the price is (.\*)')

*# HTML snippet: <p class="product-name">Color TV</p>*

loader.get\_xpath('//p[@class="product-name"]')

*# HTML snippet: <p id="price">the price is $1200</p>*

loader.get\_xpath('//p[@id="price"]', TakeFirst(), re='the price is (.\*)')

**중첩 로더**

<footer>

<a class="social" href="https://facebook.com/whatever">Like Us</a>

<a class="social" href="https://twitter.com/whatever">Follow Us</a>

<a class="email" href="mailto:whatever@example.com">Email Us</a>

</footer>

loader = ItemLoader(item=Item())

*# load stuff not in the footer*

loader.add\_xpath('social', '//footer/a[@class = "social"]/@href')

loader.add\_xpath('email', '//footer/a[@class = "email"]/@href')

loader.load\_item()

loader = ItemLoader(item=Item())

*# load stuff not in the footer*

footer\_loader = loader.nested\_xpath('//footer')

footer\_loader.add\_xpath('social', 'a[@class = "social"]/@href')

footer\_loader.add\_xpath('email', 'a[@class = "email"]/@href')

*# no need to call footer\_loader.load\_item()*

loader.load\_item()

**아이템 로더 재사용 및 확장**

**from** **itemloaders.processors** **import** MapCompose

**from** **myproject.ItemLoaders** **import** ProductLoader

**def** strip\_dashes(x):

**return** x.strip('-')

**class** **SiteSpecificLoader**(ProductLoader):

name\_in = MapCompose(strip\_dashes, ProductLoader.name\_in)

**from** **itemloaders.processors** **import** MapCompose

**from** **myproject.ItemLoaders** **import** ProductLoader

**from** **myproject.utils.xml** **import** remove\_cdata

**class** **XmlProductLoader**(ProductLoader):

name\_in = MapCompose(remove\_cdata, ProductLoader.name\_in)

[**스크랩 쉘**](https://docs.scrapy.org/en/latest/topics/shell.html)

[settings]

shell = bpython

scrapy shell <url>

*# UNIX-style*

scrapy shell ./path/to/file.html

scrapy shell ../other/path/to/file.html

scrapy shell /absolute/path/to/file.html

*# File URI*

scrapy shell file:///absolute/path/to/file.html

$ scrapy shell index.html

[ ... scrapy shell starts ... ]

[ ... traceback ... ]

twisted.internet.error.DNSLookupError: DNS lookup failed:

address 'index.html' not found: [Errno -5] No address associated with hostname.

scrapy shell 'https://scrapy.org' --nolog

scrapy shell "https://scrapy.org" --nolog

[s] Available Scrapy objects:

[s] scrapy scrapy module (contains scrapy.Request, scrapy.Selector, etc)

[s] crawler <scrapy.crawler.Crawler object at 0x7f07395dd690>

[s] item {}

[s] request <GET https://scrapy.org>

[s] response <200 https://scrapy.org/>

[s] settings <scrapy.settings.Settings object at 0x7f07395dd710>

[s] spider <DefaultSpider 'default' at 0x7f0735891690>

[s] Useful shortcuts:

[s] fetch(url[, redirect=**True**]) Fetch URL **and** update local objects (by default, redirects are followed)

[s] fetch(req) Fetch a scrapy.Request **and** update local objects

[s] shelp() Shell help (print this help)

[s] view(response) View response **in** a browser

>>>

**import** **scrapy**

**class** **MySpider**(scrapy.Spider):

name = "myspider"

start\_urls = [

"http://example.com",

"http://example.org",

"http://example.net",

]

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

*# We want to inspect one specific response.*

**if** ".org" **in** response.url:

**from** **scrapy.shell** **import** inspect\_response

inspect\_response(response, self)

*# Rest of parsing code.*

2014-01-23 17:48:31-0400 [scrapy.core.engine] DEBUG: Crawled (200) <GET http://example.com> (referer: **None**)

2014-01-23 17:48:31-0400 [scrapy.core.engine] DEBUG: Crawled (200) <GET http://example.org> (referer: **None**)

[s] Available Scrapy objects:

[s] crawler <scrapy.crawler.Crawler object at 0x1e16b50>

...

>>> response.url

'http://example.org'

**>>>** ^D

2014-01-23 17:50:03-0400 [scrapy.core.engine] DEBUG: Crawled (200) <GET http://example.net> (referer: None)

**...**

**셸 구성**

[settings]

shell = bpython

**셸 실행**

scrapy shell <url>

*# UNIX-style*

scrapy shell ./path/to/file.html

scrapy shell ../other/path/to/file.html

scrapy shell /absolute/path/to/file.html

*# File URI*

scrapy shell file:///absolute/path/to/file.html

$ scrapy shell index.html

[ ... scrapy shell starts ... ]

[ ... traceback ... ]

twisted.internet.error.DNSLookupError: DNS lookup failed:

address 'index.html' not found: [Errno -5] No address associated with hostname.

**쉘 사용하기**

**사용 가능한 바로 가기**

**사용 가능한 스크랩 개체**

**쉘 세션의 예**

scrapy shell 'https://scrapy.org' --nolog

scrapy shell "https://scrapy.org" --nolog

[s] Available Scrapy objects:

[s] scrapy scrapy module (contains scrapy.Request, scrapy.Selector, etc)

[s] crawler <scrapy.crawler.Crawler object at 0x7f07395dd690>

[s] item {}

[s] request <GET https://scrapy.org>

[s] response <200 https://scrapy.org/>

[s] settings <scrapy.settings.Settings object at 0x7f07395dd710>

[s] spider <DefaultSpider 'default' at 0x7f0735891690>

[s] Useful shortcuts:

[s] fetch(url[, redirect=**True**]) Fetch URL **and** update local objects (by default, redirects are followed)

[s] fetch(req) Fetch a scrapy.Request **and** update local objects

[s] shelp() Shell help (print this help)

[s] view(response) View response **in** a browser

>>>

**스파이더에서 쉘을 호출하여 응답 검사**

**import** **scrapy**

**class** **MySpider**(scrapy.Spider):

name = "myspider"

start\_urls = [

"http://example.com",

"http://example.org",

"http://example.net",

]

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

*# We want to inspect one specific response.*

**if** ".org" **in** response.url:

**from** **scrapy.shell** **import** inspect\_response

inspect\_response(response, self)

*# Rest of parsing code.*

2014-01-23 17:48:31-0400 [scrapy.core.engine] DEBUG: Crawled (200) <GET http://example.com> (referer: **None**)

2014-01-23 17:48:31-0400 [scrapy.core.engine] DEBUG: Crawled (200) <GET http://example.org> (referer: **None**)

[s] Available Scrapy objects:

[s] crawler <scrapy.crawler.Crawler object at 0x1e16b50>

...

>>> response.url

'http://example.org'

**>>>** ^D

2014-01-23 17:50:03-0400 [scrapy.core.engine] DEBUG: Crawled (200) <GET http://example.net> (referer: None)

**...**

[**아이템 파이프라인**](https://docs.scrapy.org/en/latest/topics/item-pipeline.html)

**from** **itemadapter** **import** ItemAdapter

**from** **scrapy.exceptions** **import** DropItem

**class** **PricePipeline**:

vat\_factor = 1.15

**def** process\_item(self, item, spider):

adapter = ItemAdapter(item)

**if** adapter.get('price'):

**if** adapter.get('price\_excludes\_vat'):

adapter['price'] = adapter['price'] \* self.vat\_factor

**return** item

**else**:

**raise** DropItem(f"Missing price in *{*item*}*")

**import** **json**

**from** **itemadapter** **import** ItemAdapter

**class** **JsonWriterPipeline**:

**def** open\_spider(self, spider):

self.file = open('items.jsonl', 'w')

**def** close\_spider(self, spider):

self.file.close()

**def** process\_item(self, item, spider):

line = json.dumps(ItemAdapter(item).asdict()) + "**\n**"

self.file.write(line)

**return** item

**import** **pymongo**

**from** **itemadapter** **import** ItemAdapter

**class** **MongoPipeline**:

collection\_name = 'scrapy\_items'

**def** \_\_init\_\_(self, mongo\_uri, mongo\_db):

self.mongo\_uri = mongo\_uri

self.mongo\_db = mongo\_db

**@classmethod**

**def** from\_crawler(cls, crawler):

**return** cls(

mongo\_uri=crawler.settings.get('MONGO\_URI'),

mongo\_db=crawler.settings.get('MONGO\_DATABASE', 'items')

)

**def** open\_spider(self, spider):

self.client = pymongo.MongoClient(self.mongo\_uri)

self.db = self.client[self.mongo\_db]

**def** close\_spider(self, spider):

self.client.close()

**def** process\_item(self, item, spider):

self.db[self.collection\_name].insert\_one(ItemAdapter(item).asdict())

**return** item

**import** **hashlib**

**from** **urllib.parse** **import** quote

**import** **scrapy**

**from** **itemadapter** **import** ItemAdapter

**from** **scrapy.utils.defer** **import** maybe\_deferred\_to\_future

**class** **ScreenshotPipeline**:

*"""Pipeline that uses Splash to render screenshot of*

*every Scrapy item."""*

SPLASH\_URL = "http://localhost:8050/render.png?url=*{}*"

**async** **def** process\_item(self, item, spider):

adapter = ItemAdapter(item)

encoded\_item\_url = quote(adapter["url"])

screenshot\_url = self.SPLASH\_URL.format(encoded\_item\_url)

request = scrapy.Request(screenshot\_url)

response = **await** maybe\_deferred\_to\_future(spider.crawler.engine.download(request, spider))

**if** response.status != 200:

*# Error happened, return item.*

**return** item

*# Save screenshot to file, filename will be hash of url.*

url = adapter["url"]

url\_hash = hashlib.md5(url.encode("utf8")).hexdigest()

filename = f"*{*url\_hash*}*.png"

**with** open(filename, "wb") **as** f:

f.write(response.body)

*# Store filename in item.*

adapter["screenshot\_filename"] = filename

**return** item

**from** **itemadapter** **import** ItemAdapter

**from** **scrapy.exceptions** **import** DropItem

**class** **DuplicatesPipeline**:

**def** \_\_init\_\_(self):

self.ids\_seen = set()

**def** process\_item(self, item, spider):

adapter = ItemAdapter(item)

**if** adapter['id'] **in** self.ids\_seen:

**raise** DropItem(f"Duplicate item found: *{*item*!r}*")

**else**:

self.ids\_seen.add(adapter['id'])

**return** item

ITEM\_PIPELINES = {

'myproject.pipelines.PricePipeline': 300,

'myproject.pipelines.JsonWriterPipeline': 800,

}

**나만의 아이템 파이프라인 작성**

**항목 파이프라인 예**

**from** **itemadapter** **import** ItemAdapter

**from** **scrapy.exceptions** **import** DropItem

**class** **PricePipeline**:

vat\_factor = 1.15

**def** process\_item(self, item, spider):

adapter = ItemAdapter(item)

**if** adapter.get('price'):

**if** adapter.get('price\_excludes\_vat'):

adapter['price'] = adapter['price'] \* self.vat\_factor

**return** item

**else**:

**raise** DropItem(f"Missing price in *{*item*}*")

**import** **json**

**from** **itemadapter** **import** ItemAdapter

**class** **JsonWriterPipeline**:

**def** open\_spider(self, spider):

self.file = open('items.jsonl', 'w')

**def** close\_spider(self, spider):

self.file.close()

**def** process\_item(self, item, spider):

line = json.dumps(ItemAdapter(item).asdict()) + "**\n**"

self.file.write(line)

**return** item

**import** **pymongo**

**from** **itemadapter** **import** ItemAdapter

**class** **MongoPipeline**:

collection\_name = 'scrapy\_items'

**def** \_\_init\_\_(self, mongo\_uri, mongo\_db):

self.mongo\_uri = mongo\_uri

self.mongo\_db = mongo\_db

**@classmethod**

**def** from\_crawler(cls, crawler):

**return** cls(

mongo\_uri=crawler.settings.get('MONGO\_URI'),

mongo\_db=crawler.settings.get('MONGO\_DATABASE', 'items')

)

**def** open\_spider(self, spider):

self.client = pymongo.MongoClient(self.mongo\_uri)

self.db = self.client[self.mongo\_db]

**def** close\_spider(self, spider):

self.client.close()

**def** process\_item(self, item, spider):

self.db[self.collection\_name].insert\_one(ItemAdapter(item).asdict())

**return** item

**import** **hashlib**

**from** **urllib.parse** **import** quote

**import** **scrapy**

**from** **itemadapter** **import** ItemAdapter

**from** **scrapy.utils.defer** **import** maybe\_deferred\_to\_future

**class** **ScreenshotPipeline**:

*"""Pipeline that uses Splash to render screenshot of*

*every Scrapy item."""*

SPLASH\_URL = "http://localhost:8050/render.png?url=*{}*"

**async** **def** process\_item(self, item, spider):

adapter = ItemAdapter(item)

encoded\_item\_url = quote(adapter["url"])

screenshot\_url = self.SPLASH\_URL.format(encoded\_item\_url)

request = scrapy.Request(screenshot\_url)

response = **await** maybe\_deferred\_to\_future(spider.crawler.engine.download(request, spider))

**if** response.status != 200:

*# Error happened, return item.*

**return** item

*# Save screenshot to file, filename will be hash of url.*

url = adapter["url"]

url\_hash = hashlib.md5(url.encode("utf8")).hexdigest()

filename = f"*{*url\_hash*}*.png"

**with** open(filename, "wb") **as** f:

f.write(response.body)

*# Store filename in item.*

adapter["screenshot\_filename"] = filename

**return** item

**from** **itemadapter** **import** ItemAdapter

**from** **scrapy.exceptions** **import** DropItem

**class** **DuplicatesPipeline**:

**def** \_\_init\_\_(self):

self.ids\_seen = set()

**def** process\_item(self, item, spider):

adapter = ItemAdapter(item)

**if** adapter['id'] **in** self.ids\_seen:

**raise** DropItem(f"Duplicate item found: *{*item*!r}*")

**else**:

self.ids\_seen.add(adapter['id'])

**return** item

**가격 확인 및 가격 없는 항목 삭제**

**from** **itemadapter** **import** ItemAdapter

**from** **scrapy.exceptions** **import** DropItem

**class** **PricePipeline**:

vat\_factor = 1.15

**def** process\_item(self, item, spider):

adapter = ItemAdapter(item)

**if** adapter.get('price'):

**if** adapter.get('price\_excludes\_vat'):

adapter['price'] = adapter['price'] \* self.vat\_factor

**return** item

**else**:

**raise** DropItem(f"Missing price in *{*item*}*")

**JSON 라인 파일에 항목 쓰기**

**import** **json**

**from** **itemadapter** **import** ItemAdapter

**class** **JsonWriterPipeline**:

**def** open\_spider(self, spider):

self.file = open('items.jsonl', 'w')

**def** close\_spider(self, spider):

self.file.close()

**def** process\_item(self, item, spider):

line = json.dumps(ItemAdapter(item).asdict()) + "**\n**"

self.file.write(line)

**return** item

**MongoDB에 항목 쓰기**

**import** **pymongo**

**from** **itemadapter** **import** ItemAdapter

**class** **MongoPipeline**:

collection\_name = 'scrapy\_items'

**def** \_\_init\_\_(self, mongo\_uri, mongo\_db):

self.mongo\_uri = mongo\_uri

self.mongo\_db = mongo\_db

**@classmethod**

**def** from\_crawler(cls, crawler):

**return** cls(

mongo\_uri=crawler.settings.get('MONGO\_URI'),

mongo\_db=crawler.settings.get('MONGO\_DATABASE', 'items')

)

**def** open\_spider(self, spider):

self.client = pymongo.MongoClient(self.mongo\_uri)

self.db = self.client[self.mongo\_db]

**def** close\_spider(self, spider):

self.client.close()

**def** process\_item(self, item, spider):

self.db[self.collection\_name].insert\_one(ItemAdapter(item).asdict())

**return** item

**항목의 스크린샷 찍기**

**import** **hashlib**

**from** **urllib.parse** **import** quote

**import** **scrapy**

**from** **itemadapter** **import** ItemAdapter

**from** **scrapy.utils.defer** **import** maybe\_deferred\_to\_future

**class** **ScreenshotPipeline**:

*"""Pipeline that uses Splash to render screenshot of*

*every Scrapy item."""*

SPLASH\_URL = "http://localhost:8050/render.png?url=*{}*"

**async** **def** process\_item(self, item, spider):

adapter = ItemAdapter(item)

encoded\_item\_url = quote(adapter["url"])

screenshot\_url = self.SPLASH\_URL.format(encoded\_item\_url)

request = scrapy.Request(screenshot\_url)

response = **await** maybe\_deferred\_to\_future(spider.crawler.engine.download(request, spider))

**if** response.status != 200:

*# Error happened, return item.*

**return** item

*# Save screenshot to file, filename will be hash of url.*

url = adapter["url"]

url\_hash = hashlib.md5(url.encode("utf8")).hexdigest()

filename = f"*{*url\_hash*}*.png"

**with** open(filename, "wb") **as** f:

f.write(response.body)

*# Store filename in item.*

adapter["screenshot\_filename"] = filename

**return** item

**중복 필터**

**from** **itemadapter** **import** ItemAdapter

**from** **scrapy.exceptions** **import** DropItem

**class** **DuplicatesPipeline**:

**def** \_\_init\_\_(self):

self.ids\_seen = set()

**def** process\_item(self, item, spider):

adapter = ItemAdapter(item)

**if** adapter['id'] **in** self.ids\_seen:

**raise** DropItem(f"Duplicate item found: *{*item*!r}*")

**else**:

self.ids\_seen.add(adapter['id'])

**return** item

**항목 파이프라인 구성 요소 활성화**

ITEM\_PIPELINES = {

'myproject.pipelines.PricePipeline': 300,

'myproject.pipelines.JsonWriterPipeline': 800,

}

[**피드 내보내기**](https://docs.scrapy.org/en/latest/topics/feed-exports.html)

**class** **MyCustomFilter**:

**def** \_\_init\_\_(self, feed\_options):

self.feed\_options = feed\_options

**def** accepts(self, item):

**if** "field1" **in** item **and** item["field1"] == "expected\_data":

**return** **True**

**return** **False**

{

'items.json': {

'format': 'json',

'encoding': 'utf8',

'store\_empty': **False**,

'item\_classes': [MyItemClass1, 'myproject.items.MyItemClass2'],

'fields': **None**,

'indent': 4,

'item\_export\_kwargs': {

'export\_empty\_fields': **True**,

},

},

'/home/user/documents/items.xml': {

'format': 'xml',

'fields': ['name', 'price'],

'item\_filter': MyCustomFilter1,

'encoding': 'latin1',

'indent': 8,

},

pathlib.Path('items.csv.gz'): {

'format': 'csv',

'fields': ['price', 'name'],

'item\_filter': 'myproject.filters.MyCustomFilter2',

'postprocessing': [MyPlugin1, 'scrapy.extensions.postprocessing.GzipPlugin'],

'gzip\_compresslevel': 5,

},

}

{

'': 'scrapy.extensions.feedexport.FileFeedStorage',

'file': 'scrapy.extensions.feedexport.FileFeedStorage',

'stdout': 'scrapy.extensions.feedexport.StdoutFeedStorage',

's3': 'scrapy.extensions.feedexport.S3FeedStorage',

'ftp': 'scrapy.extensions.feedexport.FTPFeedStorage',

}

FEED\_STORAGES = {

'ftp': **None**,

}

{

'json': 'scrapy.exporters.JsonItemExporter',

'jsonlines': 'scrapy.exporters.JsonLinesItemExporter',

'jsonl': 'scrapy.exporters.JsonLinesItemExporter',

'jl': 'scrapy.exporters.JsonLinesItemExporter',

'csv': 'scrapy.exporters.CsvItemExporter',

'xml': 'scrapy.exporters.XmlItemExporter',

'marshal': 'scrapy.exporters.MarshalItemExporter',

'pickle': 'scrapy.exporters.PickleItemExporter',

}

FEED\_EXPORTERS = {

'csv': **None**,

}

FEED\_EXPORT\_BATCH\_ITEM\_COUNT = 100

scrapy crawl spidername -o "dirname/*%(batch\_id)d*-filename*%(batch\_time)s*.json"

->projectname

-->dirname

--->1-filename2020-03-28T14-45-08.237134.json

--->2-filename2020-03-28T14-45-09.148903.json

--->3-filename2020-03-28T14-45-10.046092.json

*# myproject/utils.py*

**def** uri\_params(params, spider):

**return** {\*\*params, 'spider\_name': spider.name}

*# myproject/settings.py*

FEED\_URI\_PARAMS = 'myproject.utils.uri\_params'

scrapy crawl <spider\_name> -o "*%(spider\_name)s*.jsonl"

**직렬화 형식**

**JSON**

**JSON 라인**

**CSV**

**XML**

**간물**

**육군 원수**

**창고**

**스토리지 URI 매개변수**

**스토리지 백엔드**

**로컬 파일 시스템**

**FTP**

**시즌3**

**Google 클라우드 스토리지(GCS)**

**표준 출력**

**지연된 파일 전달**

**항목 필터링**

**class** **MyCustomFilter**:

**def** \_\_init\_\_(self, feed\_options):

self.feed\_options = feed\_options

**def** accepts(self, item):

**if** "field1" **in** item **and** item["field1"] == "expected\_data":

**return** **True**

**return** **False**

**항목 필터**

**후처리**

**내장 플러그인**

**커스텀 플러그인**

**설정**

{

'items.json': {

'format': 'json',

'encoding': 'utf8',

'store\_empty': **False**,

'item\_classes': [MyItemClass1, 'myproject.items.MyItemClass2'],

'fields': **None**,

'indent': 4,

'item\_export\_kwargs': {

'export\_empty\_fields': **True**,

},

},

'/home/user/documents/items.xml': {

'format': 'xml',

'fields': ['name', 'price'],

'item\_filter': MyCustomFilter1,

'encoding': 'latin1',

'indent': 8,

},

pathlib.Path('items.csv.gz'): {

'format': 'csv',

'fields': ['price', 'name'],

'item\_filter': 'myproject.filters.MyCustomFilter2',

'postprocessing': [MyPlugin1, 'scrapy.extensions.postprocessing.GzipPlugin'],

'gzip\_compresslevel': 5,

},

}

{

'': 'scrapy.extensions.feedexport.FileFeedStorage',

'file': 'scrapy.extensions.feedexport.FileFeedStorage',

'stdout': 'scrapy.extensions.feedexport.StdoutFeedStorage',

's3': 'scrapy.extensions.feedexport.S3FeedStorage',

'ftp': 'scrapy.extensions.feedexport.FTPFeedStorage',

}

FEED\_STORAGES = {

'ftp': **None**,

}

{

'json': 'scrapy.exporters.JsonItemExporter',

'jsonlines': 'scrapy.exporters.JsonLinesItemExporter',

'jsonl': 'scrapy.exporters.JsonLinesItemExporter',

'jl': 'scrapy.exporters.JsonLinesItemExporter',

'csv': 'scrapy.exporters.CsvItemExporter',

'xml': 'scrapy.exporters.XmlItemExporter',

'marshal': 'scrapy.exporters.MarshalItemExporter',

'pickle': 'scrapy.exporters.PickleItemExporter',

}

FEED\_EXPORTERS = {

'csv': **None**,

}

FEED\_EXPORT\_BATCH\_ITEM\_COUNT = 100

scrapy crawl spidername -o "dirname/*%(batch\_id)d*-filename*%(batch\_time)s*.json"

->projectname

-->dirname

--->1-filename2020-03-28T14-45-08.237134.json

--->2-filename2020-03-28T14-45-09.148903.json

--->3-filename2020-03-28T14-45-10.046092.json

*# myproject/utils.py*

**def** uri\_params(params, spider):

**return** {\*\*params, 'spider\_name': spider.name}

*# myproject/settings.py*

FEED\_URI\_PARAMS = 'myproject.utils.uri\_params'

scrapy crawl <spider\_name> -o "*%(spider\_name)s*.jsonl"

**피드**

{

'items.json': {

'format': 'json',

'encoding': 'utf8',

'store\_empty': **False**,

'item\_classes': [MyItemClass1, 'myproject.items.MyItemClass2'],

'fields': **None**,

'indent': 4,

'item\_export\_kwargs': {

'export\_empty\_fields': **True**,

},

},

'/home/user/documents/items.xml': {

'format': 'xml',

'fields': ['name', 'price'],

'item\_filter': MyCustomFilter1,

'encoding': 'latin1',

'indent': 8,

},

pathlib.Path('items.csv.gz'): {

'format': 'csv',

'fields': ['price', 'name'],

'item\_filter': 'myproject.filters.MyCustomFilter2',

'postprocessing': [MyPlugin1, 'scrapy.extensions.postprocessing.GzipPlugin'],

'gzip\_compresslevel': 5,

},

}

**FEED\_EXPORT\_ENCODING**

**FEED\_EXPORT\_FIELDS**

**FEED\_EXPORT\_INDENT**

**FEED\_STORE\_EMPTY**

**FEED\_STORAGES**

**FEED\_STORAGE\_FTP\_ACTIVE**

**FEED\_STORAGE\_S3\_ACL**

**FEED\_STORAGES\_BASE**

{

'': 'scrapy.extensions.feedexport.FileFeedStorage',

'file': 'scrapy.extensions.feedexport.FileFeedStorage',

'stdout': 'scrapy.extensions.feedexport.StdoutFeedStorage',

's3': 'scrapy.extensions.feedexport.S3FeedStorage',

'ftp': 'scrapy.extensions.feedexport.FTPFeedStorage',

}

FEED\_STORAGES = {

'ftp': **None**,

}

**FEED\_EXPORTERS**

**FEED\_EXPORTERS\_BASE**

{

'json': 'scrapy.exporters.JsonItemExporter',

'jsonlines': 'scrapy.exporters.JsonLinesItemExporter',

'jsonl': 'scrapy.exporters.JsonLinesItemExporter',

'jl': 'scrapy.exporters.JsonLinesItemExporter',

'csv': 'scrapy.exporters.CsvItemExporter',

'xml': 'scrapy.exporters.XmlItemExporter',

'marshal': 'scrapy.exporters.MarshalItemExporter',

'pickle': 'scrapy.exporters.PickleItemExporter',

}

FEED\_EXPORTERS = {

'csv': **None**,

}

**FEED\_EXPORT\_BATCH\_ITEM\_COUNT**

FEED\_EXPORT\_BATCH\_ITEM\_COUNT = 100

scrapy crawl spidername -o "dirname/*%(batch\_id)d*-filename*%(batch\_time)s*.json"

->projectname

-->dirname

--->1-filename2020-03-28T14-45-08.237134.json

--->2-filename2020-03-28T14-45-09.148903.json

--->3-filename2020-03-28T14-45-10.046092.json

**FEED\_URI\_PARAMS**

*# myproject/utils.py*

**def** uri\_params(params, spider):

**return** {\*\*params, 'spider\_name': spider.name}

*# myproject/settings.py*

FEED\_URI\_PARAMS = 'myproject.utils.uri\_params'

scrapy crawl <spider\_name> -o "*%(spider\_name)s*.jsonl"

[**요청 및 응답**](https://docs.scrapy.org/en/latest/topics/request-response.html)

request\_with\_cookies = Request(url="http://www.example.com",

cookies={'currency': 'USD', 'country': 'UY'})

request\_with\_cookies = Request(url="http://www.example.com",

cookies=[{'name': 'currency',

'value': 'USD',

'domain': 'example.com',

'path': '/currency'}])

Request(

url="http://www.example.com",

cookies={'currency': 'USD', 'country': 'UY'},

meta={'dont\_merge\_cookies': **True**},

)

**def** parse\_page1(self, response):

**return** scrapy.Request("http://www.example.com/some\_page.html",

callback=self.parse\_page2)

**def** parse\_page2(self, response):

*# this would log http://www.example.com/some\_page.html*

self.logger.info("Visited *%s*", response.url)

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

request = scrapy.Request('http://www.example.com/index.html',

callback=self.parse\_page2,

cb\_kwargs=dict(main\_url=response.url))

request.cb\_kwargs['foo'] = 'bar' *# add more arguments for the callback*

**yield** request

**def** parse\_page2(self, response, main\_url, foo):

**yield** dict(

main\_url=main\_url,

other\_url=response.url,

foo=foo,

)

**import** **scrapy**

**from** **scrapy.spidermiddlewares.httperror** **import** HttpError

**from** **twisted.internet.error** **import** DNSLookupError

**from** **twisted.internet.error** **import** TimeoutError, TCPTimedOutError

**class** **ErrbackSpider**(scrapy.Spider):

name = "errback\_example"

start\_urls = [

"http://www.httpbin.org/", *# HTTP 200 expected*

"http://www.httpbin.org/status/404", *# Not found error*

"http://www.httpbin.org/status/500", *# server issue*

"http://www.httpbin.org:12345/", *# non-responding host, timeout expected*

"https://example.invalid/", *# DNS error expected*

]

**def** start\_requests(self):

**for** u **in** self.start\_urls:

**yield** scrapy.Request(u, callback=self.parse\_httpbin,

errback=self.errback\_httpbin,

dont\_filter=**True**)

**def** parse\_httpbin(self, response):

self.logger.info('Got successful response from *{}*'.format(response.url))

*# do something useful here...*

**def** errback\_httpbin(self, failure):

*# log all failures*

self.logger.error(repr(failure))

*# in case you want to do something special for some errors,*

*# you may need the failure's type:*

**if** failure.check(HttpError):

*# these exceptions come from HttpError spider middleware*

*# you can get the non-200 response*

response = failure.value.response

self.logger.error('HttpError on *%s*', response.url)

**elif** failure.check(DNSLookupError):

*# this is the original request*

request = failure.request

self.logger.error('DNSLookupError on *%s*', request.url)

**elif** failure.check(TimeoutError, TCPTimedOutError):

request = failure.request

self.logger.error('TimeoutError on *%s*', request.url)

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

request = scrapy.Request('http://www.example.com/index.html',

callback=self.parse\_page2,

errback=self.errback\_page2,

cb\_kwargs=dict(main\_url=response.url))

**yield** request

**def** parse\_page2(self, response, main\_url):

**pass**

**def** errback\_page2(self, failure):

**yield** dict(

main\_url=failure.request.cb\_kwargs['main\_url'],

)

*# my\_project/settings.py*

REQUEST\_FINGERPRINTER\_CLASS = 'my\_project.utils.RequestFingerprinter'

*# my\_project/utils.py*

**from** **scrapy.utils.request** **import** fingerprint

**class** **RequestFingerprinter**:

**def** fingerprint(self, request):

**return** fingerprint(request, include\_headers=['X-ID'])

**from** **hashlib** **import** sha1

**from** **weakref** **import** WeakKeyDictionary

**from** **scrapy.utils.python** **import** to\_bytes

**class** **RequestFingerprinter**:

cache = WeakKeyDictionary()

**def** fingerprint(self, request):

**if** request **not** **in** self.cache:

fp = sha1()

fp.update(to\_bytes(request.url))

self.cache[request] = fp.digest()

**return** self.cache[request]

**from** **scrapy.utils.request** **import** fingerprint

**class** **RequestFingerprinter**:

**def** fingerprint(self, request):

**if** 'fingerprint' **in** request.meta:

**return** request.meta['fingerprint']

**return** fingerprint(request)

**from** **hashlib** **import** sha1

**from** **weakref** **import** WeakKeyDictionary

**from** **scrapy.utils.python** **import** to\_bytes

**from** **w3lib.url** **import** canonicalize\_url

**class** **RequestFingerprinter**:

cache = WeakKeyDictionary()

**def** fingerprint(self, request):

**if** request **not** **in** self.cache:

fp = sha1()

fp.update(to\_bytes(request.method))

fp.update(to\_bytes(canonicalize\_url(request.url)))

fp.update(request.body **or** b'')

self.cache[request] = fp.digest()

**return** self.cache[request]

/home/user/project/.scrapy/httpcache/my\_spider/01/0123456789abcdef0123456789abcdef01234567/response\_headers

**import** **scrapy**

**class** **StopSpider**(scrapy.Spider):

name = "stop"

start\_urls = ["https://docs.scrapy.org/en/latest/"]

**@classmethod**

**def** from\_crawler(cls, crawler):

spider = super().from\_crawler(crawler)

crawler.signals.connect(spider.on\_bytes\_received, signal=scrapy.signals.bytes\_received)

**return** spider

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

*# 'last\_chars' show that the full response was not downloaded*

**yield** {"len": len(response.text), "last\_chars": response.text[-40:]}

**def** on\_bytes\_received(self, data, request, spider):

**raise** scrapy.exceptions.StopDownload(fail=**False**)

2020-05-19 17:26:12 [scrapy.core.engine] INFO: Spider opened

2020-05-19 17:26:12 [scrapy.extensions.logstats] INFO: Crawled 0 pages (at 0 pages/min), scraped 0 items (at 0 items/min)

2020-05-19 17:26:13 [scrapy.core.downloader.handlers.http11] DEBUG: Download stopped **for** <GET https://docs.scrapy.org/en/latest/> **from** **signal** handler StopSpider.on\_bytes\_received

2020-05-19 17:26:13 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://docs.scrapy.org/en/latest/> (referer: **None**) ['download\_stopped']

2020-05-19 17:26:13 [scrapy.core.scraper] DEBUG: Scraped **from** <200 https://docs.scrapy.org/en/latest/>

{'len': 279, 'last\_chars': 'dth, initial-scale=1.0">**\n** **\n** <title>Scr'}

2020-05-19 17:26:13 [scrapy.core.engine] INFO: Closing spider (finished)

**return** [FormRequest(url="http://www.example.com/post/action",

formdata={'name': 'John Doe', 'age': '27'},

callback=self.after\_post)]

**import** **scrapy**

**def** authentication\_failed(response):

*# TODO: Check the contents of the response and return True if it failed*

*# or False if it succeeded.*

**pass**

**class** **LoginSpider**(scrapy.Spider):

name = 'example.com'

start\_urls = ['http://www.example.com/users/login.php']

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

**return** scrapy.FormRequest.from\_response(

response,

formdata={'username': 'john', 'password': 'secret'},

callback=self.after\_login

)

**def** after\_login(self, response):

**if** authentication\_failed(response):

self.logger.error("Login failed")

**return**

*# continue scraping with authenticated session...*

data = {

'name1': 'value1',

'name2': 'value2',

}

**yield** JsonRequest(url='http://www.example.com/post/action', data=data)

response.headers.getlist('Set-Cookie')

urllib.parse.urljoin(response.url, url)

response.xpath('//p')

response.css('p')

**요청 객체**

request\_with\_cookies = Request(url="http://www.example.com",

cookies={'currency': 'USD', 'country': 'UY'})

request\_with\_cookies = Request(url="http://www.example.com",

cookies=[{'name': 'currency',

'value': 'USD',

'domain': 'example.com',

'path': '/currency'}])

Request(

url="http://www.example.com",

cookies={'currency': 'USD', 'country': 'UY'},

meta={'dont\_merge\_cookies': **True**},

)

**def** parse\_page1(self, response):

**return** scrapy.Request("http://www.example.com/some\_page.html",

callback=self.parse\_page2)

**def** parse\_page2(self, response):

*# this would log http://www.example.com/some\_page.html*

self.logger.info("Visited *%s*", response.url)

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

request = scrapy.Request('http://www.example.com/index.html',

callback=self.parse\_page2,

cb\_kwargs=dict(main\_url=response.url))

request.cb\_kwargs['foo'] = 'bar' *# add more arguments for the callback*

**yield** request

**def** parse\_page2(self, response, main\_url, foo):

**yield** dict(

main\_url=main\_url,

other\_url=response.url,

foo=foo,

)

**import** **scrapy**

**from** **scrapy.spidermiddlewares.httperror** **import** HttpError

**from** **twisted.internet.error** **import** DNSLookupError

**from** **twisted.internet.error** **import** TimeoutError, TCPTimedOutError

**class** **ErrbackSpider**(scrapy.Spider):

name = "errback\_example"

start\_urls = [

"http://www.httpbin.org/", *# HTTP 200 expected*

"http://www.httpbin.org/status/404", *# Not found error*

"http://www.httpbin.org/status/500", *# server issue*

"http://www.httpbin.org:12345/", *# non-responding host, timeout expected*

"https://example.invalid/", *# DNS error expected*

]

**def** start\_requests(self):

**for** u **in** self.start\_urls:

**yield** scrapy.Request(u, callback=self.parse\_httpbin,

errback=self.errback\_httpbin,

dont\_filter=**True**)

**def** parse\_httpbin(self, response):

self.logger.info('Got successful response from *{}*'.format(response.url))

*# do something useful here...*

**def** errback\_httpbin(self, failure):

*# log all failures*

self.logger.error(repr(failure))

*# in case you want to do something special for some errors,*

*# you may need the failure's type:*

**if** failure.check(HttpError):

*# these exceptions come from HttpError spider middleware*

*# you can get the non-200 response*

response = failure.value.response

self.logger.error('HttpError on *%s*', response.url)

**elif** failure.check(DNSLookupError):

*# this is the original request*

request = failure.request

self.logger.error('DNSLookupError on *%s*', request.url)

**elif** failure.check(TimeoutError, TCPTimedOutError):

request = failure.request

self.logger.error('TimeoutError on *%s*', request.url)

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

request = scrapy.Request('http://www.example.com/index.html',

callback=self.parse\_page2,

errback=self.errback\_page2,

cb\_kwargs=dict(main\_url=response.url))

**yield** request

**def** parse\_page2(self, response, main\_url):

**pass**

**def** errback\_page2(self, failure):

**yield** dict(

main\_url=failure.request.cb\_kwargs['main\_url'],

)

*# my\_project/settings.py*

REQUEST\_FINGERPRINTER\_CLASS = 'my\_project.utils.RequestFingerprinter'

*# my\_project/utils.py*

**from** **scrapy.utils.request** **import** fingerprint

**class** **RequestFingerprinter**:

**def** fingerprint(self, request):

**return** fingerprint(request, include\_headers=['X-ID'])

**from** **hashlib** **import** sha1

**from** **weakref** **import** WeakKeyDictionary

**from** **scrapy.utils.python** **import** to\_bytes

**class** **RequestFingerprinter**:

cache = WeakKeyDictionary()

**def** fingerprint(self, request):

**if** request **not** **in** self.cache:

fp = sha1()

fp.update(to\_bytes(request.url))

self.cache[request] = fp.digest()

**return** self.cache[request]

**from** **scrapy.utils.request** **import** fingerprint

**class** **RequestFingerprinter**:

**def** fingerprint(self, request):

**if** 'fingerprint' **in** request.meta:

**return** request.meta['fingerprint']

**return** fingerprint(request)

**from** **hashlib** **import** sha1

**from** **weakref** **import** WeakKeyDictionary

**from** **scrapy.utils.python** **import** to\_bytes

**from** **w3lib.url** **import** canonicalize\_url

**class** **RequestFingerprinter**:

cache = WeakKeyDictionary()

**def** fingerprint(self, request):

**if** request **not** **in** self.cache:

fp = sha1()

fp.update(to\_bytes(request.method))

fp.update(to\_bytes(canonicalize\_url(request.url)))

fp.update(request.body **or** b'')

self.cache[request] = fp.digest()

**return** self.cache[request]

/home/user/project/.scrapy/httpcache/my\_spider/01/0123456789abcdef0123456789abcdef01234567/response\_headers

**요청과 관련된 기타 기능**

**콜백 함수에 추가 데이터 전달**

**def** parse\_page1(self, response):

**return** scrapy.Request("http://www.example.com/some\_page.html",

callback=self.parse\_page2)

**def** parse\_page2(self, response):

*# this would log http://www.example.com/some\_page.html*

self.logger.info("Visited *%s*", response.url)

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

request = scrapy.Request('http://www.example.com/index.html',

callback=self.parse\_page2,

cb\_kwargs=dict(main\_url=response.url))

request.cb\_kwargs['foo'] = 'bar' *# add more arguments for the callback*

**yield** request

**def** parse\_page2(self, response, main\_url, foo):

**yield** dict(

main\_url=main\_url,

other\_url=response.url,

foo=foo,

)

**errback을 사용하여 요청 처리에서 예외 포착**

**import** **scrapy**

**from** **scrapy.spidermiddlewares.httperror** **import** HttpError

**from** **twisted.internet.error** **import** DNSLookupError

**from** **twisted.internet.error** **import** TimeoutError, TCPTimedOutError

**class** **ErrbackSpider**(scrapy.Spider):

name = "errback\_example"

start\_urls = [

"http://www.httpbin.org/", *# HTTP 200 expected*

"http://www.httpbin.org/status/404", *# Not found error*

"http://www.httpbin.org/status/500", *# server issue*

"http://www.httpbin.org:12345/", *# non-responding host, timeout expected*

"https://example.invalid/", *# DNS error expected*

]

**def** start\_requests(self):

**for** u **in** self.start\_urls:

**yield** scrapy.Request(u, callback=self.parse\_httpbin,

errback=self.errback\_httpbin,

dont\_filter=**True**)

**def** parse\_httpbin(self, response):

self.logger.info('Got successful response from *{}*'.format(response.url))

*# do something useful here...*

**def** errback\_httpbin(self, failure):

*# log all failures*

self.logger.error(repr(failure))

*# in case you want to do something special for some errors,*

*# you may need the failure's type:*

**if** failure.check(HttpError):

*# these exceptions come from HttpError spider middleware*

*# you can get the non-200 response*

response = failure.value.response

self.logger.error('HttpError on *%s*', response.url)

**elif** failure.check(DNSLookupError):

*# this is the original request*

request = failure.request

self.logger.error('DNSLookupError on *%s*', request.url)

**elif** failure.check(TimeoutError, TCPTimedOutError):

request = failure.request

self.logger.error('TimeoutError on *%s*', request.url)

**errback 함수에서 추가 데이터 액세스**

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

request = scrapy.Request('http://www.example.com/index.html',

callback=self.parse\_page2,

errback=self.errback\_page2,

cb\_kwargs=dict(main\_url=response.url))

**yield** request

**def** parse\_page2(self, response, main\_url):

**pass**

**def** errback\_page2(self, failure):

**yield** dict(

main\_url=failure.request.cb\_kwargs['main\_url'],

)

**지문 요청**

*# my\_project/settings.py*

REQUEST\_FINGERPRINTER\_CLASS = 'my\_project.utils.RequestFingerprinter'

*# my\_project/utils.py*

**from** **scrapy.utils.request** **import** fingerprint

**class** **RequestFingerprinter**:

**def** fingerprint(self, request):

**return** fingerprint(request, include\_headers=['X-ID'])

**from** **hashlib** **import** sha1

**from** **weakref** **import** WeakKeyDictionary

**from** **scrapy.utils.python** **import** to\_bytes

**class** **RequestFingerprinter**:

cache = WeakKeyDictionary()

**def** fingerprint(self, request):

**if** request **not** **in** self.cache:

fp = sha1()

fp.update(to\_bytes(request.url))

self.cache[request] = fp.digest()

**return** self.cache[request]

**from** **scrapy.utils.request** **import** fingerprint

**class** **RequestFingerprinter**:

**def** fingerprint(self, request):

**if** 'fingerprint' **in** request.meta:

**return** request.meta['fingerprint']

**return** fingerprint(request)

**from** **hashlib** **import** sha1

**from** **weakref** **import** WeakKeyDictionary

**from** **scrapy.utils.python** **import** to\_bytes

**from** **w3lib.url** **import** canonicalize\_url

**class** **RequestFingerprinter**:

cache = WeakKeyDictionary()

**def** fingerprint(self, request):

**if** request **not** **in** self.cache:

fp = sha1()

fp.update(to\_bytes(request.method))

fp.update(to\_bytes(canonicalize\_url(request.url)))

fp.update(request.body **or** b'')

self.cache[request] = fp.digest()

**return** self.cache[request]

/home/user/project/.scrapy/httpcache/my\_spider/01/0123456789abcdef0123456789abcdef01234567/response\_headers

**REQUEST\_FINGERPRINTER\_CLASS**

**REQUEST\_FINGERPRINTER\_IMPLEMENTATION**

**자신의 요청 핑거프린팅 작성**

*# my\_project/settings.py*

REQUEST\_FINGERPRINTER\_CLASS = 'my\_project.utils.RequestFingerprinter'

*# my\_project/utils.py*

**from** **scrapy.utils.request** **import** fingerprint

**class** **RequestFingerprinter**:

**def** fingerprint(self, request):

**return** fingerprint(request, include\_headers=['X-ID'])

**from** **hashlib** **import** sha1

**from** **weakref** **import** WeakKeyDictionary

**from** **scrapy.utils.python** **import** to\_bytes

**class** **RequestFingerprinter**:

cache = WeakKeyDictionary()

**def** fingerprint(self, request):

**if** request **not** **in** self.cache:

fp = sha1()

fp.update(to\_bytes(request.url))

self.cache[request] = fp.digest()

**return** self.cache[request]

**from** **scrapy.utils.request** **import** fingerprint

**class** **RequestFingerprinter**:

**def** fingerprint(self, request):

**if** 'fingerprint' **in** request.meta:

**return** request.meta['fingerprint']

**return** fingerprint(request)

**from** **hashlib** **import** sha1

**from** **weakref** **import** WeakKeyDictionary

**from** **scrapy.utils.python** **import** to\_bytes

**from** **w3lib.url** **import** canonicalize\_url

**class** **RequestFingerprinter**:

cache = WeakKeyDictionary()

**def** fingerprint(self, request):

**if** request **not** **in** self.cache:

fp = sha1()

fp.update(to\_bytes(request.method))

fp.update(to\_bytes(canonicalize\_url(request.url)))

fp.update(request.body **or** b'')

self.cache[request] = fp.digest()

**return** self.cache[request]

**지문 제한 요청**

/home/user/project/.scrapy/httpcache/my\_spider/01/0123456789abcdef0123456789abcdef01234567/response\_headers

**Request.meta 특수 키**

**바인드 주소**

**다운로드\_시간 초과**

**다운로드\_지연**

**download\_fail\_on\_dataloss**

**max\_retry\_times**

**응답 다운로드 중지**

**import** **scrapy**

**class** **StopSpider**(scrapy.Spider):

name = "stop"

start\_urls = ["https://docs.scrapy.org/en/latest/"]

**@classmethod**

**def** from\_crawler(cls, crawler):

spider = super().from\_crawler(crawler)

crawler.signals.connect(spider.on\_bytes\_received, signal=scrapy.signals.bytes\_received)

**return** spider

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

*# 'last\_chars' show that the full response was not downloaded*

**yield** {"len": len(response.text), "last\_chars": response.text[-40:]}

**def** on\_bytes\_received(self, data, request, spider):

**raise** scrapy.exceptions.StopDownload(fail=**False**)

2020-05-19 17:26:12 [scrapy.core.engine] INFO: Spider opened

2020-05-19 17:26:12 [scrapy.extensions.logstats] INFO: Crawled 0 pages (at 0 pages/min), scraped 0 items (at 0 items/min)

2020-05-19 17:26:13 [scrapy.core.downloader.handlers.http11] DEBUG: Download stopped **for** <GET https://docs.scrapy.org/en/latest/> **from** **signal** handler StopSpider.on\_bytes\_received

2020-05-19 17:26:13 [scrapy.core.engine] DEBUG: Crawled (200) <GET https://docs.scrapy.org/en/latest/> (referer: **None**) ['download\_stopped']

2020-05-19 17:26:13 [scrapy.core.scraper] DEBUG: Scraped **from** <200 https://docs.scrapy.org/en/latest/>

{'len': 279, 'last\_chars': 'dth, initial-scale=1.0">**\n** **\n** <title>Scr'}

2020-05-19 17:26:13 [scrapy.core.engine] INFO: Closing spider (finished)

**서브클래스 요청**

**return** [FormRequest(url="http://www.example.com/post/action",

formdata={'name': 'John Doe', 'age': '27'},

callback=self.after\_post)]

**import** **scrapy**

**def** authentication\_failed(response):

*# TODO: Check the contents of the response and return True if it failed*

*# or False if it succeeded.*

**pass**

**class** **LoginSpider**(scrapy.Spider):

name = 'example.com'

start\_urls = ['http://www.example.com/users/login.php']

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

**return** scrapy.FormRequest.from\_response(

response,

formdata={'username': 'john', 'password': 'secret'},

callback=self.after\_login

)

**def** after\_login(self, response):

**if** authentication\_failed(response):

self.logger.error("Login failed")

**return**

*# continue scraping with authenticated session...*

data = {

'name1': 'value1',

'name2': 'value2',

}

**yield** JsonRequest(url='http://www.example.com/post/action', data=data)

**FormRequest 객체**

**요청 사용 예**

**return** [FormRequest(url="http://www.example.com/post/action",

formdata={'name': 'John Doe', 'age': '27'},

callback=self.after\_post)]

**import** **scrapy**

**def** authentication\_failed(response):

*# TODO: Check the contents of the response and return True if it failed*

*# or False if it succeeded.*

**pass**

**class** **LoginSpider**(scrapy.Spider):

name = 'example.com'

start\_urls = ['http://www.example.com/users/login.php']

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

**return** scrapy.FormRequest.from\_response(

response,

formdata={'username': 'john', 'password': 'secret'},

callback=self.after\_login

)

**def** after\_login(self, response):

**if** authentication\_failed(response):

self.logger.error("Login failed")

**return**

*# continue scraping with authenticated session...*

**FormRequest를 사용하여 HTTP POST를 통해 데이터 보내기**

**return** [FormRequest(url="http://www.example.com/post/action",

formdata={'name': 'John Doe', 'age': '27'},

callback=self.after\_post)]

**FormRequest.from\_response()를 사용하여 사용자 로그인 시뮬레이션**

**import** **scrapy**

**def** authentication\_failed(response):

*# TODO: Check the contents of the response and return True if it failed*

*# or False if it succeeded.*

**pass**

**class** **LoginSpider**(scrapy.Spider):

name = 'example.com'

start\_urls = ['http://www.example.com/users/login.php']

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

**return** scrapy.FormRequest.from\_response(

response,

formdata={'username': 'john', 'password': 'secret'},

callback=self.after\_login

)

**def** after\_login(self, response):

**if** authentication\_failed(response):

self.logger.error("Login failed")

**return**

*# continue scraping with authenticated session...*

**Json요청**

**JsonRequest 사용 예**

data = {

'name1': 'value1',

'name2': 'value2',

}

**yield** JsonRequest(url='http://www.example.com/post/action', data=data)

**응답 객체**

response.headers.getlist('Set-Cookie')

urllib.parse.urljoin(response.url, url)

**응답 서브클래스**

response.xpath('//p')

response.css('p')

**TextResponse 객체**

response.xpath('//p')

response.css('p')

**HTMLResponse 객체**

**XmlResponse 개체**

[**링크 추출기**](https://docs.scrapy.org/en/latest/topics/link-extractors.html)

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

**for** link **in** self.link\_extractor.extract\_links(response):

**yield** Request(link.url, callback=self.parse)

**from** **scrapy.linkextractors** **import** LinkExtractor

**링크 추출기 참조**

**from** **scrapy.linkextractors** **import** LinkExtractor

**LxmlLink 추출기**

**링크**

[**설정**](https://docs.scrapy.org/en/latest/topics/settings.html)

**설정 지정**

**설정 채우기**

**1. 명령줄 옵션**

**2. 거미별 설정**

**3. 프로젝트 설정 모듈**

**4. 명령어별 기본 설정**

**5. 기본 전역 설정**

**피클과의 호환성**

**경로 및 클래스 가져오기**

**설정에 액세스하는 방법**

**이름을 설정하는 근거**

**기본 제공 설정 참조**

**AWS\_ACCESS\_KEY\_ID**

**AWS\_SECRET\_ACCESS\_KEY**

**AWS\_SESSION\_TOKEN**

**AWS\_ENDPOINT\_URL**

**AWS\_USE\_SSL**

**AWS\_VERIFY**

**AWS\_REGION\_NAME**

**ASYNCIO\_EVENT\_LOOP**

**BOT\_NAME**

**CONCURRENT\_ITEMS**

**CONCURRENT\_REQUESTS**

**CONCURRENT\_REQUESTS\_PER\_DOMAIN**

**CONCURRENT\_REQUESTS\_PER\_IP**

**DEFAULT\_ITEM\_CLASS**

**DEFAULT\_REQUEST\_HEADERS**

**DEPTH\_LIMIT**

**DEPTH\_PRIORITY**

**DEPTH\_STATS\_VERBOSE**

**DNSCACHE\_ENABLED**

**DNSCACHE\_SIZE**

**DNS\_RESOLVER**

**DNS\_TIMEOUT**

**다운로더**

**DOWNLOADER\_HTTPCLIENTFACTORY**

**DOWNLOADER\_CLIENTCONTEXTFACTORY**

**DOWNLOADER\_CLIENT\_TLS\_CIPHERS**

**DOWNLOADER\_CLIENT\_TLS\_METHOD**

**DOWNLOADER\_CLIENT\_TLS\_VERBOSE\_LOGGING**

**DOWNLOADER\_MIDDLEWARES**

**DOWNLOADER\_MIDDLEWARES\_BASE**

**DOWNLOADER\_STATS**

**DOWNLOAD\_DELAY**

**DOWNLOAD\_HANDLERS**

**DOWNLOAD\_HANDLERS\_BASE**

**DOWNLOAD\_TIMEOUT**

**DOWNLOAD\_MAXSIZE**

**DOWNLOAD\_WARNSIZE**

**다운로드\_FAIL\_ON\_DATALOSS**

**DUPEFILTER\_CLASS**

**DUPEFILTER\_DEBUG**

**편집자**

**확장**

**EXTENSIONS\_BASE**

**FEED\_TEMPDIR**

**FEED\_STORAGE\_GCS\_ACL**

**FTP\_PASSIVE\_MODE**

**FTP\_PASSWORD**

**FTP\_USER**

**GCS\_PROJECT\_ID**

**ITEM\_PIPELINES**

**ITEM\_PIPELINES\_BASE**

**욥디르**

**LOG\_ENABLED**

**LOG\_ENCODING**

**LOG\_FILE**

**LOG\_FILE\_APPEND**

**LOG\_FORMAT**

**LOG\_DATEFORMAT**

**LOG\_FORMATTER**

**LOG\_LEVEL**

**로그\_STDOUT**

**LOG\_SHORT\_NAMES**

**LOGSTATS\_INTERVAL**

**MEMDEBUG\_ENABLED**

**MEMDEBUG\_NOTIFY**

**MEMUSAGE\_ENABLED**

**MEMUSAGE\_LIMIT\_MB**

**MEMUSAGE\_CHECK\_INTERVAL\_SECONDS**

**MEMUSAGE\_NOTIFY\_MAIL**

**MEMUSAGE\_WARNING\_MB**

**NEWSPIDER\_MODULE**

**RANDOMIZE\_DOWNLOAD\_DELAY**

**REACTOR\_THREADPOOL\_MAXSIZE**

**REDIRECT\_PRIORITY\_ADJUST**

**ROBOTSXT\_OBEY**

**ROBOTSXT\_PARSER**

**ROBOTSXT\_USER\_AGENT**

**스케줄러**

**SCHEDULER\_DEBUG**

**SCHEDULER\_DISK\_QUEUE**

**SCHEDULER\_MEMORY\_QUEUE**

**SCHEDULER\_PRIORITY\_QUEUE**

**SCRAPER\_SLOT\_MAX\_ACTIVE\_SIZE**

**거미\_계약**

**SPIDER\_CONTRACTS\_BASE**

**SPIDER\_LOADER\_CLASS**

**SPIDER\_LOADER\_WARN\_ONLY**

**SPIDER\_MIDDLEWARES**

**SPIDER\_MIDDLEWARES\_BASE**

**SPIDER\_MODULES**

**STATS\_CLASS**

**STATS\_DUMP**

**STATSMAILER\_RCPTS**

**TELNETCONSOLE\_ENABLED**

**템플릿\_DIR**

**트위스티드\_리액터**

**URLLENGTH\_LIMIT**

**사용자 에이전트**

**다른 곳에서 문서화된 설정:**

[**예외**](https://docs.scrapy.org/en/latest/topics/exceptions.html)

**def** parse\_page(self, response):

**if** 'Bandwidth exceeded' **in** response.body:

**raise** CloseSpider('bandwidth\_exceeded')

**기본 제공 예외 참조**

**def** parse\_page(self, response):

**if** 'Bandwidth exceeded' **in** response.body:

**raise** CloseSpider('bandwidth\_exceeded')

**닫기스파이더**

**def** parse\_page(self, response):

**if** 'Bandwidth exceeded' **in** response.body:

**raise** CloseSpider('bandwidth\_exceeded')

**DontCloseSpider**

**드롭아이템**

**요청 무시**

**구성되지 않음**

**지원되지 않음**

**다운로드 중지**

[**벌채 반출**](https://docs.scrapy.org/en/latest/topics/logging.html)

**import** **logging**

logging.warning("This is a warning")

**import** **logging**

logging.log(logging.WARNING, "This is a warning")

**import** **logging**

logger = logging.getLogger()

logger.warning("This is a warning")

**import** **logging**

logger = logging.getLogger('mycustomlogger')

logger.warning("This is a warning")

**import** **logging**

logger = logging.getLogger(\_\_name\_\_)

logger.warning("This is a warning")

**import** **scrapy**

**class** **MySpider**(scrapy.Spider):

name = 'myspider'

start\_urls = ['https://scrapy.org']

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

self.logger.info('Parse function called on *%s*', response.url)

**import** **logging**

**import** **scrapy**

logger = logging.getLogger('mycustomlogger')

**class** **MySpider**(scrapy.Spider):

name = 'myspider'

start\_urls = ['https://scrapy.org']

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

logger.info('Parse function called on *%s*', response.url)

**class** **PoliteLogFormatter**(logformatter.LogFormatter):

**def** dropped(self, item, exception, response, spider):

**return** {

'level': logging.INFO, *# lowering the level from logging.WARNING*

'msg': "Dropped: *%(exception)s*" + os.linesep + "*%(item)s*",

'args': {

'exception': exception,

'item': item,

}

}

2016-12-16 22:00:06 [scrapy.spidermiddlewares.httperror] INFO: Ignoring

response <500 https://quotes.toscrape.com/page/1-34/>: HTTP status code

**is** **not** handled **or** **not** allowed

**import** **logging**

**import** **scrapy**

**class** **MySpider**(scrapy.Spider):

*# ...*

**def** \_\_init\_\_(self, \*args, \*\*kwargs):

logger = logging.getLogger('scrapy.spidermiddlewares.httperror')

logger.setLevel(logging.WARNING)

super().\_\_init\_\_(\*args, \*\*kwargs)

**import** **logging**

**import** **re**

**class** **ContentFilter**(logging.Filter):

**def** filter(self, record):

match = re.search(r'\d*{3}* [Ee]rror, retrying', record.message)

**if** match:

**return** **False**

**import** **logging**

**import** **scrapy**

**class** **MySpider**(scrapy.Spider):

*# ...*

**def** \_\_init\_\_(self, \*args, \*\*kwargs):

**for** handler **in** logging.root.handlers:

handler.addFilter(ContentFilter())

**import** **logging**

**import** **scrapy**

**class** **MySpider**(scrapy.Spider):

*# ...*

**def** \_\_init\_\_(self, \*args, \*\*kwargs):

logger = logging.getLogger('my\_logger')

logger.addFilter(ContentFilter())

**import** **logging**

logging.basicConfig(

filename='log.txt',

format='*%(levelname)s*: *%(message)s*',

level=logging.INFO

)

**로그 수준**

**메시지를 기록하는 방법**

**import** **logging**

logging.warning("This is a warning")

**import** **logging**

logging.log(logging.WARNING, "This is a warning")

**import** **logging**

logger = logging.getLogger()

logger.warning("This is a warning")

**import** **logging**

logger = logging.getLogger('mycustomlogger')

logger.warning("This is a warning")

**import** **logging**

logger = logging.getLogger(\_\_name\_\_)

logger.warning("This is a warning")

**스파이더에서 로깅**

**import** **scrapy**

**class** **MySpider**(scrapy.Spider):

name = 'myspider'

start\_urls = ['https://scrapy.org']

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

self.logger.info('Parse function called on *%s*', response.url)

**import** **logging**

**import** **scrapy**

logger = logging.getLogger('mycustomlogger')

**class** **MySpider**(scrapy.Spider):

name = 'myspider'

start\_urls = ['https://scrapy.org']

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

logger.info('Parse function called on *%s*', response.url)

**로깅 구성**

**class** **PoliteLogFormatter**(logformatter.LogFormatter):

**def** dropped(self, item, exception, response, spider):

**return** {

'level': logging.INFO, *# lowering the level from logging.WARNING*

'msg': "Dropped: *%(exception)s*" + os.linesep + "*%(item)s*",

'args': {

'exception': exception,

'item': item,

}

}

2016-12-16 22:00:06 [scrapy.spidermiddlewares.httperror] INFO: Ignoring

response <500 https://quotes.toscrape.com/page/1-34/>: HTTP status code

**is** **not** handled **or** **not** allowed

**import** **logging**

**import** **scrapy**

**class** **MySpider**(scrapy.Spider):

*# ...*

**def** \_\_init\_\_(self, \*args, \*\*kwargs):

logger = logging.getLogger('scrapy.spidermiddlewares.httperror')

logger.setLevel(logging.WARNING)

super().\_\_init\_\_(\*args, \*\*kwargs)

**import** **logging**

**import** **re**

**class** **ContentFilter**(logging.Filter):

**def** filter(self, record):

match = re.search(r'\d*{3}* [Ee]rror, retrying', record.message)

**if** match:

**return** **False**

**import** **logging**

**import** **scrapy**

**class** **MySpider**(scrapy.Spider):

*# ...*

**def** \_\_init\_\_(self, \*args, \*\*kwargs):

**for** handler **in** logging.root.handlers:

handler.addFilter(ContentFilter())

**import** **logging**

**import** **scrapy**

**class** **MySpider**(scrapy.Spider):

*# ...*

**def** \_\_init\_\_(self, \*args, \*\*kwargs):

logger = logging.getLogger('my\_logger')

logger.addFilter(ContentFilter())

**로깅 설정**

**명령줄 옵션**

**사용자 정의 로그 형식**

**class** **PoliteLogFormatter**(logformatter.LogFormatter):

**def** dropped(self, item, exception, response, spider):

**return** {

'level': logging.INFO, *# lowering the level from logging.WARNING*

'msg': "Dropped: *%(exception)s*" + os.linesep + "*%(item)s*",

'args': {

'exception': exception,

'item': item,

}

}

**고급 사용자 정의**

2016-12-16 22:00:06 [scrapy.spidermiddlewares.httperror] INFO: Ignoring

response <500 https://quotes.toscrape.com/page/1-34/>: HTTP status code

**is** **not** handled **or** **not** allowed

**import** **logging**

**import** **scrapy**

**class** **MySpider**(scrapy.Spider):

*# ...*

**def** \_\_init\_\_(self, \*args, \*\*kwargs):

logger = logging.getLogger('scrapy.spidermiddlewares.httperror')

logger.setLevel(logging.WARNING)

super().\_\_init\_\_(\*args, \*\*kwargs)

**import** **logging**

**import** **re**

**class** **ContentFilter**(logging.Filter):

**def** filter(self, record):

match = re.search(r'\d*{3}* [Ee]rror, retrying', record.message)

**if** match:

**return** **False**

**import** **logging**

**import** **scrapy**

**class** **MySpider**(scrapy.Spider):

*# ...*

**def** \_\_init\_\_(self, \*args, \*\*kwargs):

**for** handler **in** logging.root.handlers:

handler.addFilter(ContentFilter())

**import** **logging**

**import** **scrapy**

**class** **MySpider**(scrapy.Spider):

*# ...*

**def** \_\_init\_\_(self, \*args, \*\*kwargs):

logger = logging.getLogger('my\_logger')

logger.addFilter(ContentFilter())

**scrapy.utils.log 모듈**

**import** **logging**

logging.basicConfig(

filename='log.txt',

format='*%(levelname)s*: *%(message)s*',

level=logging.INFO

)

[**통계 수집**](https://docs.scrapy.org/en/latest/topics/stats.html)

**class** **ExtensionThatAccessStats**:

**def** \_\_init\_\_(self, stats):

self.stats = stats

**@classmethod**

**def** from\_crawler(cls, crawler):

**return** cls(crawler.stats)

stats.set\_value('hostname', socket.gethostname())

stats.inc\_value('custom\_count')

stats.max\_value('max\_items\_scraped', value)

stats.min\_value('min\_free\_memory\_percent', value)

**일반적인 통계 수집기 사용**

**class** **ExtensionThatAccessStats**:

**def** \_\_init\_\_(self, stats):

self.stats = stats

**@classmethod**

**def** from\_crawler(cls, crawler):

**return** cls(crawler.stats)

stats.set\_value('hostname', socket.gethostname())

stats.inc\_value('custom\_count')

stats.max\_value('max\_items\_scraped', value)

stats.min\_value('min\_free\_memory\_percent', value)

**사용 가능한 통계 수집기**

**메모리 통계 수집기**

**더미 통계 수집기**

[**이메일 보내기**](https://docs.scrapy.org/en/latest/topics/email.html)

**from** **scrapy.mail** **import** MailSender

mailer = MailSender()

mailer = MailSender.from\_settings(settings)

mailer.send(to=["someone@example.com"], subject="Some subject", body="Some body", cc=["another@example.com"])

**빠른 예**

**from** **scrapy.mail** **import** MailSender

mailer = MailSender()

mailer = MailSender.from\_settings(settings)

mailer.send(to=["someone@example.com"], subject="Some subject", body="Some body", cc=["another@example.com"])

**MailSender 클래스 참조**

**메일 설정**

**MAIL\_FROM**

**MAIL\_HOST**

**MAIL\_PORT**

**MAIL\_USER**

**MAIL\_PASS**

**MAIL\_TLS**

**MAIL\_SSL**

[**텔넷 콘솔**](https://docs.scrapy.org/en/latest/topics/telnetconsole.html)

**텔넷 콘솔에 액세스하는 방법**

**텔넷 콘솔에서 사용 가능한 변수**

**Telnet 콘솔 사용 예**

**엔진 상태 보기**

**Scrapy 엔진 일시 중지, 재개 및 중지**

**텔넷 콘솔 신호**

**텔넷 설정**

**TELNETCONSOLE\_PORT**

**TELNETCONSOLE\_HOST**

**TELNETCONSOLE\_USERNAME**

**TELNETCONSOLE\_PASSWORD**

[**자주 묻는 질문**](https://docs.scrapy.org/en/latest/faq.html)

**from** **bs4** **import** BeautifulSoup

**import** **scrapy**

**class** **ExampleSpider**(scrapy.Spider):

name = "example"

allowed\_domains = ["example.com"]

start\_urls = (

'http://www.example.com/',

)

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

*# use lxml to get decent HTML parsing speed*

soup = BeautifulSoup(response.text, 'lxml')

**yield** {

"url": response.url,

"title": soup.h1.string

}

DEPTH\_PRIORITY = 1

SCHEDULER\_DISK\_QUEUE = 'scrapy.squeues.PickleFifoDiskQueue'

SCHEDULER\_MEMORY\_QUEUE = 'scrapy.squeues.FifoMemoryQueue'

SPIDER\_MIDDLEWARES = {

'scrapy.spidermiddlewares.offsite.OffsiteMiddleware': **None**,

'myproject.middlewares.CustomOffsiteMiddleware': 500,

}

scrapy runspider my\_spider.py

**class** **MySpider**(CrawlSpider):

name = 'myspider'

download\_delay = 2

*# [ ... rest of the spider code ... ]*

scrapy crawl myspider -O items.json

scrapy crawl myspider -O items.csv

scrapy crawl myspider -O items.xml

**from** **copy** **import** deepcopy

**from** **itemadapter** **import** is\_item, ItemAdapter

**class** **MultiplyItemsMiddleware**:

**def** process\_spider\_output(self, response, result, spider):

**for** item **in** result:

**if** is\_item(item):

adapter = ItemAdapter(item)

**for** \_ **in** range(adapter['multiply\_by']):

**yield** deepcopy(item)

**Scrapy는 BeautifulSoup 또는 lxml과 어떻게 비교됩니까?**

**BeautifulSoup과 함께 Scrapy를 사용할 수 있습니까?**

**from** **bs4** **import** BeautifulSoup

**import** **scrapy**

**class** **ExampleSpider**(scrapy.Spider):

name = "example"

allowed\_domains = ["example.com"]

start\_urls = (

'http://www.example.com/',

)

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

*# use lxml to get decent HTML parsing speed*

soup = BeautifulSoup(response.text, 'lxml')

**yield** {

"url": response.url,

"title": soup.h1.string

}

**Scrapy가 Django에서 X를 "훔쳤습니까"?**

**Scrapy는 HTTP 프록시와 함께 작동합니까?**

**다른 페이지에 속성이 있는 항목을 긁어내려면 어떻게 해야 합니까?**

**내 스파이더에서 사용자 로그인을 시뮬레이션하려면 어떻게 해야 합니까?**

**Scrapy는 너비 우선 또는 깊이 우선 순서로 크롤링합니까?**

DEPTH\_PRIORITY = 1

SCHEDULER\_DISK\_QUEUE = 'scrapy.squeues.PickleFifoDiskQueue'

SCHEDULER\_MEMORY\_QUEUE = 'scrapy.squeues.FifoMemoryQueue'

**내 Scrapy 크롤러에 메모리 누수가 있습니다. 어떡해?**

**Scrapy가 메모리를 덜 사용하도록 하려면 어떻게 해야 합니까?**

**많은 허용 도메인으로 인한 메모리 오류를 방지하려면 어떻게 해야 합니까?**

SPIDER\_MIDDLEWARES = {

'scrapy.spidermiddlewares.offsite.OffsiteMiddleware': **None**,

'myproject.middlewares.CustomOffsiteMiddleware': 500,

}

**내 스파이더에서 기본 HTTP 인증을 사용할 수 있습니까?**

**Scrapy가 내 모국어 대신 영어로 페이지를 다운로드하는 이유는 무엇입니까?**

**Scrapy 프로젝트의 예는 어디에서 찾을 수 있습니까?**

**프로젝트를 생성하지 않고 스파이더를 실행할 수 있습니까?**

scrapy runspider my\_spider.py

**"필터링된 오프사이트 요청" 메시지가 나타납니다. 어떻게 고칠 수 있습니까?**

**프로덕션에서 Scrapy 크롤러를 배포하는 데 권장되는 방법은 무엇입니까?**

**대규모 내보내기에 JSON을 사용할 수 있습니까?**

**시그널 핸들러에서 지연된 (Twisted) 반환할 수 있습니까?**

**응답 상태 코드 999는 무엇을 의미합니까?**

**class** **MySpider**(CrawlSpider):

name = 'myspider'

download\_delay = 2

*# [ ... rest of the spider code ... ]*

**내 스파이더에서 pdb.set\_trace()를 호출하여 디버깅할 수 있습니까?**

**모든 스크랩 항목을 JSON/CSV/XML 파일로 덤프하는 가장 간단한 방법은 무엇입니까?**

scrapy crawl myspider -O items.json

scrapy crawl myspider -O items.csv

scrapy crawl myspider -O items.xml

**일부 형식에서 사용되는 이 거대하고 비밀스러운 \_\_VIEWSTATE 매개변수는 무엇입니까?**

**큰 XML/CSV 데이터 피드를 구문 분석하는 가장 좋은 방법은 무엇입니까?**

**Scrapy는 쿠키를 자동으로 관리합니까?**

**Scrapy에서 보내고 받는 쿠키를 어떻게 볼 수 있습니까?**

**거미에게 스스로 멈추도록 지시하려면 어떻게 해야 합니까?**

**내 Scrapy 봇이 금지되는 것을 방지하려면 어떻게 해야 합니까?**

**스파이더 인수 또는 설정을 사용하여 스파이더를 구성해야 합니까?**

**XML 문서를 스크랩하고 있는데 XPath 선택기가 항목을 반환하지 않습니다.**

**항목 파이프라인에서 항목을 여러 항목으로 분할하는 방법은 무엇입니까?**

**from** **copy** **import** deepcopy

**from** **itemadapter** **import** is\_item, ItemAdapter

**class** **MultiplyItemsMiddleware**:

**def** process\_spider\_output(self, response, result, spider):

**for** item **in** result:

**if** is\_item(item):

adapter = ItemAdapter(item)

**for** \_ **in** range(adapter['multiply\_by']):

**yield** deepcopy(item)

**Scrapy는 IPv6 주소를 지원합니까?**

**<class 'ValueError'>를 처리하는 방법: select() 예외에서 파일 설명자가 범위를 벗어났습니까?**

**주어진 응답의 다운로드를 어떻게 취소할 수 있습니까?**

**runspider를 실행하면 오류가 발생합니다: <filename> 파일에 스파이더가 없습니다.**

[**디버깅 스파이더**](https://docs.scrapy.org/en/latest/topics/debug.html)

**import** **scrapy**

**from** **myproject.items** **import** MyItem

**class** **MySpider**(scrapy.Spider):

name = 'myspider'

start\_urls = (

'http://example.com/page1',

'http://example.com/page2',

)

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

*# <processing code not shown>*

*# collect `item\_urls`*

**for** item\_url **in** item\_urls:

**yield** scrapy.Request(item\_url, self.parse\_item)

**def** parse\_item(self, response):

*# <processing code not shown>*

item = MyItem()

*# populate `item` fields*

*# and extract item\_details\_url*

**yield** scrapy.Request(item\_details\_url, self.parse\_details, cb\_kwargs={'item': item})

**def** parse\_details(self, response, item):

*# populate more `item` fields*

**return** item

**구문 분석 명령**

**스크랩 쉘**

**브라우저에서 열려**

**벌채 반출**

[**거미 계약**](https://docs.scrapy.org/en/latest/topics/contracts.html)

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

*""" This function parses a sample response. Some contracts are mingled*

*with this docstring.*

*@url http://www.amazon.com/s?field-keywords=selfish+gene*

*@returns items 1 16*

*@returns requests 0 0*

*@scrapes Title Author Year Price*

*"""*

**@url** url

**@cb\_kwargs** {"arg1": "value1", "arg2": "value2", ...}

**@returns** item(s)|request(s) [min [max]]

**@scrapes** field\_1 field\_2 ...

SPIDER\_CONTRACTS = {

'myproject.contracts.ResponseCheck': 10,

'myproject.contracts.ItemValidate': 10,

}

**from** **scrapy.contracts** **import** Contract

**from** **scrapy.exceptions** **import** ContractFail

**class** **HasHeaderContract**(Contract):

*""" Demo contract which checks the presence of a custom header*

*@has\_header X-CustomHeader*

*"""*

name = 'has\_header'

**def** pre\_process(self, response):

**for** header **in** self.args:

**if** header **not** **in** response.headers:

**raise** ContractFail('X-CustomHeader not present')

**import** **os**

**import** **scrapy**

**class** **ExampleSpider**(scrapy.Spider):

name = 'example'

**def** \_\_init\_\_(self):

**if** os.environ.get('SCRAPY\_CHECK'):

**pass** *# Do some scraper adjustments when a check is running*

**맞춤형 계약**

SPIDER\_CONTRACTS = {

'myproject.contracts.ResponseCheck': 10,

'myproject.contracts.ItemValidate': 10,

}

**from** **scrapy.contracts** **import** Contract

**from** **scrapy.exceptions** **import** ContractFail

**class** **HasHeaderContract**(Contract):

*""" Demo contract which checks the presence of a custom header*

*@has\_header X-CustomHeader*

*"""*

name = 'has\_header'

**def** pre\_process(self, response):

**for** header **in** self.args:

**if** header **not** **in** response.headers:

**raise** ContractFail('X-CustomHeader not present')

**검사 실행 감지**

**import** **os**

**import** **scrapy**

**class** **ExampleSpider**(scrapy.Spider):

name = 'example'

**def** \_\_init\_\_(self):

**if** os.environ.get('SCRAPY\_CHECK'):

**pass** *# Do some scraper adjustments when a check is running*

[**일반적인 관행**](https://docs.scrapy.org/en/latest/topics/practices.html)

**import** **scrapy**

**from** **scrapy.crawler** **import** CrawlerProcess

**class** **MySpider**(scrapy.Spider):

*# Your spider definition*

...

process = CrawlerProcess(settings={

"FEEDS": {

"items.json": {"format": "json"},

},

})

process.crawl(MySpider)

process.start() *# the script will block here until the crawling is finished*

**from** **scrapy.crawler** **import** CrawlerProcess

**from** **scrapy.utils.project** **import** get\_project\_settings

process = CrawlerProcess(get\_project\_settings())

*# 'followall' is the name of one of the spiders of the project.*

process.crawl('followall', domain='scrapy.org')

process.start() *# the script will block here until the crawling is finished*

**from** **twisted.internet** **import** reactor

**import** **scrapy**

**from** **scrapy.crawler** **import** CrawlerRunner

**from** **scrapy.utils.log** **import** configure\_logging

**class** **MySpider**(scrapy.Spider):

*# Your spider definition*

...

configure\_logging({'LOG\_FORMAT': '*%(levelname)s*: *%(message)s*'})

runner = CrawlerRunner()

d = runner.crawl(MySpider)

d.addBoth(**lambda** \_: reactor.stop())

reactor.run() *# the script will block here until the crawling is finished*

**import** **scrapy**

**from** **scrapy.crawler** **import** CrawlerProcess

**from** **scrapy.utils.project** **import** get\_project\_settings

**class** **MySpider1**(scrapy.Spider):

*# Your first spider definition*

...

**class** **MySpider2**(scrapy.Spider):

*# Your second spider definition*

...

settings = get\_project\_settings()

process = CrawlerProcess(settings)

process.crawl(MySpider1)

process.crawl(MySpider2)

process.start() *# the script will block here until all crawling jobs are finished*

**import** **scrapy**

**from** **twisted.internet** **import** reactor

**from** **scrapy.crawler** **import** CrawlerRunner

**from** **scrapy.utils.log** **import** configure\_logging

**from** **scrapy.utils.project** **import** get\_project\_settings

**class** **MySpider1**(scrapy.Spider):

*# Your first spider definition*

...

**class** **MySpider2**(scrapy.Spider):

*# Your second spider definition*

...

configure\_logging()

settings = get\_project\_settings()

runner = CrawlerRunner(settings)

runner.crawl(MySpider1)

runner.crawl(MySpider2)

d = runner.join()

d.addBoth(**lambda** \_: reactor.stop())

reactor.run() *# the script will block here until all crawling jobs are finished*

**from** **twisted.internet** **import** reactor, defer

**from** **scrapy.crawler** **import** CrawlerRunner

**from** **scrapy.utils.log** **import** configure\_logging

**from** **scrapy.utils.project** **import** get\_project\_settings

**class** **MySpider1**(scrapy.Spider):

*# Your first spider definition*

...

**class** **MySpider2**(scrapy.Spider):

*# Your second spider definition*

...

settings = get\_project\_settings()

configure\_logging(settings)

runner = CrawlerRunner(settings)

**@defer**.inlineCallbacks

**def** crawl():

**yield** runner.crawl(MySpider1)

**yield** runner.crawl(MySpider2)

reactor.stop()

crawl()

reactor.run() *# the script will block here until the last crawl call is finished*

http://somedomain.com/urls-to-crawl/spider1/part1.list

http://somedomain.com/urls-to-crawl/spider1/part2.list

http://somedomain.com/urls-to-crawl/spider1/part3.list

curl http://scrapy1.mycompany.com:6800/schedule.json -d project=myproject -d spider=spider1 -d part=1

curl http://scrapy2.mycompany.com:6800/schedule.json -d project=myproject -d spider=spider1 -d part=2

curl http://scrapy3.mycompany.com:6800/schedule.json -d project=myproject -d spider=spider1 -d part=3

**스크립트에서 Scrapy 실행**

**import** **scrapy**

**from** **scrapy.crawler** **import** CrawlerProcess

**class** **MySpider**(scrapy.Spider):

*# Your spider definition*

...

process = CrawlerProcess(settings={

"FEEDS": {

"items.json": {"format": "json"},

},

})

process.crawl(MySpider)

process.start() *# the script will block here until the crawling is finished*

**from** **scrapy.crawler** **import** CrawlerProcess

**from** **scrapy.utils.project** **import** get\_project\_settings

process = CrawlerProcess(get\_project\_settings())

*# 'followall' is the name of one of the spiders of the project.*

process.crawl('followall', domain='scrapy.org')

process.start() *# the script will block here until the crawling is finished*

**from** **twisted.internet** **import** reactor

**import** **scrapy**

**from** **scrapy.crawler** **import** CrawlerRunner

**from** **scrapy.utils.log** **import** configure\_logging

**class** **MySpider**(scrapy.Spider):

*# Your spider definition*

...

configure\_logging({'LOG\_FORMAT': '*%(levelname)s*: *%(message)s*'})

runner = CrawlerRunner()

d = runner.crawl(MySpider)

d.addBoth(**lambda** \_: reactor.stop())

reactor.run() *# the script will block here until the crawling is finished*

**동일한 프로세스에서 여러 스파이더 실행**

**import** **scrapy**

**from** **scrapy.crawler** **import** CrawlerProcess

**from** **scrapy.utils.project** **import** get\_project\_settings

**class** **MySpider1**(scrapy.Spider):

*# Your first spider definition*

...

**class** **MySpider2**(scrapy.Spider):

*# Your second spider definition*

...

settings = get\_project\_settings()

process = CrawlerProcess(settings)

process.crawl(MySpider1)

process.crawl(MySpider2)

process.start() *# the script will block here until all crawling jobs are finished*

**import** **scrapy**

**from** **twisted.internet** **import** reactor

**from** **scrapy.crawler** **import** CrawlerRunner

**from** **scrapy.utils.log** **import** configure\_logging

**from** **scrapy.utils.project** **import** get\_project\_settings

**class** **MySpider1**(scrapy.Spider):

*# Your first spider definition*

...

**class** **MySpider2**(scrapy.Spider):

*# Your second spider definition*

...

configure\_logging()

settings = get\_project\_settings()

runner = CrawlerRunner(settings)

runner.crawl(MySpider1)

runner.crawl(MySpider2)

d = runner.join()

d.addBoth(**lambda** \_: reactor.stop())

reactor.run() *# the script will block here until all crawling jobs are finished*

**from** **twisted.internet** **import** reactor, defer

**from** **scrapy.crawler** **import** CrawlerRunner

**from** **scrapy.utils.log** **import** configure\_logging

**from** **scrapy.utils.project** **import** get\_project\_settings

**class** **MySpider1**(scrapy.Spider):

*# Your first spider definition*

...

**class** **MySpider2**(scrapy.Spider):

*# Your second spider definition*

...

settings = get\_project\_settings()

configure\_logging(settings)

runner = CrawlerRunner(settings)

**@defer**.inlineCallbacks

**def** crawl():

**yield** runner.crawl(MySpider1)

**yield** runner.crawl(MySpider2)

reactor.stop()

crawl()

reactor.run() *# the script will block here until the last crawl call is finished*

**분산 크롤링**

http://somedomain.com/urls-to-crawl/spider1/part1.list

http://somedomain.com/urls-to-crawl/spider1/part2.list

http://somedomain.com/urls-to-crawl/spider1/part3.list

curl http://scrapy1.mycompany.com:6800/schedule.json -d project=myproject -d spider=spider1 -d part=1

curl http://scrapy2.mycompany.com:6800/schedule.json -d project=myproject -d spider=spider1 -d part=2

curl http://scrapy3.mycompany.com:6800/schedule.json -d project=myproject -d spider=spider1 -d part=3

**금지를 피하기**

[**광범위한 크롤링**](https://docs.scrapy.org/en/latest/topics/broad-crawls.html)

SCHEDULER\_PRIORITY\_QUEUE = 'scrapy.pqueues.DownloaderAwarePriorityQueue'

CONCURRENT\_REQUESTS = 100

REACTOR\_THREADPOOL\_MAXSIZE = 20

LOG\_LEVEL = 'INFO'

COOKIES\_ENABLED = **False**

RETRY\_ENABLED = **False**

DOWNLOAD\_TIMEOUT = 15

REDIRECT\_ENABLED = **False**

AJAXCRAWL\_ENABLED = **True**

**올바른 SCHEDULER\_PRIORITY\_QUEUE 사용**

SCHEDULER\_PRIORITY\_QUEUE = 'scrapy.pqueues.DownloaderAwarePriorityQueue'

**동시성 증가**

CONCURRENT\_REQUESTS = 100

**Twisted IO 스레드 풀 최대 크기 늘리기**

REACTOR\_THREADPOOL\_MAXSIZE = 20

**자신의 DNS 설정**

**로그 수준 줄이기**

LOG\_LEVEL = 'INFO'

**쿠키 비활성화**

COOKIES\_ENABLED = **False**

**재시도 비활성화**

RETRY\_ENABLED = **False**

**다운로드 시간 초과 줄이기**

DOWNLOAD\_TIMEOUT = 15

**리디렉션 비활성화**

REDIRECT\_ENABLED = **False**

**"Ajax 크롤링 가능한 페이지" 크롤링 활성화**

AJAXCRAWL\_ENABLED = **True**

**BFO 순서로 크롤링**

**메모리 누수에 주의**

**특정 Twisted Reactor 설치**

[**스크래핑을 위해 브라우저의 개발자 도구 사용**](https://docs.scrapy.org/en/latest/topics/developer-tools.html)

**import** **scrapy**

**import** **json**

**class** **QuoteSpider**(scrapy.Spider):

name = 'quote'

allowed\_domains = ['quotes.toscrape.com']

page = 1

start\_urls = ['https://quotes.toscrape.com/api/quotes?page=1']

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

data = json.loads(response.text)

**for** quote **in** data["quotes"]:

**yield** {"quote": quote["text"]}

**if** data["has\_next"]:

self.page += 1

url = f"https://quotes.toscrape.com/api/quotes?page=*{*self.page*}*"

**yield** scrapy.Request(url=url, callback=self.parse)

**from** **scrapy** **import** Request

request = Request.from\_curl(

"curl 'https://quotes.toscrape.com/api/quotes?page=1' -H 'User-Agent: Mozil"

"la/5.0 (X11; Linux x86\_64; rv:67.0) Gecko/20100101 Firefox/67.0' -H 'Acce"

"pt: \*/\*' -H 'Accept-Language: ca,en-US;q=0.7,en;q=0.3' --compressed -H 'X"

"-Requested-With: XMLHttpRequest' -H 'Proxy-Authorization: Basic QFRLLTAzM"

"zEwZTAxLTk5MWUtNDFiNC1iZWRmLTJjNGI4M2ZiNDBmNDpAVEstMDMzMTBlMDEtOTkxZS00MW"

"I0LWJlZGYtMmM0YjgzZmI0MGY0' -H 'Connection: keep-alive' -H 'Referer: http"

"://quotes.toscrape.com/scroll' -H 'Cache-Control: max-age=0'")

**라이브 브라우저 DOM 검사 시 주의 사항**

**웹사이트 검사**

**네트워크 도구**

**import** **scrapy**

**import** **json**

**class** **QuoteSpider**(scrapy.Spider):

name = 'quote'

allowed\_domains = ['quotes.toscrape.com']

page = 1

start\_urls = ['https://quotes.toscrape.com/api/quotes?page=1']

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

data = json.loads(response.text)

**for** quote **in** data["quotes"]:

**yield** {"quote": quote["text"]}

**if** data["has\_next"]:

self.page += 1

url = f"https://quotes.toscrape.com/api/quotes?page=*{*self.page*}*"

**yield** scrapy.Request(url=url, callback=self.parse)

**from** **scrapy** **import** Request

request = Request.from\_curl(

"curl 'https://quotes.toscrape.com/api/quotes?page=1' -H 'User-Agent: Mozil"

"la/5.0 (X11; Linux x86\_64; rv:67.0) Gecko/20100101 Firefox/67.0' -H 'Acce"

"pt: \*/\*' -H 'Accept-Language: ca,en-US;q=0.7,en;q=0.3' --compressed -H 'X"

"-Requested-With: XMLHttpRequest' -H 'Proxy-Authorization: Basic QFRLLTAzM"

"zEwZTAxLTk5MWUtNDFiNC1iZWRmLTJjNGI4M2ZiNDBmNDpAVEstMDMzMTBlMDEtOTkxZS00MW"

"I0LWJlZGYtMmM0YjgzZmI0MGY0' -H 'Connection: keep-alive' -H 'Referer: http"

"://quotes.toscrape.com/scroll' -H 'Cache-Control: max-age=0'")

[**동적으로 로드된 콘텐츠 선택**](https://docs.scrapy.org/en/latest/topics/dynamic-content.html)

scrapy fetch --nolog https://example.com > response.html

data = json.loads(response.text)

selector = Selector(data['html'])

**import** **scrapy**

**from** **playwright.async\_api** **import** async\_playwright

**class** **PlaywrightSpider**(scrapy.Spider):

name = "playwright"

start\_urls = ["data:,"] *# avoid using the default Scrapy downloader*

**async** **def** parse(self, response):

**async** **with** async\_playwright() **as** pw:

browser = **await** pw.chromium.launch()

page = **await** browser.new\_page()

**await** page.goto("https:/example.org")

title = **await** page.title()

**return** {"title": title}

**데이터 소스 찾기**

**웹 페이지의 소스 코드 검사**

scrapy fetch --nolog https://example.com > response.html

**재생 요청**

**다양한 응답 형식 처리**

data = json.loads(response.text)

selector = Selector(data['html'])

**자바스크립트 코드 파싱**

**사전 렌더링 JavaScript**

**헤드리스 브라우저 사용**

**import** **scrapy**

**from** **playwright.async\_api** **import** async\_playwright

**class** **PlaywrightSpider**(scrapy.Spider):

name = "playwright"

start\_urls = ["data:,"] *# avoid using the default Scrapy downloader*

**async** **def** parse(self, response):

**async** **with** async\_playwright() **as** pw:

browser = **await** pw.chromium.launch()

page = **await** browser.new\_page()

**await** page.goto("https:/example.org")

title = **await** page.title()

**return** {"title": title}

[**메모리 누수 디버깅**](https://docs.scrapy.org/en/latest/topics/leaks.html)

telnet localhost 6023

>>> prefs()

Live References

ExampleSpider 1 oldest: 15s ago

HtmlResponse 10 oldest: 1s ago

Selector 2 oldest: 0s ago

FormRequest 878 oldest: 7s ago

**return** Request(f"http://www.somenastyspider.com/product.php?pid=*{*product\_id*}*",

callback=self.parse, cb\_kwargs={'referer': response})

**>>>** prefs()

Live References

SomenastySpider 1 oldest: 15s ago

HtmlResponse 3890 oldest: 265s ago

Selector 2 oldest: 0s ago

Request 3878 oldest: 250s ago

pip install Pympler

**메모리 누수의 일반적인 원인**

**요청이 너무 많습니까?**

**trackref로 메모리 누수 디버깅**

telnet localhost 6023

>>> prefs()

Live References

ExampleSpider 1 oldest: 15s ago

HtmlResponse 10 oldest: 1s ago

Selector 2 oldest: 0s ago

FormRequest 878 oldest: 7s ago

**return** Request(f"http://www.somenastyspider.com/product.php?pid=*{*product\_id*}*",

callback=self.parse, cb\_kwargs={'referer': response})

**>>>** prefs()

Live References

SomenastySpider 1 oldest: 15s ago

HtmlResponse 3890 oldest: 265s ago

Selector 2 oldest: 0s ago

Request 3878 oldest: 250s ago

**어떤 개체가 추적됩니까?**

**실제 예**

**return** Request(f"http://www.somenastyspider.com/product.php?pid=*{*product\_id*}*",

callback=self.parse, cb\_kwargs={'referer': response})

**>>>** prefs()

Live References

SomenastySpider 1 oldest: 15s ago

HtmlResponse 3890 oldest: 265s ago

Selector 2 oldest: 0s ago

Request 3878 oldest: 250s ago

**거미가 너무 많습니까?**

**scrapy.utils.trackref 모듈**

**muppy로 메모리 누수 디버깅**

pip install Pympler

**누출 없는 누출**

[**파일 및 이미지 다운로드 및 처리**](https://docs.scrapy.org/en/latest/topics/media-pipeline.html)

ITEM\_PIPELINES = {'scrapy.pipelines.images.ImagesPipeline': 1}

ITEM\_PIPELINES = {'scrapy.pipelines.files.FilesPipeline': 1}

FILES\_STORE = '/path/to/valid/dir'

IMAGES\_STORE = '/path/to/valid/dir'

http://www.example.com/image.jpg

3afec3b4765f8f0a07b78f98c07b83f013567a0a

3afec3b4765f8f0a07b78f98c07b83f013567a0a.jpg

http://www.example.com/product/images/large/front/0000000004166

00b08510e4\_front.jpg

<IMAGES\_STORE>/full/<FILE\_NAME>

ftp://username:password**@address**:port/path

ftp://address:port/path

IMAGES\_STORE = 's3://bucket/images'

IMAGES\_STORE\_S3\_ACL = 'public-read'

AWS\_ENDPOINT\_URL = 'http://minio.example.com:9000'

AWS\_USE\_SSL = **False** *# or True (None by default)*

AWS\_VERIFY = **False** *# or True (None by default)*

IMAGES\_STORE = 'gs://bucket/images/'

GCS\_PROJECT\_ID = 'project\_id'

IMAGES\_STORE\_GCS\_ACL = 'publicRead'

**import** **scrapy**

**class** **MyItem**(scrapy.Item):

*# ... other item fields ...*

image\_urls = scrapy.Field()

images = scrapy.Field()

FILES\_URLS\_FIELD = 'field\_name\_for\_your\_files\_urls'

FILES\_RESULT\_FIELD = 'field\_name\_for\_your\_processed\_files'

IMAGES\_URLS\_FIELD = 'field\_name\_for\_your\_images\_urls'

IMAGES\_RESULT\_FIELD = 'field\_name\_for\_your\_processed\_images'

*# 120 days of delay for files expiration*

FILES\_EXPIRES = 120

*# 30 days of delay for images expiration*

IMAGES\_EXPIRES = 30

IMAGES\_THUMBS = {

'small': (50, 50),

'big': (270, 270),

}

<IMAGES\_STORE>/thumbs/<size\_name>/<image\_id>.jpg

<IMAGES\_STORE>/full/63bbfea82b8880ed33cdb762aa11fab722a90a24.jpg

<IMAGES\_STORE>/thumbs/small/63bbfea82b8880ed33cdb762aa11fab722a90a24.jpg

<IMAGES\_STORE>/thumbs/big/63bbfea82b8880ed33cdb762aa11fab722a90a24.jpg

IMAGES\_MIN\_HEIGHT = 110

IMAGES\_MIN\_WIDTH = 110

MEDIA\_ALLOW\_REDIRECTS = **True**

**import** **os**

**from** **urllib.parse** **import** urlparse

**from** **scrapy.pipelines.files** **import** FilesPipeline

**class** **MyFilesPipeline**(FilesPipeline):

**def** file\_path(self, request, response=**None**, info=**None**, \*, item=**None**):

**return** 'files/' + os.path.basename(urlparse(request.url).path)

**from** **itemadapter** **import** ItemAdapter

**def** get\_media\_requests(self, item, info):

adapter = ItemAdapter(item)

**for** file\_url **in** adapter['file\_urls']:

**yield** scrapy.Request(file\_url)

[(**True**,

{'checksum': '2b00042f7481c7b056c4b410d28f33cf',

'path': 'full/0a79c461a4062ac383dc4fade7bc09f1384a3910.jpg',

'url': 'http://www.example.com/files/product1.pdf',

'status': 'downloaded'}),

(**False**,

Failure(...))]

**from** **itemadapter** **import** ItemAdapter

**from** **scrapy.exceptions** **import** DropItem

**def** item\_completed(self, results, item, info):

file\_paths = [x['path'] **for** ok, x **in** results **if** ok]

**if** **not** file\_paths:

**raise** DropItem("Item contains no files")

adapter = ItemAdapter(item)

adapter['file\_paths'] = file\_paths

**return** item

**import** **os**

**from** **urllib.parse** **import** urlparse

**from** **scrapy.pipelines.images** **import** ImagesPipeline

**class** **MyImagesPipeline**(ImagesPipeline):

**def** file\_path(self, request, response=**None**, info=**None**, \*, item=**None**):

**return** 'files/' + os.path.basename(urlparse(request.url).path)

**import** **scrapy**

**from** **itemadapter** **import** ItemAdapter

**from** **scrapy.exceptions** **import** DropItem

**from** **scrapy.pipelines.images** **import** ImagesPipeline

**class** **MyImagesPipeline**(ImagesPipeline):

**def** get\_media\_requests(self, item, info):

**for** image\_url **in** item['image\_urls']:

**yield** scrapy.Request(image\_url)

**def** item\_completed(self, results, item, info):

image\_paths = [x['path'] **for** ok, x **in** results **if** ok]

**if** **not** image\_paths:

**raise** DropItem("Item contains no images")

adapter = ItemAdapter(item)

adapter['image\_paths'] = image\_paths

**return** item

ITEM\_PIPELINES = {

'myproject.pipelines.MyImagesPipeline': 300

}

**파일 파이프라인 사용**

**이미지 파이프라인 사용**

**미디어 파이프라인 활성화**

ITEM\_PIPELINES = {'scrapy.pipelines.images.ImagesPipeline': 1}

ITEM\_PIPELINES = {'scrapy.pipelines.files.FilesPipeline': 1}

FILES\_STORE = '/path/to/valid/dir'

IMAGES\_STORE = '/path/to/valid/dir'

**파일 이름 지정**

http://www.example.com/image.jpg

3afec3b4765f8f0a07b78f98c07b83f013567a0a

3afec3b4765f8f0a07b78f98c07b83f013567a0a.jpg

http://www.example.com/product/images/large/front/0000000004166

00b08510e4\_front.jpg

**기본 파일 이름 지정**

http://www.example.com/image.jpg

3afec3b4765f8f0a07b78f98c07b83f013567a0a

3afec3b4765f8f0a07b78f98c07b83f013567a0a.jpg

**사용자 지정 파일 이름 지정**

http://www.example.com/product/images/large/front/0000000004166

00b08510e4\_front.jpg

**지원되는 스토리지**

<IMAGES\_STORE>/full/<FILE\_NAME>

ftp://username:password**@address**:port/path

ftp://address:port/path

IMAGES\_STORE = 's3://bucket/images'

IMAGES\_STORE\_S3\_ACL = 'public-read'

AWS\_ENDPOINT\_URL = 'http://minio.example.com:9000'

AWS\_USE\_SSL = **False** *# or True (None by default)*

AWS\_VERIFY = **False** *# or True (None by default)*

IMAGES\_STORE = 'gs://bucket/images/'

GCS\_PROJECT\_ID = 'project\_id'

IMAGES\_STORE\_GCS\_ACL = 'publicRead'

**파일 시스템 스토리지**

<IMAGES\_STORE>/full/<FILE\_NAME>

**FTP 서버 저장**

ftp://username:password**@address**:port/path

ftp://address:port/path

**아마존 S3 스토리지**

IMAGES\_STORE = 's3://bucket/images'

IMAGES\_STORE\_S3\_ACL = 'public-read'

AWS\_ENDPOINT\_URL = 'http://minio.example.com:9000'

AWS\_USE\_SSL = **False** *# or True (None by default)*

AWS\_VERIFY = **False** *# or True (None by default)*

**구글 클라우드 스토리지**

IMAGES\_STORE = 'gs://bucket/images/'

GCS\_PROJECT\_ID = 'project\_id'

IMAGES\_STORE\_GCS\_ACL = 'publicRead'

**사용 예**

**import** **scrapy**

**class** **MyItem**(scrapy.Item):

*# ... other item fields ...*

image\_urls = scrapy.Field()

images = scrapy.Field()

FILES\_URLS\_FIELD = 'field\_name\_for\_your\_files\_urls'

FILES\_RESULT\_FIELD = 'field\_name\_for\_your\_processed\_files'

IMAGES\_URLS\_FIELD = 'field\_name\_for\_your\_images\_urls'

IMAGES\_RESULT\_FIELD = 'field\_name\_for\_your\_processed\_images'

**추가 기능**

*# 120 days of delay for files expiration*

FILES\_EXPIRES = 120

*# 30 days of delay for images expiration*

IMAGES\_EXPIRES = 30

IMAGES\_THUMBS = {

'small': (50, 50),

'big': (270, 270),

}

<IMAGES\_STORE>/thumbs/<size\_name>/<image\_id>.jpg

<IMAGES\_STORE>/full/63bbfea82b8880ed33cdb762aa11fab722a90a24.jpg

<IMAGES\_STORE>/thumbs/small/63bbfea82b8880ed33cdb762aa11fab722a90a24.jpg

<IMAGES\_STORE>/thumbs/big/63bbfea82b8880ed33cdb762aa11fab722a90a24.jpg

IMAGES\_MIN\_HEIGHT = 110

IMAGES\_MIN\_WIDTH = 110

MEDIA\_ALLOW\_REDIRECTS = **True**

**파일 만료**

*# 120 days of delay for files expiration*

FILES\_EXPIRES = 120

*# 30 days of delay for images expiration*

IMAGES\_EXPIRES = 30

**이미지의 썸네일 생성**

IMAGES\_THUMBS = {

'small': (50, 50),

'big': (270, 270),

}

<IMAGES\_STORE>/thumbs/<size\_name>/<image\_id>.jpg

<IMAGES\_STORE>/full/63bbfea82b8880ed33cdb762aa11fab722a90a24.jpg

<IMAGES\_STORE>/thumbs/small/63bbfea82b8880ed33cdb762aa11fab722a90a24.jpg

<IMAGES\_STORE>/thumbs/big/63bbfea82b8880ed33cdb762aa11fab722a90a24.jpg

**작은 이미지 필터링**

IMAGES\_MIN\_HEIGHT = 110

IMAGES\_MIN\_WIDTH = 110

**리디렉션 허용**

MEDIA\_ALLOW\_REDIRECTS = **True**

**미디어 파이프라인 확장**

**import** **os**

**from** **urllib.parse** **import** urlparse

**from** **scrapy.pipelines.files** **import** FilesPipeline

**class** **MyFilesPipeline**(FilesPipeline):

**def** file\_path(self, request, response=**None**, info=**None**, \*, item=**None**):

**return** 'files/' + os.path.basename(urlparse(request.url).path)

**from** **itemadapter** **import** ItemAdapter

**def** get\_media\_requests(self, item, info):

adapter = ItemAdapter(item)

**for** file\_url **in** adapter['file\_urls']:

**yield** scrapy.Request(file\_url)

[(**True**,

{'checksum': '2b00042f7481c7b056c4b410d28f33cf',

'path': 'full/0a79c461a4062ac383dc4fade7bc09f1384a3910.jpg',

'url': 'http://www.example.com/files/product1.pdf',

'status': 'downloaded'}),

(**False**,

Failure(...))]

**from** **itemadapter** **import** ItemAdapter

**from** **scrapy.exceptions** **import** DropItem

**def** item\_completed(self, results, item, info):

file\_paths = [x['path'] **for** ok, x **in** results **if** ok]

**if** **not** file\_paths:

**raise** DropItem("Item contains no files")

adapter = ItemAdapter(item)

adapter['file\_paths'] = file\_paths

**return** item

**import** **os**

**from** **urllib.parse** **import** urlparse

**from** **scrapy.pipelines.images** **import** ImagesPipeline

**class** **MyImagesPipeline**(ImagesPipeline):

**def** file\_path(self, request, response=**None**, info=**None**, \*, item=**None**):

**return** 'files/' + os.path.basename(urlparse(request.url).path)

**커스텀 이미지 파이프라인 예시**

**import** **scrapy**

**from** **itemadapter** **import** ItemAdapter

**from** **scrapy.exceptions** **import** DropItem

**from** **scrapy.pipelines.images** **import** ImagesPipeline

**class** **MyImagesPipeline**(ImagesPipeline):

**def** get\_media\_requests(self, item, info):

**for** image\_url **in** item['image\_urls']:

**yield** scrapy.Request(image\_url)

**def** item\_completed(self, results, item, info):

image\_paths = [x['path'] **for** ok, x **in** results **if** ok]

**if** **not** image\_paths:

**raise** DropItem("Item contains no images")

adapter = ItemAdapter(item)

adapter['image\_paths'] = image\_paths

**return** item

ITEM\_PIPELINES = {

'myproject.pipelines.MyImagesPipeline': 300

}

[**거미 배포**](https://docs.scrapy.org/en/latest/topics/deploy.html)

**Scrapyd 서버에 배포**

**Zyte Scrapy Cloud에 배포**

[**자동 조절 확장**](https://docs.scrapy.org/en/latest/topics/autothrottle.html)

**디자인 목표**

**작동 방식**

**조절 알고리즘**

**설정**

**AUTOTHROTTLE\_ENABLED**

**AUTOTHROTTLE\_START\_DELAY**

**AUTOTHROTTLE\_MAX\_DELAY**

**AUTOTHROTTLE\_TARGET\_CONCURRENCY**

**AUTOTHROTTLE\_DEBUG**

[**벤치마킹**](https://docs.scrapy.org/en/latest/topics/benchmarking.html)

scrapy bench

2016-12-16 21:18:48 [scrapy.utils.log] INFO: Scrapy 1.2.2 started (bot: quotesbot)

2016-12-16 21:18:48 [scrapy.utils.log] INFO: Overridden settings: {'CLOSESPIDER\_TIMEOUT': 10, 'ROBOTSTXT\_OBEY': **True**, 'SPIDER\_MODULES': ['quotesbot.spiders'], 'LOGSTATS\_INTERVAL': 1, 'BOT\_NAME': 'quotesbot', 'LOG\_LEVEL': 'INFO', 'NEWSPIDER\_MODULE': 'quotesbot.spiders'}

2016-12-16 21:18:49 [scrapy.middleware] INFO: Enabled extensions:

['scrapy.extensions.closespider.CloseSpider',

'scrapy.extensions.logstats.LogStats',

'scrapy.extensions.telnet.TelnetConsole',

'scrapy.extensions.corestats.CoreStats']

2016-12-16 21:18:49 [scrapy.middleware] INFO: Enabled downloader middlewares:

['scrapy.downloadermiddlewares.robotstxt.RobotsTxtMiddleware',

'scrapy.downloadermiddlewares.httpauth.HttpAuthMiddleware',

'scrapy.downloadermiddlewares.downloadtimeout.DownloadTimeoutMiddleware',

'scrapy.downloadermiddlewares.defaultheaders.DefaultHeadersMiddleware',

'scrapy.downloadermiddlewares.useragent.UserAgentMiddleware',

'scrapy.downloadermiddlewares.retry.RetryMiddleware',

'scrapy.downloadermiddlewares.redirect.MetaRefreshMiddleware',

'scrapy.downloadermiddlewares.httpcompression.HttpCompressionMiddleware',

'scrapy.downloadermiddlewares.redirect.RedirectMiddleware',

'scrapy.downloadermiddlewares.cookies.CookiesMiddleware',

'scrapy.downloadermiddlewares.stats.DownloaderStats']

2016-12-16 21:18:49 [scrapy.middleware] INFO: Enabled spider middlewares:

['scrapy.spidermiddlewares.httperror.HttpErrorMiddleware',

'scrapy.spidermiddlewares.offsite.OffsiteMiddleware',

'scrapy.spidermiddlewares.referer.RefererMiddleware',

'scrapy.spidermiddlewares.urllength.UrlLengthMiddleware',

'scrapy.spidermiddlewares.depth.DepthMiddleware']

2016-12-16 21:18:49 [scrapy.middleware] INFO: Enabled item pipelines:

[]

2016-12-16 21:18:49 [scrapy.core.engine] INFO: Spider opened

2016-12-16 21:18:49 [scrapy.extensions.logstats] INFO: Crawled 0 pages (at 0 pages/min), scraped 0 items (at 0 items/min)

2016-12-16 21:18:50 [scrapy.extensions.logstats] INFO: Crawled 70 pages (at 4200 pages/min), scraped 0 items (at 0 items/min)

2016-12-16 21:18:51 [scrapy.extensions.logstats] INFO: Crawled 134 pages (at 3840 pages/min), scraped 0 items (at 0 items/min)

2016-12-16 21:18:52 [scrapy.extensions.logstats] INFO: Crawled 198 pages (at 3840 pages/min), scraped 0 items (at 0 items/min)

2016-12-16 21:18:53 [scrapy.extensions.logstats] INFO: Crawled 254 pages (at 3360 pages/min), scraped 0 items (at 0 items/min)

2016-12-16 21:18:54 [scrapy.extensions.logstats] INFO: Crawled 302 pages (at 2880 pages/min), scraped 0 items (at 0 items/min)

2016-12-16 21:18:55 [scrapy.extensions.logstats] INFO: Crawled 358 pages (at 3360 pages/min), scraped 0 items (at 0 items/min)

2016-12-16 21:18:56 [scrapy.extensions.logstats] INFO: Crawled 406 pages (at 2880 pages/min), scraped 0 items (at 0 items/min)

2016-12-16 21:18:57 [scrapy.extensions.logstats] INFO: Crawled 438 pages (at 1920 pages/min), scraped 0 items (at 0 items/min)

2016-12-16 21:18:58 [scrapy.extensions.logstats] INFO: Crawled 470 pages (at 1920 pages/min), scraped 0 items (at 0 items/min)

2016-12-16 21:18:59 [scrapy.core.engine] INFO: Closing spider (closespider\_timeout)

2016-12-16 21:18:59 [scrapy.extensions.logstats] INFO: Crawled 518 pages (at 2880 pages/min), scraped 0 items (at 0 items/min)

2016-12-16 21:19:00 [scrapy.statscollectors] INFO: Dumping Scrapy stats:

{'downloader/request\_bytes': 229995,

'downloader/request\_count': 534,

'downloader/request\_method\_count/GET': 534,

'downloader/response\_bytes': 1565504,

'downloader/response\_count': 534,

'downloader/response\_status\_count/200': 534,

'finish\_reason': 'closespider\_timeout',

'finish\_time': datetime.datetime(2016, 12, 16, 16, 19, 0, 647725),

'log\_count/INFO': 17,

'request\_depth\_max': 19,

'response\_received\_count': 534,

'scheduler/dequeued': 533,

'scheduler/dequeued/memory': 533,

'scheduler/enqueued': 10661,

'scheduler/enqueued/memory': 10661,

'start\_time': datetime.datetime(2016, 12, 16, 16, 18, 49, 799869)}

2016-12-16 21:19:00 [scrapy.core.engine] INFO: Spider closed (closespider\_timeout)

[**작업: 크롤링 일시 중지 및 재개**](https://docs.scrapy.org/en/latest/topics/jobs.html)

scrapy crawl somespider -s JOBDIR=crawls/somespider-1

scrapy crawl somespider -s JOBDIR=crawls/somespider-1

**def** parse\_item(self, response):

*# parse item here*

self.state['items\_count'] = self.state.get('items\_count', 0) + 1

**작업 디렉토리**

**사용 방법**

scrapy crawl somespider -s JOBDIR=crawls/somespider-1

scrapy crawl somespider -s JOBDIR=crawls/somespider-1

**배치 간에 지속적인 상태 유지**

**def** parse\_item(self, response):

*# parse item here*

self.state['items\_count'] = self.state.get('items\_count', 0) + 1

**지속성 문제**

**쿠키 만료**

**직렬화 요청**

[**코루틴**](https://docs.scrapy.org/en/latest/topics/coroutines.html)

**from** **itemadapter** **import** ItemAdapter

**class** **DbPipeline**:

**def** \_update\_item(self, data, item):

adapter = ItemAdapter(item)

adapter['field'] = data

**return** item

**def** process\_item(self, item, spider):

adapter = ItemAdapter(item)

dfd = db.get\_some\_data(adapter['id'])

dfd.addCallback(self.\_update\_item, item)

**return** dfd

**from** **itemadapter** **import** ItemAdapter

**class** **DbPipeline**:

**async** **def** process\_item(self, item, spider):

adapter = ItemAdapter(item)

adapter['field'] = **await** db.get\_some\_data(adapter['id'])

**return** item

**class** **MySpiderDeferred**(Spider):

*# ...*

**async** **def** parse(self, response):

additional\_response = **await** treq.get('https://additional.url')

additional\_data = **await** treq.content(additional\_response)

*# ... use response and additional\_data to yield items and requests*

**class** **MySpiderAsyncio**(Spider):

*# ...*

**async** **def** parse(self, response):

**async** **with** aiohttp.ClientSession() **as** session:

**async** **with** session.get('https://additional.url') **as** additional\_response:

additional\_data = **await** additional\_response.text()

*# ... use response and additional\_data to yield items and requests*

**class** **UniversalSpiderMiddleware**:

**def** process\_spider\_output(self, response, result, spider):

**for** r **in** result:

*# ... do something with r*

**yield** r

**async** **def** process\_spider\_output\_async(self, response, result, spider):

**async** **for** r **in** result:

*# ... do something with r*

**yield** r

**지원되는 호출 가능**

**일반적인 사용법**

**from** **itemadapter** **import** ItemAdapter

**class** **DbPipeline**:

**def** \_update\_item(self, data, item):

adapter = ItemAdapter(item)

adapter['field'] = data

**return** item

**def** process\_item(self, item, spider):

adapter = ItemAdapter(item)

dfd = db.get\_some\_data(adapter['id'])

dfd.addCallback(self.\_update\_item, item)

**return** dfd

**from** **itemadapter** **import** ItemAdapter

**class** **DbPipeline**:

**async** **def** process\_item(self, item, spider):

adapter = ItemAdapter(item)

adapter['field'] = **await** db.get\_some\_data(adapter['id'])

**return** item

**class** **MySpiderDeferred**(Spider):

*# ...*

**async** **def** parse(self, response):

additional\_response = **await** treq.get('https://additional.url')

additional\_data = **await** treq.content(additional\_response)

*# ... use response and additional\_data to yield items and requests*

**class** **MySpiderAsyncio**(Spider):

*# ...*

**async** **def** parse(self, response):

**async** **with** aiohttp.ClientSession() **as** session:

**async** **with** session.get('https://additional.url') **as** additional\_response:

additional\_data = **await** additional\_response.text()

*# ... use response and additional\_data to yield items and requests*

**동기 및 비동기 스파이더 미들웨어 혼합**

**유니버설 스파이더 미들웨어**

**class** **UniversalSpiderMiddleware**:

**def** process\_spider\_output(self, response, result, spider):

**for** r **in** result:

*# ... do something with r*

**yield** r

**async** **def** process\_spider\_output\_async(self, response, result, spider):

**async** **for** r **in** result:

*# ... do something with r*

**yield** r

[**비동기**](https://docs.scrapy.org/en/latest/topics/asyncio.html)

install\_reactor('twisted.internet.asyncioreactor.AsyncioSelectorReactor')

**import** **asyncio**

asyncio.set\_event\_loop\_policy(asyncio.WindowsSelectorEventLoopPolicy())

**class** **MySpider**(Spider):

...

**async** **def** parse(self, response):

d = treq.get('https://example.com/additional')

additional\_response = **await** deferred\_to\_future(d)

**class** **MySpider**(Spider):

...

**async** **def** parse(self, response):

d = treq.get('https://example.com/additional')

extra\_response = **await** maybe\_deferred\_to\_future(d)

**from** **scrapy.utils.reactor** **import** is\_asyncio\_reactor\_installed

**class** **MyComponent**:

**def** \_\_init\_\_(self):

**if** **not** is\_asyncio\_reactor\_installed():

**raise** ValueError(

f"*{*MyComponent.\_\_qualname\_\_*}* requires the asyncio Twisted "

f"reactor. Make sure you have it configured in the "

f"TWISTED\_REACTOR setting. See the asyncio documentation "

f"of Scrapy for more information."

)

**asyncio 리액터 설치**

install\_reactor('twisted.internet.asyncioreactor.AsyncioSelectorReactor')

**사용자 정의 asyncio 루프 사용**

**Windows 관련 참고 사항**

**import** **asyncio**

asyncio.set\_event\_loop\_policy(asyncio.WindowsSelectorEventLoopPolicy())

**지연 대기 중**

**class** **MySpider**(Spider):

...

**async** **def** parse(self, response):

d = treq.get('https://example.com/additional')

additional\_response = **await** deferred\_to\_future(d)

**class** **MySpider**(Spider):

...

**async** **def** parse(self, response):

d = treq.get('https://example.com/additional')

extra\_response = **await** maybe\_deferred\_to\_future(d)

**요구 사항으로 asyncio 적용**

**from** **scrapy.utils.reactor** **import** is\_asyncio\_reactor\_installed

**class** **MyComponent**:

**def** \_\_init\_\_(self):

**if** **not** is\_asyncio\_reactor\_installed():

**raise** ValueError(

f"*{*MyComponent.\_\_qualname\_\_*}* requires the asyncio Twisted "

f"reactor. Make sure you have it configured in the "

f"TWISTED\_REACTOR setting. See the asyncio documentation "

f"of Scrapy for more information."

)

[**아키텍처 개요**](https://docs.scrapy.org/en/latest/topics/architecture.html)

**개요**

**데이터 흐름**

**구성품**

**스크래피 엔진**

**스케줄러**

**다운로더**

**거미**

**아이템 파이프라인**

**다운로더 미들웨어**

**스파이더 미들웨어**

**이벤트 기반 네트워킹**

[**다운로더 미들웨어**](https://docs.scrapy.org/en/latest/topics/downloader-middleware.html)

DOWNLOADER\_MIDDLEWARES = {

'myproject.middlewares.CustomDownloaderMiddleware': 543,

}

DOWNLOADER\_MIDDLEWARES = {

'myproject.middlewares.CustomDownloaderMiddleware': 543,

'scrapy.downloadermiddlewares.useragent.UserAgentMiddleware': **None**,

}

**for** i, url **in** enumerate(urls):

**yield** scrapy.Request(url, meta={'cookiejar': i},

callback=self.parse\_page)

**def** parse\_page(self, response):

*# do some processing*

**return** scrapy.Request("http://www.example.com/otherpage",

meta={'cookiejar': response.meta['cookiejar']},

callback=self.parse\_other\_page)

2011-04-06 14:35:10-0300 [scrapy.core.engine] INFO: Spider opened

2011-04-06 14:35:10-0300 [scrapy.downloadermiddlewares.cookies] DEBUG: Sending cookies to: <GET http://www.diningcity.com/netherlands/index.html>

Cookie: clientlanguage\_nl=en\_EN

2011-04-06 14:35:14-0300 [scrapy.downloadermiddlewares.cookies] DEBUG: Received cookies from: <200 http://www.diningcity.com/netherlands/index.html>

Set-Cookie: JSESSIONID=B~FA4DC0C496C8762AE4F1A620EAB34F38; Path=/

Set-Cookie: ip\_isocode=US

Set-Cookie: clientlanguage\_nl=en\_EN; Expires=Thu, 07-Apr-2011 21:21:34 GMT; Path=/

2011-04-06 14:49:50-0300 [scrapy.core.engine] DEBUG: Crawled (200) <GET http://www.diningcity.com/netherlands/index.html> (referer: **None**)

[...]

**from** **scrapy.spiders** **import** CrawlSpider

**class** **SomeIntranetSiteSpider**(CrawlSpider):

http\_user = 'someuser'

http\_pass = 'somepass'

http\_auth\_domain = 'intranet.example.com'

name = 'intranet.example.com'

*# .. rest of the spider code omitted ...*

/path/to/cache/dir/example.com/72/72811f648e718090f041317756c03adb0ada46c7

**class** **MySpider**(CrawlSpider):

handle\_httpstatus\_list = [301, 302]

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

**if** **not** response.text:

new\_request\_or\_none = get\_retry\_request(

response.request,

spider=self,

reason='empty',

)

**return** new\_request\_or\_none

**다운로더 미들웨어 활성화**

DOWNLOADER\_MIDDLEWARES = {

'myproject.middlewares.CustomDownloaderMiddleware': 543,

}

DOWNLOADER\_MIDDLEWARES = {

'myproject.middlewares.CustomDownloaderMiddleware': 543,

'scrapy.downloadermiddlewares.useragent.UserAgentMiddleware': **None**,

}

**나만의 다운로더 미들웨어 작성**

**내장 다운로더 미들웨어 참조**

**for** i, url **in** enumerate(urls):

**yield** scrapy.Request(url, meta={'cookiejar': i},

callback=self.parse\_page)

**def** parse\_page(self, response):

*# do some processing*

**return** scrapy.Request("http://www.example.com/otherpage",

meta={'cookiejar': response.meta['cookiejar']},

callback=self.parse\_other\_page)

2011-04-06 14:35:10-0300 [scrapy.core.engine] INFO: Spider opened

2011-04-06 14:35:10-0300 [scrapy.downloadermiddlewares.cookies] DEBUG: Sending cookies to: <GET http://www.diningcity.com/netherlands/index.html>

Cookie: clientlanguage\_nl=en\_EN

2011-04-06 14:35:14-0300 [scrapy.downloadermiddlewares.cookies] DEBUG: Received cookies from: <200 http://www.diningcity.com/netherlands/index.html>

Set-Cookie: JSESSIONID=B~FA4DC0C496C8762AE4F1A620EAB34F38; Path=/

Set-Cookie: ip\_isocode=US

Set-Cookie: clientlanguage\_nl=en\_EN; Expires=Thu, 07-Apr-2011 21:21:34 GMT; Path=/

2011-04-06 14:49:50-0300 [scrapy.core.engine] DEBUG: Crawled (200) <GET http://www.diningcity.com/netherlands/index.html> (referer: **None**)

[...]

**from** **scrapy.spiders** **import** CrawlSpider

**class** **SomeIntranetSiteSpider**(CrawlSpider):

http\_user = 'someuser'

http\_pass = 'somepass'

http\_auth\_domain = 'intranet.example.com'

name = 'intranet.example.com'

*# .. rest of the spider code omitted ...*

/path/to/cache/dir/example.com/72/72811f648e718090f041317756c03adb0ada46c7

**class** **MySpider**(CrawlSpider):

handle\_httpstatus\_list = [301, 302]

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

**if** **not** response.text:

new\_request\_or\_none = get\_retry\_request(

response.request,

spider=self,

reason='empty',

)

**return** new\_request\_or\_none

**쿠키미들웨어**

**for** i, url **in** enumerate(urls):

**yield** scrapy.Request(url, meta={'cookiejar': i},

callback=self.parse\_page)

**def** parse\_page(self, response):

*# do some processing*

**return** scrapy.Request("http://www.example.com/otherpage",

meta={'cookiejar': response.meta['cookiejar']},

callback=self.parse\_other\_page)

2011-04-06 14:35:10-0300 [scrapy.core.engine] INFO: Spider opened

2011-04-06 14:35:10-0300 [scrapy.downloadermiddlewares.cookies] DEBUG: Sending cookies to: <GET http://www.diningcity.com/netherlands/index.html>

Cookie: clientlanguage\_nl=en\_EN

2011-04-06 14:35:14-0300 [scrapy.downloadermiddlewares.cookies] DEBUG: Received cookies from: <200 http://www.diningcity.com/netherlands/index.html>

Set-Cookie: JSESSIONID=B~FA4DC0C496C8762AE4F1A620EAB34F38; Path=/

Set-Cookie: ip\_isocode=US

Set-Cookie: clientlanguage\_nl=en\_EN; Expires=Thu, 07-Apr-2011 21:21:34 GMT; Path=/

2011-04-06 14:49:50-0300 [scrapy.core.engine] DEBUG: Crawled (200) <GET http://www.diningcity.com/netherlands/index.html> (referer: **None**)

[...]

**스파이더당 여러 쿠키 세션**

**for** i, url **in** enumerate(urls):

**yield** scrapy.Request(url, meta={'cookiejar': i},

callback=self.parse\_page)

**def** parse\_page(self, response):

*# do some processing*

**return** scrapy.Request("http://www.example.com/otherpage",

meta={'cookiejar': response.meta['cookiejar']},

callback=self.parse\_other\_page)

**COOKIES\_ENABLED**

**쿠키\_디버그**

2011-04-06 14:35:10-0300 [scrapy.core.engine] INFO: Spider opened

2011-04-06 14:35:10-0300 [scrapy.downloadermiddlewares.cookies] DEBUG: Sending cookies to: <GET http://www.diningcity.com/netherlands/index.html>

Cookie: clientlanguage\_nl=en\_EN

2011-04-06 14:35:14-0300 [scrapy.downloadermiddlewares.cookies] DEBUG: Received cookies from: <200 http://www.diningcity.com/netherlands/index.html>

Set-Cookie: JSESSIONID=B~FA4DC0C496C8762AE4F1A620EAB34F38; Path=/

Set-Cookie: ip\_isocode=US

Set-Cookie: clientlanguage\_nl=en\_EN; Expires=Thu, 07-Apr-2011 21:21:34 GMT; Path=/

2011-04-06 14:49:50-0300 [scrapy.core.engine] DEBUG: Crawled (200) <GET http://www.diningcity.com/netherlands/index.html> (referer: **None**)

[...]

**DefaultHeaders미들웨어**

**DownloadTimeout미들웨어**

**HttpAuth 미들웨어**

**from** **scrapy.spiders** **import** CrawlSpider

**class** **SomeIntranetSiteSpider**(CrawlSpider):

http\_user = 'someuser'

http\_pass = 'somepass'

http\_auth\_domain = 'intranet.example.com'

name = 'intranet.example.com'

*# .. rest of the spider code omitted ...*

**HttpCache 미들웨어**

/path/to/cache/dir/example.com/72/72811f648e718090f041317756c03adb0ada46c7

**더미 정책(기본값)**

**RFC2616 정책**

**파일 시스템 스토리지 백엔드(기본값)**

/path/to/cache/dir/example.com/72/72811f648e718090f041317756c03adb0ada46c7

**DBM 스토리지 백엔드**

**나만의 스토리지 백엔드 작성**

**HTTPCache 미들웨어 설정**

**HTTPCACHE\_ENABLED**

**HTTPCACHE\_EXPIRATION\_SECS**

**HTTPCACHE\_DIR**

**HTTPCACHE\_IGNORE\_HTTP\_CODES**

**HTTPCACHE\_IGNORE\_MISSING**

**HTTPCACHE\_IGNORE\_SCHEMES**

**HTTPCACHE\_STORAGE**

**HTTPCACHE\_DBM\_MODULE**

**HTTPCACHE\_POLICY**

**HTTPCACHE\_GZIP**

**HTTPCACHE\_ALWAYS\_STORE**

**HTTPCACHE\_IGNORE\_RESPONSE\_CACHE\_CONTROLS**

**HttpCompression 미들웨어**

**HttpCompressionMiddleware 설정**

**압축\_활성화됨**

**HttpProxy 미들웨어**

**리디렉션미들웨어**

**class** **MySpider**(CrawlSpider):

handle\_httpstatus\_list = [301, 302]

**리디렉션 미들웨어 설정**

**REDIRECT\_ENABLED**

**REDIRECT\_MAX\_TIMES**

**MetaRefresh미들웨어**

**MetaRefreshMiddleware 설정**

**METAREFRESH\_ENABLED**

**METAREFRESH\_IGNORE\_TAGS**

**METAREFRESH\_MAXDELAY**

**재시도 미들웨어**

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

**if** **not** response.text:

new\_request\_or\_none = get\_retry\_request(

response.request,

spider=self,

reason='empty',

)

**return** new\_request\_or\_none

**재시도 미들웨어 설정**

**RETRY\_ENABLED**

**RETRY\_TIMES**

**RETRY\_HTTP\_CODES**

**RETRY\_PRIORITY\_ADJUST**

**RobotsTxt미들웨어**

**프로테고 파서**

**RobotFileParser**

**레피 파서**

**Robotexclusionrulesparser**

**새 파서에 대한 지원 구현**

**다운로더 통계**

**UserAgent미들웨어**

**AjaxCrawl 미들웨어**

**AjaxCrawlMiddleware 설정**

**AJAXCRAWL\_ENABLED**

**HttpProxyMiddleware 설정**

**HTTPPROXY\_ENABLED**

**HTTPPROXY\_AUTH\_ENCODING**

[**스파이더 미들웨어**](https://docs.scrapy.org/en/latest/topics/spider-middleware.html)

SPIDER\_MIDDLEWARES = {

'myproject.middlewares.CustomSpiderMiddleware': 543,

}

SPIDER\_MIDDLEWARES = {

'myproject.middlewares.CustomSpiderMiddleware': 543,

'scrapy.spidermiddlewares.offsite.OffsiteMiddleware': **None**,

}

**class** **MySpider**(CrawlSpider):

handle\_httpstatus\_list = [404]

DEBUG: Filtered offsite request to 'www.othersite.com': <GET http://www.othersite.com/some/page.html>

**스파이더 미들웨어 활성화**

SPIDER\_MIDDLEWARES = {

'myproject.middlewares.CustomSpiderMiddleware': 543,

}

SPIDER\_MIDDLEWARES = {

'myproject.middlewares.CustomSpiderMiddleware': 543,

'scrapy.spidermiddlewares.offsite.OffsiteMiddleware': **None**,

}

**자신의 스파이더 미들웨어 작성**

**내장 스파이더 미들웨어 참조**

**class** **MySpider**(CrawlSpider):

handle\_httpstatus\_list = [404]

DEBUG: Filtered offsite request to 'www.othersite.com': <GET http://www.othersite.com/some/page.html>

**깊이 미들웨어**

**HttpError 미들웨어**

**class** **MySpider**(CrawlSpider):

handle\_httpstatus\_list = [404]

**HttpErrorMiddleware 설정**

**HTTPERROR\_ALLOWED\_CODES**

**HTTPERROR\_ALLOW\_ALL**

**오프사이트미들웨어**

DEBUG: Filtered offsite request to 'www.othersite.com': <GET http://www.othersite.com/some/page.html>

**리퍼러미들웨어**

**Referer미들웨어 설정**

**REFERER\_ENABLED**

**REFERRER\_POLICY**

**REFERRER\_POLICY에 허용되는 값**

**UrlLength미들웨어**

[**확장**](https://docs.scrapy.org/en/latest/topics/extensions.html)

EXTENSIONS = {

'scrapy.extensions.corestats.CoreStats': 500,

'scrapy.extensions.telnet.TelnetConsole': 500,

}

EXTENSIONS = {

'scrapy.extensions.corestats.CoreStats': **None**,

}

**import** **logging**

**from** **scrapy** **import** signals

**from** **scrapy.exceptions** **import** NotConfigured

logger = logging.getLogger(\_\_name\_\_)

**class** **SpiderOpenCloseLogging**:

**def** \_\_init\_\_(self, item\_count):

self.item\_count = item\_count

self.items\_scraped = 0

**@classmethod**

**def** from\_crawler(cls, crawler):

*# first check if the extension should be enabled and raise*

*# NotConfigured otherwise*

**if** **not** crawler.settings.getbool('MYEXT\_ENABLED'):

**raise** NotConfigured

*# get the number of items from settings*

item\_count = crawler.settings.getint('MYEXT\_ITEMCOUNT', 1000)

*# instantiate the extension object*

ext = cls(item\_count)

*# connect the extension object to signals*

crawler.signals.connect(ext.spider\_opened, signal=signals.spider\_opened)

crawler.signals.connect(ext.spider\_closed, signal=signals.spider\_closed)

crawler.signals.connect(ext.item\_scraped, signal=signals.item\_scraped)

*# return the extension object*

**return** ext

**def** spider\_opened(self, spider):

logger.info("opened spider *%s*", spider.name)

**def** spider\_closed(self, spider):

logger.info("closed spider *%s*", spider.name)

**def** item\_scraped(self, item, spider):

self.items\_scraped += 1

**if** self.items\_scraped % self.item\_count == 0:

logger.info("scraped *%d* items", self.items\_scraped)

kill -QUIT <pid>

**확장 설정**

**확장 로드 및 활성화**

EXTENSIONS = {

'scrapy.extensions.corestats.CoreStats': 500,

'scrapy.extensions.telnet.TelnetConsole': 500,

}

**사용 가능, 활성화 및 비활성화된 확장**

**확장 프로그램 비활성화**

EXTENSIONS = {

'scrapy.extensions.corestats.CoreStats': **None**,

}

**나만의 확장 프로그램 작성**

**import** **logging**

**from** **scrapy** **import** signals

**from** **scrapy.exceptions** **import** NotConfigured

logger = logging.getLogger(\_\_name\_\_)

**class** **SpiderOpenCloseLogging**:

**def** \_\_init\_\_(self, item\_count):

self.item\_count = item\_count

self.items\_scraped = 0

**@classmethod**

**def** from\_crawler(cls, crawler):

*# first check if the extension should be enabled and raise*

*# NotConfigured otherwise*

**if** **not** crawler.settings.getbool('MYEXT\_ENABLED'):

**raise** NotConfigured

*# get the number of items from settings*

item\_count = crawler.settings.getint('MYEXT\_ITEMCOUNT', 1000)

*# instantiate the extension object*

ext = cls(item\_count)

*# connect the extension object to signals*

crawler.signals.connect(ext.spider\_opened, signal=signals.spider\_opened)

crawler.signals.connect(ext.spider\_closed, signal=signals.spider\_closed)

crawler.signals.connect(ext.item\_scraped, signal=signals.item\_scraped)

*# return the extension object*

**return** ext

**def** spider\_opened(self, spider):

logger.info("opened spider *%s*", spider.name)

**def** spider\_closed(self, spider):

logger.info("closed spider *%s*", spider.name)

**def** item\_scraped(self, item, spider):

self.items\_scraped += 1

**if** self.items\_scraped % self.item\_count == 0:

logger.info("scraped *%d* items", self.items\_scraped)

**샘플 확장**

**import** **logging**

**from** **scrapy** **import** signals

**from** **scrapy.exceptions** **import** NotConfigured

logger = logging.getLogger(\_\_name\_\_)

**class** **SpiderOpenCloseLogging**:

**def** \_\_init\_\_(self, item\_count):

self.item\_count = item\_count

self.items\_scraped = 0

**@classmethod**

**def** from\_crawler(cls, crawler):

*# first check if the extension should be enabled and raise*

*# NotConfigured otherwise*

**if** **not** crawler.settings.getbool('MYEXT\_ENABLED'):

**raise** NotConfigured

*# get the number of items from settings*

item\_count = crawler.settings.getint('MYEXT\_ITEMCOUNT', 1000)

*# instantiate the extension object*

ext = cls(item\_count)

*# connect the extension object to signals*

crawler.signals.connect(ext.spider\_opened, signal=signals.spider\_opened)

crawler.signals.connect(ext.spider\_closed, signal=signals.spider\_closed)

crawler.signals.connect(ext.item\_scraped, signal=signals.item\_scraped)

*# return the extension object*

**return** ext

**def** spider\_opened(self, spider):

logger.info("opened spider *%s*", spider.name)

**def** spider\_closed(self, spider):

logger.info("closed spider *%s*", spider.name)

**def** item\_scraped(self, item, spider):

self.items\_scraped += 1

**if** self.items\_scraped % self.item\_count == 0:

logger.info("scraped *%d* items", self.items\_scraped)

**기본 제공 확장 참조**

kill -QUIT <pid>

**범용 확장**

**로그 통계 확장**

**핵심 통계 확장**

**텔넷 콘솔 확장**

**메모리 사용량 확장**

**메모리 디버거 확장**

**스파이더 확장 닫기**

**CLOSESPIDER\_TIMEOUT**

**CLOSESPIDER\_ITEMCOUNT개**

**CLOSESPIDER\_PAGECOUNT**

**CLOSESPIDER\_ERRORCOUNT**

**StatsMailer 확장**

**디버깅 확장**

kill -QUIT <pid>

**스택 추적 덤프 확장**

kill -QUIT <pid>

**디버거 확장**

[**신호**](https://docs.scrapy.org/en/latest/topics/signals.html)

**from** **scrapy** **import** signals

**from** **scrapy** **import** Spider

**class** **DmozSpider**(Spider):

name = "dmoz"

allowed\_domains = ["dmoz.org"]

start\_urls = [

"http://www.dmoz.org/Computers/Programming/Languages/Python/Books/",

"http://www.dmoz.org/Computers/Programming/Languages/Python/Resources/",

]

**@classmethod**

**def** from\_crawler(cls, crawler, \*args, \*\*kwargs):

spider = super(DmozSpider, cls).from\_crawler(crawler, \*args, \*\*kwargs)

crawler.signals.connect(spider.spider\_closed, signal=signals.spider\_closed)

**return** spider

**def** spider\_closed(self, spider):

spider.logger.info('Spider closed: *%s*', spider.name)

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

**pass**

**class** **SignalSpider**(scrapy.Spider):

name = 'signals'

start\_urls = ['https://quotes.toscrape.com/page/1/']

**@classmethod**

**def** from\_crawler(cls, crawler, \*args, \*\*kwargs):

spider = super(SignalSpider, cls).from\_crawler(crawler, \*args, \*\*kwargs)

crawler.signals.connect(spider.item\_scraped, signal=signals.item\_scraped)

**return** spider

**async** **def** item\_scraped(self, item):

*# Send the scraped item to the server*

response = **await** treq.post(

'http://example.com/post',

json.dumps(item).encode('ascii'),

headers={b'Content-Type': [b'application/json']}

)

**return** response

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

**for** quote **in** response.css('div.quote'):

**yield** {

'text': quote.css('span.text::text').get(),

'author': quote.css('small.author::text').get(),

'tags': quote.css('div.tags a.tag::text').getall(),

}

**지연된 신호 처리기**

**class** **SignalSpider**(scrapy.Spider):

name = 'signals'

start\_urls = ['https://quotes.toscrape.com/page/1/']

**@classmethod**

**def** from\_crawler(cls, crawler, \*args, \*\*kwargs):

spider = super(SignalSpider, cls).from\_crawler(crawler, \*args, \*\*kwargs)

crawler.signals.connect(spider.item\_scraped, signal=signals.item\_scraped)

**return** spider

**async** **def** item\_scraped(self, item):

*# Send the scraped item to the server*

response = **await** treq.post(

'http://example.com/post',

json.dumps(item).encode('ascii'),

headers={b'Content-Type': [b'application/json']}

)

**return** response

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

**for** quote **in** response.css('div.quote'):

**yield** {

'text': quote.css('span.text::text').get(),

'author': quote.css('small.author::text').get(),

'tags': quote.css('div.tags a.tag::text').getall(),

}

**내장 신호 참조**

**엔진 신호**

**engine\_started**

**engine\_stopped**

**아이템 신호**

**item\_scraped**

**item\_dropped**

**item\_error**

**스파이더 시그널**

**거미\_닫힌**

**스파이더\_오픈**

**Spider\_idle**

**거미\_오류**

**요청 신호**

**request\_scheduled**

**request\_dropped**

**request\_reached\_downloader**

**request\_left\_downloader**

**bytes\_received**

**headers\_received**

**응답 신호**

**응답\_수신**

**응답\_다운로드**

[**스케줄러**](https://docs.scrapy.org/en/latest/topics/scheduler.html)

**기본 스케줄러 재정의**

**최소한의 스케줄러 인터페이스**

**기본 Scrapy 스케줄러**

[**아이템 수출업자**](https://docs.scrapy.org/en/latest/topics/exporters.html)

**from** **itemadapter** **import** ItemAdapter

**from** **scrapy.exporters** **import** XmlItemExporter

**class** **PerYearXmlExportPipeline**:

*"""Distribute items across multiple XML files according to their 'year' field"""*

**def** open\_spider(self, spider):

self.year\_to\_exporter = {}

**def** close\_spider(self, spider):

**for** exporter, xml\_file **in** self.year\_to\_exporter.values():

exporter.finish\_exporting()

xml\_file.close()

**def** \_exporter\_for\_item(self, item):

adapter = ItemAdapter(item)

year = adapter['year']

**if** year **not** **in** self.year\_to\_exporter:

xml\_file = open(f'*{*year*}*.xml', 'wb')

exporter = XmlItemExporter(xml\_file)

exporter.start\_exporting()

self.year\_to\_exporter[year] = (exporter, xml\_file)

**return** self.year\_to\_exporter[year][0]

**def** process\_item(self, item, spider):

exporter = self.\_exporter\_for\_item(item)

exporter.export\_item(item)

**return** item

**import** **scrapy**

**def** serialize\_price(value):

**return** f'$ *{*str(value)*}*'

**class** **Product**(scrapy.Item):

name = scrapy.Field()

price = scrapy.Field(serializer=serialize\_price)

**from** **scrapy.exporters** **import** XmlItemExporter

**class** **ProductXmlExporter**(XmlItemExporter):

**def** serialize\_field(self, field, name, value):

**if** name == 'price':

**return** f'$ *{*str(value)*}*'

**return** super().serialize\_field(field, name, value)

Item(name='Color TV', price='1200')

Item(name='DVD player', price='200')

['field1', 'field2']

{'field1': 'Field 1', 'field2': 'Field 2'}

**항목 내보내기 사용**

**from** **itemadapter** **import** ItemAdapter

**from** **scrapy.exporters** **import** XmlItemExporter

**class** **PerYearXmlExportPipeline**:

*"""Distribute items across multiple XML files according to their 'year' field"""*

**def** open\_spider(self, spider):

self.year\_to\_exporter = {}

**def** close\_spider(self, spider):

**for** exporter, xml\_file **in** self.year\_to\_exporter.values():

exporter.finish\_exporting()

xml\_file.close()

**def** \_exporter\_for\_item(self, item):

adapter = ItemAdapter(item)

year = adapter['year']

**if** year **not** **in** self.year\_to\_exporter:

xml\_file = open(f'*{*year*}*.xml', 'wb')

exporter = XmlItemExporter(xml\_file)

exporter.start\_exporting()

self.year\_to\_exporter[year] = (exporter, xml\_file)

**return** self.year\_to\_exporter[year][0]

**def** process\_item(self, item, spider):

exporter = self.\_exporter\_for\_item(item)

exporter.export\_item(item)

**return** item

**항목 필드의 직렬화**

**import** **scrapy**

**def** serialize\_price(value):

**return** f'$ *{*str(value)*}*'

**class** **Product**(scrapy.Item):

name = scrapy.Field()

price = scrapy.Field(serializer=serialize\_price)

**from** **scrapy.exporters** **import** XmlItemExporter

**class** **ProductXmlExporter**(XmlItemExporter):

**def** serialize\_field(self, field, name, value):

**if** name == 'price':

**return** f'$ *{*str(value)*}*'

**return** super().serialize\_field(field, name, value)

**1. 필드에서 직렬 변환기 선언**

**import** **scrapy**

**def** serialize\_price(value):

**return** f'$ *{*str(value)*}*'

**class** **Product**(scrapy.Item):

name = scrapy.Field()

price = scrapy.Field(serializer=serialize\_price)

**2. serialize\_field() 메서드 재정의**

**from** **scrapy.exporters** **import** XmlItemExporter

**class** **ProductXmlExporter**(XmlItemExporter):

**def** serialize\_field(self, field, name, value):

**if** name == 'price':

**return** f'$ *{*str(value)*}*'

**return** super().serialize\_field(field, name, value)

**기본 제공 항목 내보내기 참조**

Item(name='Color TV', price='1200')

Item(name='DVD player', price='200')

['field1', 'field2']

{'field1': 'Field 1', 'field2': 'Field 2'}

**기본 항목 내보내기**

['field1', 'field2']

{'field1': 'Field 1', 'field2': 'Field 2'}

**PythonItemExporter**

**XmlItemExporter**

**CSV 항목 내보내기**

**PickleItemExporter**

**PprintItemExporter**

**JsonItemExporter**

**JsonLinesItemExporter**

**MarshalItemExporter**

[**구성품**](https://docs.scrapy.org/en/latest/topics/components.html)

**from** **pkg\_resources** **import** parse\_version

**import** **scrapy**

**class** **MyComponent**:

**def** \_\_init\_\_(self):

**if** parse\_version(scrapy.\_\_version\_\_) < parse\_version('2.7'):

**raise** RuntimeError(

f"*{*MyComponent.\_\_qualname\_\_*}* requires Scrapy 2.7 or "

f"later, which allow defining the process\_spider\_output "

f"method of spider middlewares as an asynchronous "

f"generator."

)

**구성 요소 요구 사항 적용**

**from** **pkg\_resources** **import** parse\_version

**import** **scrapy**

**class** **MyComponent**:

**def** \_\_init\_\_(self):

**if** parse\_version(scrapy.\_\_version\_\_) < parse\_version('2.7'):

**raise** RuntimeError(

f"*{*MyComponent.\_\_qualname\_\_*}* requires Scrapy 2.7 or "

f"later, which allow defining the process\_spider\_output "

f"method of spider middlewares as an asynchronous "

f"generator."

)

[**핵심 API**](https://docs.scrapy.org/en/latest/topics/api.html)

**크롤러 API**

**설정 API**

**스파이더로더 API**

**신호 API**

**통계 수집기 API**

[**릴리즈 노트**](https://docs.scrapy.org/en/latest/news.html)

feedexport/success\_count/<storage type>

feedexport/failed\_count/<storage type>

urllength/request\_ignored\_count

httpcompression/response\_bytes

httpcompression/response\_count

**for** href **in** response.css('li.page a::attr(href)').extract():

url = response.urljoin(href)

**yield** scrapy.Request(url, self.parse, encoding=response.encoding)

**for** a **in** response.css('li.page a'):

**yield** response.follow(a, self.parse)

**class** **MyItem**(scrapy.Item):

url = scrapy.Field()

**class** **MySpider**(scrapy.Spider):

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

**return** MyItem(url=response.url)

**class** **MySpider**(scrapy.Spider):

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

**return** {'url': response.url}

**class** **MySpider**(scrapy.Spider):

custom\_settings = {

"DOWNLOAD\_DELAY": 5.0,

"RETRY\_ENABLED": **False**,

}

**from** **scrapy** **import** log

log.msg('MESSAGE', log.INFO)

**import** **logging**

logging.info('MESSAGE')

**class** **MySpider**(scrapy.Spider):

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

self.logger.info('Response received')

**from** **scrapy.crawler** **import** CrawlerProcess

process = CrawlerProcess({

'USER\_AGENT': 'Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1)'

})

process.crawl(MySpider)

process.start()

69 Daniel Graña <dangra@...>

37 Pablo Hoffman <pablo@...>

13 Mikhail Korobov <kmike84@...>

9 Alex Cepoi <alex.cepoi@...>

9 alexanderlukanin13 <alexander.lukanin.13@...>

8 Rolando Espinoza La fuente <darkrho@...>

8 Lukasz Biedrycki <lukasz.biedrycki@...>

6 Nicolas Ramirez <nramirez.uy@...>

3 Paul Tremberth <paul.tremberth@...>

2 Martin Olveyra <molveyra@...>

2 Stefan <misc@...>

2 Rolando Espinoza <darkrho@...>

2 Loren Davie <loren@...>

2 irgmedeiros <irgmedeiros@...>

1 Stefan Koch <taikano@...>

1 Stefan <cct@...>

1 scraperdragon <dragon@...>

1 Kumara Tharmalingam <ktharmal@...>

1 Francesco Piccinno <stack.box@...>

1 Marcos Campal <duendex@...>

1 Dragon Dave <dragon@...>

1 Capi Etheriel <barraponto@...>

1 cacovsky <amarquesferraz@...>

1 Berend Iwema <berend@...>

130 Pablo Hoffman <pablo@...>

97 Daniel Graña <dangra@...>

20 Nicolás Ramírez <nramirez.uy@...>

13 Mikhail Korobov <kmike84@...>

12 Pedro Faustino <pedrobandim@...>

11 Steven Almeroth <sroth77@...>

5 Rolando Espinoza La fuente <darkrho@...>

4 Michal Danilak <mimino.coder@...>

4 Alex Cepoi <alex.cepoi@...>

4 Alexandr N Zamaraev (aka tonal) <tonal@...>

3 paul <paul.tremberth@...>

3 Martin Olveyra <molveyra@...>

3 Jordi Llonch <llonchj@...>

3 arijitchakraborty <myself.arijit@...>

2 Shane Evans <shane.evans@...>

2 joehillen <joehillen@...>

2 Hart <HartSimha@...>

2 Dan <ellisd23@...>

1 Zuhao Wan <wanzuhao@...>

1 whodatninja <blake@...>

1 vkrest <v.krestiannykov@...>

1 tpeng <pengtaoo@...>

1 Tom Mortimer-Jones <tom@...>

1 Rocio Aramberri <roschegel@...>

1 Pedro <pedro@...>

1 notsobad <wangxiaohugg@...>

1 Natan L <kuyanatan.nlao@...>

1 Mark Grey <mark.grey@...>

1 Luan <luanpab@...>

1 Libor Nenadál <libor.nenadal@...>

1 Juan M Uys <opyate@...>

1 Jonas Brunsgaard <jonas.brunsgaard@...>

1 Ilya Baryshev <baryshev@...>

1 Hasnain Lakhani <m.hasnain.lakhani@...>

1 Emanuel Schorsch <emschorsch@...>

1 Chris Tilden <chris.tilden@...>

1 Capi Etheriel <barraponto@...>

1 cacovsky <amarquesferraz@...>

1 Berend Iwema <berend@...>

**스크래피 2.7.1 (2022-11-02)**

**새로운 기능**

**버그 수정**

**선적 서류 비치**

**품질 보증**

**스크래피 2.7.0 (2022-10-17)**

**수정된 요구 사항**

**지원 중단**

**새로운 기능**

**버그 수정**

**선적 서류 비치**

**품질 보증**

**스크래피 2.6.3 (2022-09-27)**

**스크래피 2.6.2 (2022-07-25)**

**스크래피 2.6.1 (2022-03-01)**

**스크래피 2.6.0 (2022-03-01)**

**보안 버그 수정**

**수정된 요구 사항**

**이전 버전과 호환되지 않는 변경 사항**

**지원 중단 제거**

**지원 중단**

**새로운 기능**

**버그 수정**

**선적 서류 비치**

**품질 보증**

**스크래피 2.5.1 (2021-10-05)**

**스크래피 2.5.0 (2021-04-06)**

feedexport/success\_count/<storage type>

feedexport/failed\_count/<storage type>

urllength/request\_ignored\_count

httpcompression/response\_bytes

httpcompression/response\_count

**지원 중단 제거**

**지원 중단**

**새로운 기능**

feedexport/success\_count/<storage type>

feedexport/failed\_count/<storage type>

urllength/request\_ignored\_count

httpcompression/response\_bytes

httpcompression/response\_count

**버그 수정**

**선적 서류 비치**

**품질 보증**

**스크래피 2.4.1 (2020-11-17)**

**스크래피 2.4.0 (2020-10-11)**

**수정된 요구 사항**

**이전 버전과 호환되지 않는 변경 사항**

**지원 중단 제거**

**지원 중단**

**새로운 기능**

**버그 수정**

**선적 서류 비치**

**품질 보증**

**스크래피 2.3.0 (2020-08-04)**

**지원 중단 제거**

**지원 중단**

**새로운 기능**

**버그 수정**

**선적 서류 비치**

**품질 보증**

**스크래피 2.2.1 (2020-07-17)**

**스크래피 2.2.0 (2020-06-24)**

**이전 버전과 호환되지 않는 변경 사항**

**지원 중단**

**새로운 기능**

**버그 수정**

**선적 서류 비치**

**품질 보증**

**스크래피 2.1.0 (2020-04-24)**

**이전 버전과 호환되지 않는 변경 사항**

**지원 중단 제거**

**지원 중단**

**새로운 기능**

**버그 수정**

**선적 서류 비치**

**품질 보증**

**스크래피 2.0.1 (2020-03-18)**

**스크래피 2.0.0 (2020-03-03)**

**이전 버전과 호환되지 않는 변경 사항**

**지원 중단 제거**

**지원 중단**

**새로운 기능**

**버그 수정**

**선적 서류 비치**

**품질 보증**

**스케줄러 대기열 클래스에 대한 변경 사항**

**스크래피 1.8.3 (2022-07-25)**

**스크래피 1.8.2 (2022-03-01)**

**스크래피 1.8.1 (2021-10-05)**

**스크래피 1.8.0 (2019-10-28)**

**이전 버전과 호환되지 않는 변경 사항**

**새로운 기능**

**버그 수정**

**선적 서류 비치**

**지원 중단 제거**

**지원 중단**

**기타 변경 사항**

**스크래피 1.7.4 (2019-10-21)**

**스크래피 1.7.3 (2019-08-01)**

**스크래피 1.7.2 (2019-07-23)**

**스크래피 1.7.1 (2019-07-18)**

**스크래피 1.7.0 (2019-07-18)**

**이전 버전과 호환되지 않는 변경 사항**

**새로운 기능**

**버그 수정**

**선적 서류 비치**

**지원 중단 제거**

**지원 중단**

**기타 변경 사항**

**스크래피 1.6.0 (2019-01-30)**

**선택기 API 변경 사항**

**텔넷 콘솔**

**새로운 확장 기능**

**새로운 FilePipeline 및 MediaPipeline 기능**

**scrapy.contracts 개선 사항**

**사용성 개선**

**버그 수정**

**문서 개선 사항**

**지원 중단 제거**

**기타 개선 사항, 정리**

**스크래피 1.5.2 (2019-01-22)**

**스크래피 1.5.1 (2018-07-12)**

**스크래피 1.5.0 (2017-12-29)**

**이전 버전과 호환되지 않는 변경 사항**

**새로운 기능**

**버그 수정**

**문서**

**스크래피 1.4.0 (2017-05-18)**

**for** href **in** response.css('li.page a::attr(href)').extract():

url = response.urljoin(href)

**yield** scrapy.Request(url, self.parse, encoding=response.encoding)

**for** a **in** response.css('li.page a'):

**yield** response.follow(a, self.parse)

**지원 중단 및 이전 버전과 호환되지 않는 변경 사항**

**새로운 기능**

**버그 수정**

**정리 및 리팩토링**

**선적 서류 비치**

**스크래피 1.3.3 (2017-03-10)**

**버그 수정**

**스크래피 1.3.2 (2017-02-13)**

**버그 수정**

**스크래피 1.3.1 (2017-02-08)**

**새로운 기능**

**버그 수정**

**선적 서류 비치**

**정리**

**스크래피 1.3.0 (2016-12-21)**

**새로운 기능**

**종속성 및 정리**

**스크래피 1.2.3 (2017-03-03)**

**스크래피 1.2.2 (2016-12-06)**

**버그 수정**

**선적 서류 비치**

**기타 변경 사항**

**스크래피 1.2.1 (2016-10-21)**

**버그 수정**

**선적 서류 비치**

**기타 변경 사항**

**스크래피 1.2.0 (2016-10-03)**

**새로운 기능**

**버그 수정**

**리팩토링**

**테스트 및 요구 사항**

**선적 서류 비치**

**스크래피 1.1.4 (2017-03-03)**

**스크래피 1.1.3 (2016-09-22)**

**버그 수정**

**선적 서류 비치**

**스크래피 1.1.2 (2016-08-18)**

**버그 수정**

**스크래피 1.1.1 (2016-07-13)**

**버그 수정**

**새로운 기능**

**선적 서류 비치**

**테스트**

**스크래피 1.1.0 (2016-05-11)**

**베타 Python 3 지원**

**추가 새로운 기능 및 개선 사항**

**지원 중단 및 제거**

**재배치**

**버그 수정**

**스크래피 1.0.7 (2017-03-03)**

**스크래피 1.0.6 (2016-05-04)**

**스크래피 1.0.5 (2016-02-04)**

**스크래피 1.0.4 (2015-12-30)**

**스크래피 1.0.3 (2015-08-11)**

**스크래피 1.0.2 (2015-08-06)**

**스크래피 1.0.1 (2015-07-01)**

**스크래피 1.0.0 (2015-06-19)**

**class** **MyItem**(scrapy.Item):

url = scrapy.Field()

**class** **MySpider**(scrapy.Spider):

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

**return** MyItem(url=response.url)

**class** **MySpider**(scrapy.Spider):

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

**return** {'url': response.url}

**class** **MySpider**(scrapy.Spider):

custom\_settings = {

"DOWNLOAD\_DELAY": 5.0,

"RETRY\_ENABLED": **False**,

}

**from** **scrapy** **import** log

log.msg('MESSAGE', log.INFO)

**import** **logging**

logging.info('MESSAGE')

**class** **MySpider**(scrapy.Spider):

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

self.logger.info('Response received')

**from** **scrapy.crawler** **import** CrawlerProcess

process = CrawlerProcess({

'USER\_AGENT': 'Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1)'

})

process.crawl(MySpider)

process.start()

**스파이더에서 사전 반환 지원**

**class** **MyItem**(scrapy.Item):

url = scrapy.Field()

**class** **MySpider**(scrapy.Spider):

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

**return** MyItem(url=response.url)

**class** **MySpider**(scrapy.Spider):

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

**return** {'url': response.url}

**스파이더별 설정(GSoC 2014)**

**class** **MySpider**(scrapy.Spider):

custom\_settings = {

"DOWNLOAD\_DELAY": 5.0,

"RETRY\_ENABLED": **False**,

}

**파이썬 로깅**

**from** **scrapy** **import** log

log.msg('MESSAGE', log.INFO)

**import** **logging**

logging.info('MESSAGE')

**class** **MySpider**(scrapy.Spider):

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

self.logger.info('Response received')

**크롤러 API 리팩토링(GSoC 2014)**

**from** **scrapy.crawler** **import** CrawlerProcess

process = CrawlerProcess({

'USER\_AGENT': 'Mozilla/4.0 (compatible; MSIE 7.0; Windows NT 5.1)'

})

process.crawl(MySpider)

process.start()

**모듈 재배치**

**재배치의 전체 목록**

**변경 로그**

**스크래피 0.24.6 (2015-04-20)**

**스크래피 0.24.5 (2015-02-25)**

**스크래피 0.24.4 (2014-08-09)**

**스크래피 0.24.3 (2014-08-09)**

**스크래피 0.24.2 (2014-07-08)**

**스크래피 0.24.1 (2014-06-27)**

**스크래피 0.24.0 (2014-06-26)**

**개선 사항**

**버그 수정**

**스크래피 0.22.2(2014-02-14 출시)**

**스크래피 0.22.1(2014-02-08 출시)**

**Scrapy 0.22.0(2014-01-17 출시)**

**개선 사항**

**수정**

**스크래피 0.20.2(2013-12-09 출시)**

**스크래피 0.20.1(2013-11-28 출시)**

**Scrapy 0.20.0(2013-11-08 출시)**

69 Daniel Graña <dangra@...>

37 Pablo Hoffman <pablo@...>

13 Mikhail Korobov <kmike84@...>

9 Alex Cepoi <alex.cepoi@...>

9 alexanderlukanin13 <alexander.lukanin.13@...>

8 Rolando Espinoza La fuente <darkrho@...>

8 Lukasz Biedrycki <lukasz.biedrycki@...>

6 Nicolas Ramirez <nramirez.uy@...>

3 Paul Tremberth <paul.tremberth@...>

2 Martin Olveyra <molveyra@...>

2 Stefan <misc@...>

2 Rolando Espinoza <darkrho@...>

2 Loren Davie <loren@...>

2 irgmedeiros <irgmedeiros@...>

1 Stefan Koch <taikano@...>

1 Stefan <cct@...>

1 scraperdragon <dragon@...>

1 Kumara Tharmalingam <ktharmal@...>

1 Francesco Piccinno <stack.box@...>

1 Marcos Campal <duendex@...>

1 Dragon Dave <dragon@...>

1 Capi Etheriel <barraponto@...>

1 cacovsky <amarquesferraz@...>

1 Berend Iwema <berend@...>

**개선 사항**

**버그 수정**

**다른**

**감사**

69 Daniel Graña <dangra@...>

37 Pablo Hoffman <pablo@...>

13 Mikhail Korobov <kmike84@...>

9 Alex Cepoi <alex.cepoi@...>

9 alexanderlukanin13 <alexander.lukanin.13@...>

8 Rolando Espinoza La fuente <darkrho@...>

8 Lukasz Biedrycki <lukasz.biedrycki@...>

6 Nicolas Ramirez <nramirez.uy@...>

3 Paul Tremberth <paul.tremberth@...>

2 Martin Olveyra <molveyra@...>

2 Stefan <misc@...>

2 Rolando Espinoza <darkrho@...>

2 Loren Davie <loren@...>

2 irgmedeiros <irgmedeiros@...>

1 Stefan Koch <taikano@...>

1 Stefan <cct@...>

1 scraperdragon <dragon@...>

1 Kumara Tharmalingam <ktharmal@...>

1 Francesco Piccinno <stack.box@...>

1 Marcos Campal <duendex@...>

1 Dragon Dave <dragon@...>

1 Capi Etheriel <barraponto@...>

1 cacovsky <amarquesferraz@...>

1 Berend Iwema <berend@...>

**스크래피 0.18.4(2013년 10월 10일 출시)**

**Scrapy 0.18.3(2013-10-03 출시)**

**스크래피 0.18.2(2013-09-03 출시)**

**스크래피 0.18.1(2013-08-27 출시)**

**스크래피 0.18.0(2013-08-09 출시)**

130 Pablo Hoffman <pablo@...>

97 Daniel Graña <dangra@...>

20 Nicolás Ramírez <nramirez.uy@...>

13 Mikhail Korobov <kmike84@...>

12 Pedro Faustino <pedrobandim@...>

11 Steven Almeroth <sroth77@...>

5 Rolando Espinoza La fuente <darkrho@...>

4 Michal Danilak <mimino.coder@...>

4 Alex Cepoi <alex.cepoi@...>

4 Alexandr N Zamaraev (aka tonal) <tonal@...>

3 paul <paul.tremberth@...>

3 Martin Olveyra <molveyra@...>

3 Jordi Llonch <llonchj@...>

3 arijitchakraborty <myself.arijit@...>

2 Shane Evans <shane.evans@...>

2 joehillen <joehillen@...>

2 Hart <HartSimha@...>

2 Dan <ellisd23@...>

1 Zuhao Wan <wanzuhao@...>

1 whodatninja <blake@...>

1 vkrest <v.krestiannykov@...>

1 tpeng <pengtaoo@...>

1 Tom Mortimer-Jones <tom@...>

1 Rocio Aramberri <roschegel@...>

1 Pedro <pedro@...>

1 notsobad <wangxiaohugg@...>

1 Natan L <kuyanatan.nlao@...>

1 Mark Grey <mark.grey@...>

1 Luan <luanpab@...>

1 Libor Nenadál <libor.nenadal@...>

1 Juan M Uys <opyate@...>

1 Jonas Brunsgaard <jonas.brunsgaard@...>

1 Ilya Baryshev <baryshev@...>

1 Hasnain Lakhani <m.hasnain.lakhani@...>

1 Emanuel Schorsch <emschorsch@...>

1 Chris Tilden <chris.tilden@...>

1 Capi Etheriel <barraponto@...>

1 cacovsky <amarquesferraz@...>

1 Berend Iwema <berend@...>

**스크래피 0.16.5(2013-05-30 출시)**

**Scrapy 0.16.4(2013-01-23 출시)**

**Scrapy 0.16.3(2012-12-07 출시)**

**스크래피 0.16.2(2012-11-09 출시)**

**스크래피 0.16.1(2012-10-26 출시)**

**Scrapy 0.16.0(2012-10-18 출시)**

**스크랩 0.14.4**

**스크랩 0.14.3**

**스크랩 0.14.2**

**스크랩 0.14.1**

**스크랩 0.14**

**새로운 기능 및 설정**

**코드 재배열 및 제거**

**스크랩 0.12**

**새로운 기능 및 개선 사항**

**스크랩 변경**

**설정 변경**

**더 이상 사용되지 않는/사용되지 않는 기능**

**스크랩 0.10**

**새로운 기능 및 개선 사항**

**명령줄 도구 변경 사항**

**API 변경 사항**

**설정 변경**

**스크랩 0.9**

**새로운 기능 및 개선 사항**

**API 변경 사항**

**기본 설정 변경**

**스크랩 0.8**

**새로운 기능**

**이전 버전과 호환되지 않는 변경 사항**

**스크랩 0.7**

[**Scrapy에 기여**](https://docs.scrapy.org/en/latest/contributing.html)

tox -e docs-coverage

tox

tox -e py37

tox -e py37,py38 -p auto

tox -- scrapy tests -x *# stop after first failure*

tox -e py37 -- scrapy tests -n auto

scrapy.loader

tests/test\_loader.py

**버그 보고**

**패치 작성**

tox -e docs-coverage

**패치 제출**

**코딩 스타일**

**문서화 정책**

**테스트**

tox

tox -e py37

tox -e py37,py38 -p auto

tox -- scrapy tests -x *# stop after first failure*

tox -e py37 -- scrapy tests -n auto

scrapy.loader

tests/test\_loader.py

**테스트 실행**

tox

tox -e py37

tox -e py37,py38 -p auto

tox -- scrapy tests -x *# stop after first failure*

tox -e py37 -- scrapy tests -n auto

**테스트 작성**

scrapy.loader

tests/test\_loader.py

[**버전 관리 및 API 안정성**](https://docs.scrapy.org/en/latest/versioning.html)

**버전 관리**

**API 안정성**

**지원 중단 정책**