WenQuanYi Micro Hei [Scale=0.9]WenQuanYi Micro Hei Mono songWenQuanYi Micro Hei sfWenQuanYi Micro Hei "zh" = 0pt plus 1pt

---

scrapy-cookbook Documentation

åŔŚåÿČ 0.2.2

Xiong Neng

11ælJĹ 10, 2017

Contents

Contents:

1 ScrapyæŢŹçÍŃ01- åĚěéŮÍçŕĞ

ScrapyæŸŕäÿĂäÿłäÿžäžĘçĹňåŔŰç¡ŚçńŹæŢřæ■őïijŇæŔŘåŔŰçżŞæđĎæĂğæŢřæ■őèĂŇçijŰåĘŹçŽĎåžŢçŢÍ
ä£ąæĄŕåď˙ĎçŘĘæĹŰå■ŸåĆÍåŐĘåŔšæŢřæ■őç■L'äÿĂçşżåĹŮçŽĎçÍŃåžŔäÿ■ãĂĆåĚűæIJĂåĹİæŸŕäÿžäžĘéąţ
äź§åŔŕäžěåžŢçŢÍåIJÍèŐůåŔŰAPIæL'Ăè£ŤåŻđçŽĎæŢřæ■ő(æŕŤåęĆWeb Services)æĹŰèĂĚéĂŽçŢÍçŽĎç¡ŚçżIJçĹňèŹńãĂĆ

Scrapyäź§èČ¡åÿőä¡ăåőđçŐřéńŸéŸűçŽĎçĹňèŹńæąĘæđűïijŇæŕŤåęĆçĹňåŔŰæŮűçŽĎç¡ŚçńŹèőď'èŕĄãĂĄåĘ

1.1 åőL'èčĚscrapy

æĹŚçŽĎætŃèŕŢçŐŕåćČæŸŕcentos6.5

å■Ğçžğpythonå˙ŁřæIJĂæŰřçL'ĹçŽĎ2.7ïijŇäÿŃéİćçŽĎæL'ĂæIJL'æ■ěéłď'éČ¡åĹĞæ■ćåĹřrootçŤÍæĹů

çŤśäžŐscrapyçŻőåL'■åŔłèČ¡è£ŘèąŇåIJÍpython2äÿŁïijŇæL'Ăäžěå˙ĚĹæŻť'æŰřcentosäÿŁéİćçŽĎpythonå˙ŁřæL
Python 2.7.11ïijŇ åĚůä¡ŞæŰźæşŢèŕůgoogleäÿŃå¿Ĺåď'Žè£Źæăůç˙ŽĎæŢŹçÍŃãĂĆ

åĚĹåőL'èčĚäÿĂäžŻä¿İètŰè¡ŕäžű

```
yum install python-devel
yum install libffi-devel
yum install openssl-devel
```

çĎűåŘŐåőL'èčĚpyopensslážŞ

```
pip install pyopenssl
```

åőL'èčĚxlml

```
yum install python-lxml
yum install libxml2-devel
yum install libxslt-devel
```

åőL'èčĚservice-identity

```
pip install service-identity
```

åőL'èčĚtwisted

```
pip install scrapy
```

åőL'èčĚscrapy

```
pip install scrapy -U
```

æţŃèŕŢscrapy

```
scrapy bench
```

æIJĂçżĹæĹŘåŁ§ïijŇåď'łäÿ■åőźæŸŞäžĘïijĄ

1.2 çőÅå■Ţçď'žä¿Ń

åĹŻåžžäÿĂäÿłpythonæžŘæŰĞäžűïijŇåŘ■äÿžstackoverflow.pyïijŇåĘĚåőźåęĆäÿŃïijŽ

```
import scrapy


class StackOverflowSpider(scrapy.Spider):
    name = 'stackoverflow'
    start_urls = ['http://stackoverflow.com/questions?sort=votes']

    def parse(self, response):
        for href in response.css('.question-summary h3 a::attr(href)
↪'):
            full_url = response.urljoin(href.extract())
            yield scrapy.Request(full_url, callback=self.parse_
↪question)

    def parse_question(self, response):
        yield {
            'title': response.css('h1 a::text').extract()[0],
            'votes': response.css('.question .vote-count-post::text
↪').extract()[0],
            'body': response.css('.question .post-text').
↪extract()[0],
            'tags': response.css('.question .post-tag::text').
↪extract(),
            'link': response.url,
        }
```

è£ŘèąŇïijŽ

```
scrapy runspider stackoverflow_spider.py -o top-stackoverflow-
↪questions.json
```

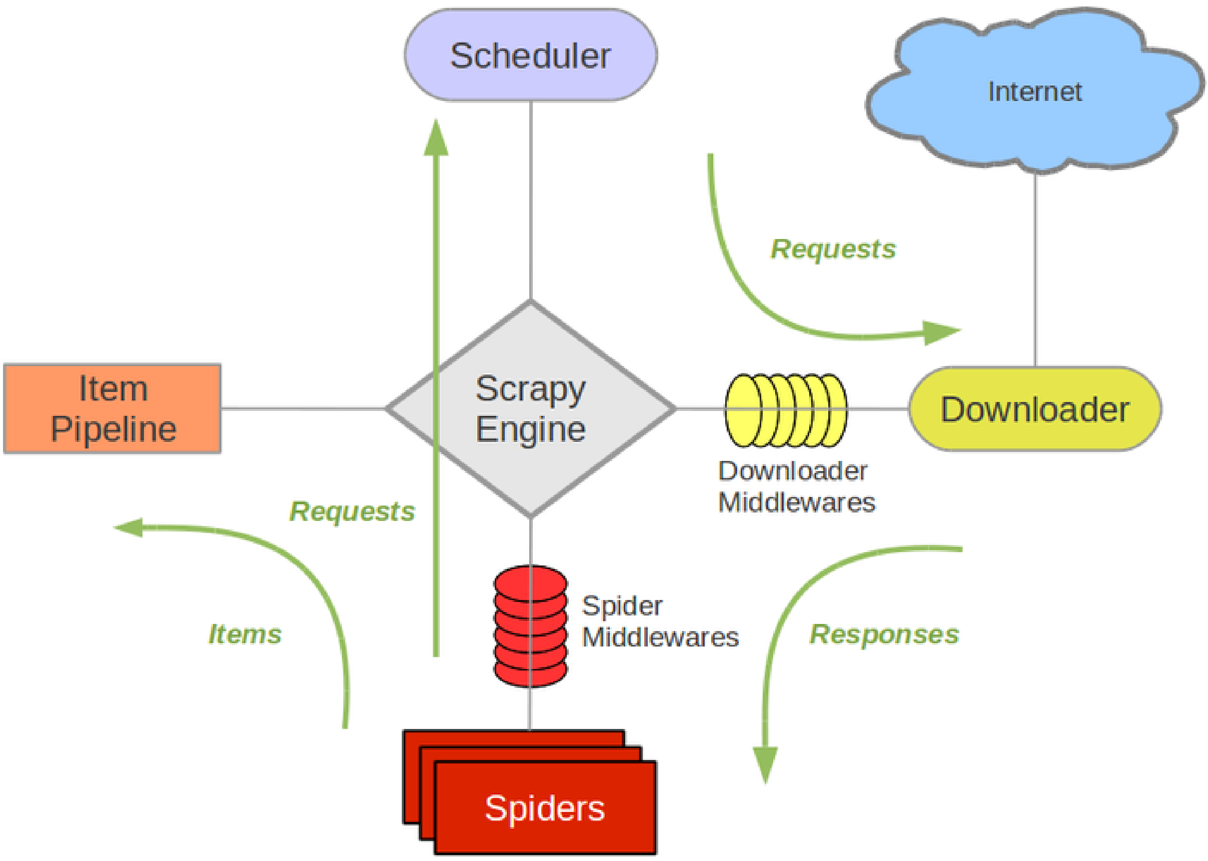
çżŞæđIJçśżäijijäÿŃéİćïijŽ

```
[{
    "body": "... LONG HTML HERE ...",
    "link": "http://stackoverflow.com/questions/11227809/why-is-
↪processing-a-sorted-array-faster-than-an-unsorted-array",
    "tags": ["java", "c++", "performance", "optimization"],
    "title": "Why is processing a sorted array faster than an␣
↪unsorted array?",
    "votes": "9924"
},
{
    "body": "... LONG HTML HERE ...",
    "link": "http://stackoverflow.com/questions/1260748/how-do-i-
↪remove-a-git-submodule",
    "tags": ["git", "git-submodules"],
```

```
    "title": "How do I remove a Git submodule?",
    "votes": "1764"
},
...]
```

å¡Şä¡ăè£ŘèąŇscrapy runspider somefile.pyè£Źæİąèŕ■åŔěçŽĎæŮűåĂŹïijŇScrapyäijŽåŐżåŕżæL
start_urlsåśđæĂğåőŽäźL'äžĘåijĂåğŃçŽĎURLïijŇçĹňèŹńäijŽéĂŽè£ĞåőČæİěæđĎåżžåĹİåğŃçŽĎèŕůæ
æĹŚäżňåIJĺparseåŽđèřČæŰźæşŢäÿ■éĂŽè£Ğä¡£çŤĺcsséĂL'æŃľåŹĺæŔŘåŔŰæŕŔäÿłæŔŘéŮőéąţéİćéŞ¿æŐ
åźűæşĺåĘŇparse_questionåŽđèřČæŰźæşŢïijŇåIJĺè£ŹäÿłèŕůæśĆåőŇæĹŘåŔŐèćńæL'ğèąŇãĂĆ

åď'ĎçŘĘæţĄçĺŃåŻ¿ïijŽ



ScrapyçŽĎäÿĂäÿłåě¡åď'ĎæŸŕæL'ĂæIJL'èŕůæśĆéČ¡æŸŕèćńèřČåžęåźűåijĆæ■ěåď'ĎçŘĘïijŇåřśçőŮæ§Řäÿłèŕů

æĹŚäżňçŽĎçď'žä¿Ńäÿ■åřĘèğčæđŘçżŞæđIJçŤ§æĹŘjsonæăijåijŔïijŇä¡ăè£ŸåŔŕäżěåŕijåĞžäÿžåĚűäżŰæăijåijĘ
S3äÿŁãĂĆ ä¡ăè£ŸåŔŕäżěéĂŽè£Ğpipeline åřĘåőČäżňå■ŸåĆĺåĹřæŢřæ■őåžŞäÿ■åŐżïijŇè£ŹäžŻæŢřæ■őä£İå■

1.3 ScrapyçĽźæĂğäÿĂèğĹ

ä¡ăåůšçżŔåŔŕäżěéĂŽè£ĞScrapyäżŐäÿĂäÿłç¡ŚçńŹäÿŁéİćçĹňåŔŰæŢřæ■őåźűåřĘåĚűèğčæđŘä£İå■ŸäÿŃæİě
ScrapyæŔŘä¿ŻäžĘæŻt'åd'ŽçŽĎçĽźæĂğæİěèőľä¡ăçĹňåŔŰæŻt'åŁăåőźæŸŞåŠŇéńŸæŢĹãĂĆæŕŤåęĆïijŽ

1. åĘĚç¡őæŤŕæŇĄæĽ'ľåśŢçŽĎCSSéĂL'æŃľåŹĺåŠŇXPathèąĺè¿¿åijŔæİěäżŐHTML/XMLæžŘçăĄäÿ■éĂ
2. æŔŘä¿Żäžď'äžŠåijŔshellæŐğåĹűåŔřèŕŢéłŇCSSåŠŇXPathèąĺè¿¿åijŔïijŇè£ŹäÿłåIJĺèřČèŕŢä¡ăçŽĎèIJŸ
3. åĘĚç¡őæŤŕæŇĄçŤ§æĹŘåď'Žçğ■æăijåijŔçŽĎèőćéŸĚåŕijåĞžïijĹJSONãĂĄCSVãĂĄXMLïijL'åźűåřĘåő

4. åĄěåčőçŽĎçijŰçăĄæŤŕæŇĄåŠŇèĞłåŁĺèŕĘåĹńïijŇçŤĺäžŐåď˙ĎçŘĘåď˙ŰæŰĞãĂĄéİđæăĞåĞĘåŠŇéŤŹ
5. åŔŕæĽľ˙åśŢïijŇåĚĄèőÿä¡ăä¡£çŤĺsignals åŠŇåŔŇåě¡çŽĎAPI(middlewares, extensions, åŠŇpipelines)æİěçijŰåĘŹèĞłåőŽäźĽ˙æŔŠäžűåŁ§èČ¡ãĂĆ
6. åď˙ğéĞŔçŽĎåĘĚç¡őæĽľ˙åśŢåŠŇäÿ■éŮť˙äžűä¿Żä¡£çŤĺïijŽ
    - cookies and session handling
    - HTTP features like compression, authentication, caching
    - user-agent spoofing
    - robots.txt
    - crawl depth restriction
    - and more
7. è£ŸæIJL˙åĚűäžŰåě¡åď˙Žåě¡äÿIJäÿIJïijŇærŤåęĆåŔŕéĞ■åď˙■åĹľçŤĺèIJŸèŻŽæİěçĹňåŔŰSitemapsåŠŇX äÿĂäÿłèů§çĹňåŔŰåĚČçť˙ăåĚşèĄŢçŽĎåłŠä¡Şçőąé ĄŞæİě èĞłåŁĺäÿŃè¡¡åŻ¿çL’ĞïijŇ äÿĂäÿłçijŞå■ŸDNSèğčæđŘåŹĺç■L’ç■L’ãĂĆ

2 ScrapyæŢŹçíŃ02- åőŇæŢť˙çď˙žä¿Ń

è£ŹçŕĞæŰĞçńăæĹŚäžňéĂŽè£ĞäÿĂäÿłærŤè¿ČåőŇæŢť˙çŽĎä¿Ńå■ŘæİěæŢŹä¡ää¡£çŤĺScrapyïijŇæĹŚéĂL’æ

è£ŹéĞŇæĹŚäžňåřĘåőŇæĹŘåęĆäÿŃåĞăäÿłæ■ěéłď˙ïijŽ

- åĹŻåžžäÿĂäÿłæŰřçŽĎScrapyåůěçíŃ
- åőŽäźĽä¡ăæL’ĂéIJĂèęĄèęĄæŁ¡åŔŰçŽĎItemářźèšą
- çijŰåĘŹäÿĂäÿłspideræİěçĹňåŔŰæ§Řäÿłç¡ŚçńŹåźűæŔŘåŔŰåĞžæL’ĂæIJL’çŽĎItemářźèšą
- çijŰåĘŹäÿĂäÿłItem Piplineæİěå■ŸåĆĺæŔŘåŔŰåĞžæİěçŽĎItemářźèšą

Scrapyä¡£çŤĺPythonèŕ■èĺĂçijŰåĘŹïijŇåęĆæđIJä¡ăářźè£ŹéŮĺèŕ■èĺĂè£Ÿäÿ■çĘ§ïijŇèřůåĚĹåŐżå■ęäźăäÿŃå§ž

2.1 åĹŻåžžScrapyåůěçíŃ

åIJĺäžżä¡Ţä¡ăåŰIJæňćçŽĎçŻőå¡ŢæL’ğèąŇåęĆäÿŃåŚ¡äžď

```
scrapy startproject coolscrapy
```

åřĘäijŽåĹŻåžžcoolscrapyæŰĞäžűåď˙źïijŇåĚűçŻőå¡ŢçżŞæđĎåęĆäÿŃïijŽ

```
coolscrapy/
    scrapy.cfg            # éČĺç¡šéĚ■ç¡őæŰĞäžű

    coolscrapy/           #␣
↪Pythonæĺąå İŮïijŇä¡ăæL’ĂæIJL’çŽĎäżčçăĄéČ¡æŤ¿è£ŹéĞŇéİć
        __init__.py
```

```
        items.py            # ItemåőŽäźL'æŰĞäżű

        pipelines.py        # pipelinesåőŽäźL'æŰĞäżű

        settings.py         # éĚ■ç¡őæŰĞäżű

        spiders/            #␣
↪æL'ĂæIJL'çĹňèŹńspidereČ¡æŤ¿è£ŹäÿłæŰĞäżűåd'źäÿŃéİć
            __init__.py
            ...
```

2.2 åőŽäźL'æĹŚäžňçŽĎItem

æĹŚäžňéĂŽè£ĞåĹŻåžžäÿĂäÿłscrapy.ItemçśżïijŇåźűåőŽäźL'åőČçŽĎçśżåđŇäÿžscrapy.FieldçŽĎåśđæĂğïijŇ
æĹŚäžňåĞĘåd'ĞåřĘèŹŐåŮĚç¡ŚæŰřéŮżåĹŮèąĺçŽĎåŘ■çğřãĂĄéŞ¿æŐěåIJřåİĂåŠŇæŚŸèęĄçĹňåŔŰäÿŃæİè

```
import scrapy

class HuxiuItem(scrapy.Item):
    title = scrapy.Field()     # æăĞécŸ
    link = scrapy.Field()      # éŞ¿æŐě
    desc = scrapy.Field()      # çőĂè£ř
    posttime = scrapy.Field() # åŔŚåÿČæŮűéŮt'
```

äź§èőÿä¡ăèğL'å¿ŮåőŽäźL'è£ŹäÿłItemæIJL'çĆźéžżçČęïijŇä¡ĘæŸřåőŽäźL'åőŇäźŃåŘŐä¡ääŔřäžěå¿ŮåĹřèőÿåc

2.3 çňňäÿĂäÿłSpider

èIJŸèŻŽåřśæŸřä¡ääőŽäźL'çŽĎäÿĂäžŽçśżïijŇScrapyä¡£çŤĺåőČäžňæİěäżŐäÿĂäÿłdomainïijĹæĹŰdomainçżĎ
åIJĺèIJŸèŻŽçśżäÿ■åőŽäźL'äžĘäÿĂäÿłåĹİåğŃåŇŰçŽĎURLäÿŃè¡¡åĹŮèąĺïijŇäžěåŔŁæĂŐæăůèů§èÿłéŞ¿æŐě

åőŽäźL'äÿĂäÿłSpiderïijŇåŔłéIJÂçžğæL'£`scrapy.Spider`çśżåźűåőŽäžŐäÿĂäžŽåśđæĂğïijŽ

- name: SpideråŘ■çğřïijŇå£ĚéążæŸřåŤřäÿĂçŽĎ
- start_urls: åĹİåğŃåŇŰäÿŃè¡¡éŞ¿æŐěURL
- parse(): çŤĺæİèèğčæđŘäÿŃè¡¡åŘŐçŽĎResponseåŕźèśąïijŇèŕěåŕźèśąäź§æŸřè£ŹäÿłæŰźæşŢçŽĎåŤřäÿĂ
  åőČèt'§èt'čèğčæđŘè£ŤåŽđéąţéİćæŢřæ■őåźűæŔŘåŔŰåĞžçŻÿåžŢçŽĎItemïijĹè£ŤåŽđItemåŕźèśąïijL'ïij

æĹŚäžňåIJĺcoolscrapy/spidersæŰĞäžűåd'źäÿŃéİćæŰřåžžhuxiu_spider.
pyïijŇåĘĚåőźåęĆäÿŃïijŽ

```
#!/usr/bin/env python
# -*- encoding: utf-8 -*-
"""
Topic: sample
Desc :
```

```
"""
from coolscrapy.items import HuxiuItem
import scrapy

class HuxiuSpider(scrapy.Spider):
    name = "huxiu"
    allowed_domains = ["huxiu.com"]
    start_urls = [
        "http://www.huxiu.com/index.php"
    ]

    def parse(self, response):
        for sel in response.xpath('//div[@class="mod-info-flow"]/
↪div/div[@class="mob-ctt"]'):
            item = HuxiuItem()
            item['title'] = sel.xpath('h3/a/text()')[0].extract()
            item['link'] = sel.xpath('h3/a/@href')[0].extract()
            url = response.urljoin(item['link'])
            item['desc'] = sel.xpath('div[@class="mob-sub"]/text()
↪')[0].extract()
            print(item['title'],item['link'],item['desc'])
```

2.4 è£ŘèąŇçĹňèŹń

åIJÍæăźçŻőå¡ŢæL'ğèąŇäÿŃéİćçŽĎåŚ¡äžd'ïijŇåĚűäÿ■huxiuæŸŕä¡ăåőŽäźL'çŽĎspideråŘ■å■ŮïijŽ

```
scrapy crawl huxiu
```

åęĆæđIJäÿĂåĹĞæ■čåÿÿïijŇåžŤèŕěåŔŕäžěæL'Şå■řåĞžæŕŔäÿĂäÿłæŰřéŮż

2.5 åď̌ĎçŘĘéŞ¿æŐě

åęĆæđIJæČşçżğçż■èů§èÿłæŕŔäÿłæŰřéŮżéŞ¿æŐěè£ŻåŐżïijŇçIJŃçIJŃåőČçŽĎèŕęçżĘåĘĚåőźçŽĎèŕÍïijŇéĆčçĎűåŘŐæşĺåĘŇäÿĂäÿłåŻđèřČåĞ¡æŢřæİěèğčæđŘæŰřéŮżèŕęæČĚãĂĆ

```
from coolscrapy.items import HuxiuItem
import scrapy

class HuxiuSpider(scrapy.Spider):
    name = "huxiu"
    allowed_domains = ["huxiu.com"]
    start_urls = [
        "http://www.huxiu.com/index.php"
    ]

    def parse(self, response):
        for sel in response.xpath('//div[@class="mod-info-flow"]/
↪div/div[@class="mob-ctt"]'):
```

```
            item = HuxiuItem()
            item['title'] = sel.xpath('h3/a/text()')[0].extract()
            item['link'] = sel.xpath('h3/a/@href')[0].extract()
            url = response.urljoin(item['link'])
            item['desc'] = sel.xpath('div[@class="mob-sub"]/text()
↪')[0].extract()
            # print(item['title'],item['link'],item['desc'])
            yield scrapy.Request(url, callback=self.parse_article)

    def parse_article(self, response):
        detail = response.xpath('//div[@class="article-wrap"]')
        item = HuxiuItem()
        item['title'] = detail.xpath('h1/text()')[0].extract()
        item['link'] = response.url
        item['posttime'] = detail.xpath(
            'div[@class="article-author"]/span[@class="article-time
↪"]/text()')[0].extract()
        print(item['title'],item['link'],item['posttime'])
        yield item
```

çŐřåIJÍparseåŔłæŔŘåŔŰæĎ§åĚť èűčçŽĎéŞ¿æŐěïijŇçĎűåŘŐářĘéŞ¿æŐěåĘĚåőźèğčæđŘäžd'çżŹåŔęåd'ŰçŽ
ä¡ăåŔŕäžěå§žäžŐè£ŹäýłæđĎåžžæŻť'åŁăåd'■æİĆçŽĎçĹňèŹńçÍŇåžŔäžĘãĂĆ

2.6 åŕijåĞžæŁŞåŔŰæŢřæ■ő

æIJĂçőĂå■ŢçŽĎä£Ìå■ŸæŁŞåŔŰæŢřæ■őçŽĎæŰźåijŔæŸŕä¡£çŤĺjsonæăijåijŔçŽĎæŰĞäžűä£Ìå■ŸåIJÍæIJňåI

```
scrapy crawl huxiu -o items.json
```

åIJÍæijŤçd'žçŽĎåřŔçşżçż§éĞŇéİćè£ŹçğæŰźåijŔèűşåd'§äžĘãĂĆäÿ■è£ĞåęĆæđIJä¡äèęĄæđĎåžžåd'■æİĆçŽ
æIJĂåě¡èĞłåůšçijŰåĘŹItem PipelineãĂĆ

2.7 ä£Ìå■ŸæŢřæ■őåĹřæŢřæ■őåžŞ

äÿŁéİćæĹŚäžňäžŃçż■äžĘåŔŕäžěåřĘæŁŞåŔŰçŽĎItemåŕijåĞžäÿžjsonæăijåijŔçŽĎæŰĞäžűïijŇäÿ■è£ĞæIJĂå
pipelines.pyåőŽäźĽ

```
# -*- coding: utf-8 -*-
import datetime
import redis
import json
import logging
from contextlib import contextmanager

from scrapy import signals
from scrapy.exporters import JsonItemExporter
from scrapy.pipelines.images import ImagesPipeline
```

```
from scrapy.exceptions import DropItem
from sqlalchemy.orm import sessionmaker
from coolscrapy.models import News, db_connect, create_news_table,␣
↪Article


class ArticleDataBasePipeline(object):
    """ä£Ìå■ŸæŰĞçńääĹřæŢřæ■őåžŞ"""

    def __init__(self):
        engine = db_connect()
        create_news_table(engine)
        self.Session = sessionmaker(bind=engine)

    def open_spider(self, spider):
        """This method is called when the spider is opened."""
        pass

    def process_item(self, item, spider):
        a = Article(url=item["url"],
                    title=item["title"].encode("utf-8"),
                    publish_time=item["publish_time"].encode("utf-8
↪"),
                    body=item["body"].encode("utf-8"),
                    source_site=item["source_site"].encode("utf-8"))
        with session_scope(self.Session) as session:
            session.add(a)

    def close_spider(self, spider):
        pass
```

äÿŁéÍćæĹŚä¡£çŤÍäžĘpythonäÿ■çŽĎSQLAlchemyæÌëä£Ìå■ŸæŢřæ■őåžŞïijŇè£ŹäÿłæŸŕäÿĂäÿłéÌđåÿÿäijŸçğ

çĎűåŘŐåIJÍsetting.pyäÿ■éĚ■ç¡õè£ŹäÿłPipelineïijŇè£ŸæIJL'æŢřæ■őåžŞéŞ¿æŐěç■L'ä£qæĄŕïijŽ

```
ITEM_PIPELINES = {
    'coolscrapy.pipelines.ArticleDataBasePipeline': 5,
}

# linux pip install MySQL-python
DATABASE = {'drivername': 'mysql',
            'host': '192.168.203.95',
            'port': '3306',
            'username': 'root',
            'password': 'mysql',
            'database': 'spider',
            'query': {'charset': 'utf8'}}
```

åĘ■æňąè£ŘèąŇçĹňèŹń

```
scrapy crawl huxiu
```

éĆčäźĹæĽĂæIJĽæŰřéŮżçŽĎæŰĞçńăéČ¡å■ŸåĆĺåĹřæŢřæ■őåžŞäÿ■åŐżäžĘãĂĆ

2.8 äÿŃäÿĂæ■ě

æIJňçńääŘłæŸŕåÿęä¡äéćEçŢěäžĘscrapyæIJĂå§žæIJňçŽĎåŁ§èČ¡ïijŇè£ŸæIJL'å¿Ĺåd'ŽéŃŸçžğçL'źæĂğæšąæl

3 ScrapyæŢŹçÍŃ03- Spiderèŕęèğč

SpideræŸŕçĹňèŹńæąĘæđűçŽĎæăÿå£ČïijŇçĹňåŔŰæţĄçÍŃåęĆäÿŃïijŽ

1. åĚĹåĹİåğŃåŇŰèřůæśCURLåĹŮèąĺïijŇåźűæŇĞåőŽäÿŃè¡¡åŘŐåd'ĎçŘĘresponseçŽĎåŻđèřČåĞ¡æŢřã
2. åIJÍparseåŻđèřČäÿ■èğčæđŘresponseåźűè£ŤåŻđå■ŮåĚÿ,`Item`åŕźèśą,`Request`åŕźèśąæĹŰåőČäžňçŽĎ
3. åIJÍåŻđèřČåĞ¡æŢréĞŇéİćïijŇä¡äéĂŽè£Ğä¡£çŤĺéĂL'æŃľåŹĺïijĹåŘŇæăůåŔŕäžěä¡£çŤÍBeautifulSoup,l
4. æIJĂåŘŐè£ŤåŻđçŽĎè£ŹäžŻIteméĂŽåÿÿäijŽèćńæŇĄäźĚåŇŰåĹřæŢřæ■őåžŞäÿ■(ä¡£çŤÍItem Pipeline)æĹŰèĂĚä¡£çŤÍFeed exportsåřĘåĚűä£Ìå■ŸåĹřæŰĞäžűäÿ■ãĂĆ

år¡çőąè£ŹäÿłæţĄçÍŃéĂĆåŘĹäžŐæL'ĂæIJL'çŽĎèIJŸèŻŻïijŇä¡ĘæŸŕScrapyéĞŇéİćäÿžäÿ■åŘŇçŽĎä¡£çŤÍçŻ

3.1 CrawlSpider

éŞ¿æŐěçĹňåŔŰèIJŸèŻŻïijŇäÿŞéŮÍäÿžéĆčäžŻçĹňåŔŰæIJL'çL'źåőŽègĎå¿ŃçŽĎéŞ¿æŐěåĘĚåőźèĂŇåĞĘåd
åęĆæđIJä¡äèğL'å¿ŮåőČè£Ÿäÿ■èűşäžěéĂĆåŘĹä¡äçŽĎéIJĂæśČïijŇåŕźäžěåĚĹçżğæL'£åőČçĎűåŘŐèęĘçŽŰçŻ

åőČéŹđ'äžĘäžŐ`scrapy.Spider`çśżçżğæL'£çŽĎåśđæĂğåđ'ŰïijŇè£ŸæIJL'äÿĂäÿłæŰřçŽĎåśđæĂğ`rules`

äÿĂäÿłèŕęçżĘçŽĎä¿Ńå■ŘïijŽ

```
from coolscrapy.items import HuxiuItem
import scrapy
from scrapy.spiders import CrawlSpider, Rule
from scrapy.linkextractors import LinkExtractor


class LinkSpider(CrawlSpider):
    name = "link"
    allowed_domains = ["huxiu.com"]
    start_urls = [
        "http://www.huxiu.com/index.php"
    ]

    rules = (
        # æŔŘåŘŰåŇźéĚ■æ■čåĹŹåijŕ'/group?f=index_group'éŞ¿æŐě␣
↪(ä¡ĘæŸŕäÿ■èČ¡åŇźéĚ■'deny.php')
```

```
        #␣
↪åźűäÿŤäijžéĂŠå¡ŠçĹňåŔŰ(åęĆædIJæšąæIJL'åőžäźL'callbackïijŇéżŸèőd'follow=True).
↪
        Rule(LinkExtractor(allow=('/group?f=index_group', ), deny=(
↪'deny\.php', ))),
        # æŔŘåŔŰåŇźéĚ■'/article/\d+/\d+.html
↪'çŽĎéŞ¿æŐěïijŇåźűä¡£çŤĺparse_
↪itemæİěèğčædŘåőČäżňäÿŃè¡¡åŘŐçŽĎåĘĚåőźïijŇäÿ■éĂŠå¡Š
        Rule(LinkExtractor(allow=('/article/\d+/\d+\.html', )),␣
↪callback='parse_item'),
    )

    def parse_item(self, response):
        self.logger.info('Hi, this is an item page! %s', response.
↪url)
        detail = response.xpath('//div[@class="article-wrap"]')
        item = HuxiuItem()
        item['title'] = detail.xpath('h1/text()')[0].extract()
        item['link'] = response.url
        item['posttime'] = detail.xpath(
            'div[@class="article-author"]/span[@class="article-time
↪"]/text()')[0].extract()
        print(item['title'],item['link'],item['posttime'])
        yield item
```

3.2 XMLFeedSpider

XMLèőćéŸĚèIJŸèŻŽïijÑçŤĺæİěçĹňåŔŰXMLå¡ćåijŔçŽĎèőćéŸĚåĘĚåőźïijÑéĂŽè£ĞæŞŘäÿłæŇĞåőŽçŽĎè
åŔŕä¡£çŤĺiternodes, xml, åŠŇhtmläÿL'çğ■å¡ćåijŔçŽĎè£■äžčåŹĺïijŇäÿ■è£Ğå¡ŞåĘĚåőźærŤè¿Čåď'ŽçŽĎ
éżŸèőď'äź§æŸŕåőČïijŇåŔŕäžèèŁĆçIJAåĘĚå■ŸæŔŘå■ĞæĂğèČ¡ïijŇäÿ■éIJĂèęĄåřĘæŢť'äÿłDOMåŁăè¡¡åĹřå
åď'ĎçŘĘXMLçŽĎæŮűåĂŹæIJĂåě¡åĚĹRemoving namespaces

æŐěäÿŃæİěæĹŚéĂŽè£ĞçĹňåŔŰæĹŚçŽĎå■ŽåőćèőćéŸ ĚXMLæİěåśŢçď'žåőČçŽĎä¡£çŤĺæŰźæşŢãĂĆ

```
from coolscrapy.items import BlogItem
import scrapy
from scrapy.spiders import XMLFeedSpider


class XMLSpider(XMLFeedSpider):
    name = "xml"
    namespaces = [('atom', 'http://www.w3.org/2005/Atom')]
    allowed_domains = ["github.io"]
    start_urls = [
        "http://www.pycoding.com/atom.xml"
    ]
    iterator = 'xml'  #␣
↪çijžçIJAçŽĎiternodesïijŇèšŇäijijåŕźäžŐæIJL'namespaceçŽĎxmläÿ■èąŇ
```

```
    itertag = 'atom:entry'

    def parse_node(self, response, node):
        # self.logger.info('Hi, this is a <%s> node!', self.itertag)
        item = BlogItem()
        item['title'] = node.xpath('atom:title/text()')[0].extract()
        item['link'] = node.xpath('atom:link/@href')[0].extract()
        item['id'] = node.xpath('atom:id/text()')[0].extract()
        item['published'] = node.xpath('atom:published/text()')[0].
↪extract()
        item['updated'] = node.xpath('atom:updated/text()')[0].
↪extract()
        self.logger.info('|'.join([item['title'],item['link'],item[
↪'id'],item['published']]))
        return item
```

3.3 CSVFeedSpider

è£Źäÿłèů§äÿŁéİćçŽĎXMLFeedSpideråż ŁçśżäijijïijŇåŇžåĹńåIJĺäžŐåőČäijŽäÿĂèąŇäÿĂèąŇçŽĎè£■äžčïijŇè
æfŔæňąè£■äžčèąŇçŽĎæŮűåĂŹäijŽèřČçŤĺparse_row()æŰźæşŢãĂĆ

```
from coolscrapy.items import BlogItem
from scrapy.spiders import CSVFeedSpider


class CSVSpider(CSVFeedSpider):
    name = "csv"
    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 = BlogItem()
        item['id'] = row['id']
        item['name'] = row['name']
        return item
```

3.4 SitemapSpider

çńŹçĆźåIJřåŻ¿èIJŸèŻŽïijŇåĚĄèőÿä¡ää¡£çŤĺSitemapsåŔŚçŐřURLåŘŐçĹňåŔŰæŢťäÿłçńŹçĆźãĂĆ
è£ŸæŤŕæŇĄåţŇåěŮçŽĎçńŹçĆźåIJřåŻ¿äžěåŔŁäžŐrobots.txtäÿ■åŔŚçŐřçńŹçĆźURL

4 ScrapyæŢŹçÍŃ04- Selectorèŕęèğč

åIJÍä¡ăçĹňåŔŰç¡Śéąţç ŽĎæŮűåĂŹïijŇæIJĂæŹőéĄ■çŽĎäžŇæČĚářśæŸŕåIJÍéąţéİćæžŘçăĄäÿ■æŔŘåŔŰéIJĂè

1. BeautifulSoupæŸŕpythonäÿ■äÿĂäÿłéİđåÿÿæţĄèąŇçŽĎæŁŞåŔŰåžŞ, åőČè£ŸèČ¡åŘĹçŘĘçŽĎåď ĎçŘĘéŤŹèŕŕæăijåijŔçŽĎæăĞç■¿ïijŇä¡ĘæŸŕæIJL'äÿĂäÿłåŤŕäÿĂçijžçĆźåř
2. lxmlæŸŕäÿĂäÿłå§žäžŐElementTreeçŽĎXMLèğčædŘåžŞ(åŘŇæŮűè£ŸèČ¡èğčædŘHTML), äÿ■è£Ğlxmlåźűäÿ■æŸŕPythonæăĞåĞĘåžŞ

èĂŇScrapyåődçŐřäžĘèĞłåůśçŽĎæŢřæ■őæŔŘåŔŰæIJžåĹűïijŇåőČäžňèćńçğřäÿžéĂL'æŃľåŹĺïijŇéĂŽè£ĞXI

XPathæŸŕäÿĂçŤĺæİéåIJÍXMLäÿ■éĂL'æŃľèŁĆçĆźçŽĎèŕ■èĺĂïijŇåŘŇæŮűåŔŕäžěçŤĺåIJÍHTMLäÿŁéİćãĂĆ CSSæŸŕäÿĂçğ■HTMLæŰĞæąčäÿŁéİćçŽĎæăůåijŔèŕ■èĺĂãĂĆ

ScrapyéĂL'æŃľåŹĺædĎåžžåIJÍlxmlå§žçąĂäźŇäÿŁïijŇæL'Ăäžěåŕäžěä£ÌèŕĄéĂ§åžęåŠŇåĞEçąőæĂğãĂĆ

æIJňçńăæĹŚäžňæİèèŕęçżĘèőšèğčäÿŇéĂL'æŃľåŹĺçŽĎåůěä¡IJåŐ§çŘĘïijŇè£ŸæIJL'åőČäžňædĄåĚűçőÅå■Ţå

åőŇæŢt'çŽĎAPIèŕůæ§ěçIJŃSelectoråŔĆèĂČ

4.1 åĚşäžŐéĂL'æŃľåŹĺ

ScrapyåÿőæĹŚäžňäÿŇè¡¡åőŇéąţéİćåŘŐïijŇæĹŚäžňæĂŐæăůåIJÍæżąæŸŕhtmlæăĞç■¿çŽĎåĘĚåőźäÿ■æL'¿åĹř

- /html/head/title: éĂL'æŃľ<title>èŁĆçĆź, åőČä¡■äžŐhtmlæŰĞæąčçŽĎ<head>èŁĆçĆźåĘĚ
- /html/head/title/text(): éĂL'æŃľäÿŁéİćçŽĎ<title>èŁĆçĆźçŽĎåĘĚåőź.
- //td: éĂL'æŃľéąţéİćäÿ■æL'ĂæIJL'çŽĎåĚČçt'ă
- //div[@class="mine"]: éĂL'æŃľæL'ĂæIJL'æŇěæIJL'åśđæĂğclass="mine"çŽĎdivåĚČçt'ă

Scrapyä¡£çŤĺcssåŠŇxpathéĂL'æŃľåŹĺæİěåőŽä¡■åĚČçt'äïijŇåőČæIJL'åŻŽäÿłå§žæIJňæŰźæşŢïijŽ

- xpath(): è£ŤåŻđéĂL'æŃľåŹĺåĹŮèąĺïijŇæŕŔäÿłéĂL'æŃľåŹĺäžčèąĺä¡£çŤĺxpathèŕ■æşŢéĂL'æŃľçŽĎèŁ
- css(): è£ŤåŻđéĂL'æŃľåŹĺåĹŮèąĺïijŇæŕŔäÿłéĂL'æŃľåŹĺäžčèąĺä¡£çŤĺcssèŕ■æşŢéĂL'æŃľçŽĎèŁĆçĆź
- extract(): è£ŤåŻđèćńéĂL'æŃľåĚČçt'ăçŽĎunicodeå■Ůçňęäÿš
- re(): è£ŤåŻđéĂŽè£Ğæ■čåĹŹèąĺè¿¿åijŔæŔŘåŔŰçŽĎunicodeå■ŮçňęäÿšåĹŮèąĺ

4.2 ä¡£çŤĺéĂL'æŃľåŹĺ

äÿŇéİćæĹŚäžňéĂŽè£ĞScrapy shellæijŤçd'žäÿŇéĂL'æŃľåŹĺçŽĎä¡£çŤĺïijŇåĄĞèő¿æĹŚäžňæIJL'åęĆäÿŇçŽĺ //doc.scrapy.org/en/latest/_static/selectors-sample1.htmlïijŇåĘĚåőźåęĆäÿŇïijŽ

```
<html>
 <head>
  <base href='http://example.com/' />
  <title>Example website</title>
 </head>
```

```
 <body>
  <div id='images'>
   <a href='image1.html'>Name: My image 1 <br /><img src='image1_
↪thumb.jpg' /></a>
   <a href='image2.html'>Name: My image 2 <br /><img src='image2_
↪thumb.jpg' /></a>
   <a href='image3.html'>Name: My image 3 <br /><img src='image3_
↪thumb.jpg' /></a>
   <a href='image4.html'>Name: My image 4 <br /><img src='image4_
↪thumb.jpg' /></a>
   <a href='image5.html'>Name: My image 5 <br /><img src='image5_
↪thumb.jpg' /></a>
  </div>
 </body>
</html>
```

éęŰåĚĹæĹŚäžňæL'Şåij Ăshell

```
scrapy shell http://doc.scrapy.org/en/latest/_static/selectors-
↪sample1.html
```

è£ŘèąŇ

```
>>> response.xpath('//title/text()')
[<Selector (text) xpath=//title/text()>]
>>> response.css('title::text')
[<Selector (text) xpath=//title/text()>]
```

çżŞædIJåŔŕäżěçIJŃåĞž,xpath() åŠŇcss() æŰźæşŢè£ŤåŻđçŽĎæŸýSelectorListåőđä¿ŃïijŇæŸŕäÿĂ

```
>>> response.css('img').xpath('@src').extract()
[u'image1_thumb.jpg',
 u'image2_thumb.jpg',
 u'image3_thumb.jpg',
 u'image4_thumb.jpg',
 u'image5_thumb.jpg']
```

å£Ěéąžä¡£çŤÍ.extract() æL'■èČ¡æŔŘåŔŰæIJĂçżĹçŽĎæŢřæ■őïijŇåęĆæđIJä¡ääŔłæČşèŐůå¿ŮçňñäÿĂäÿ
extract_first()

```
>>> response.xpath('//div[@id="images"]/a/text()').extract_first()
u'Name: My image 1 '
```

åęĆæđIJæšąæIJL'æL'¿åĹřïijŇäijŽè£ŤåŻđNoneïijŇè£ŸåŔŕéĂL'æŃľéżŸèőd'åĂij

```
>>> response.xpath('//div[@id="not-exists"]/text()').extract_
↪first(default='not-found')
'not-found'
```

èĂŇCSSéĂL'æŃľåŹĺè£ŸåŔŕäžěä¡£çŤÍCSS3æăĞåĞĘïijŽ

```
>>> response.css('title::text').extract()
[u'Example website']
```

äÿŃéÍćæŸŕåĞääÿłæŕŤè¿ČåĚÍéÍćçŽĎçď'žä¿Ńïij Ž

```
>>> response.xpath('//base/@href').extract()
[u'http://example.com/']

>>> response.css('base::attr(href)').extract()
[u'http://example.com/']

>>> response.xpath('//a[contains(@href, "image")]/@href').extract()
[u'image1.html',
 u'image2.html',
 u'image3.html',
 u'image4.html',
 u'image5.html']

>>> response.css('a[href*=image]::attr(href)').extract()
[u'image1.html',
 u'image2.html',
 u'image3.html',
 u'image4.html',
 u'image5.html']

>>> response.xpath('//a[contains(@href, "image")]/img/@src').
 ↪extract()
[u'image1_thumb.jpg',
 u'image2_thumb.jpg',
 u'image3_thumb.jpg',
 u'image4_thumb.jpg',
 u'image5_thumb.jpg']

>>> response.css('a[href*=image] img::attr(src)').extract()
[u'image1_thumb.jpg',
 u'image2_thumb.jpg',
 u'image3_thumb.jpg',
 u'image4_thumb.jpg',
 u'image5_thumb.jpg']
```

4.3 åţŇåěŮéĂĽæŃĺåŹÍ

xpath()åŠŇcss()è£ŤåŻđçŽĎæŸŕéĂĽæŃĺåŹÍåĹŮèąĺïijŇæĽĂäžěä¡ăåŘŕäžěçżğçż■ä¡£çŤÍåőČäžňçŽĎæŮ

```
>>> links = response.xpath('//a[contains(@href, "image")]')
>>> links.extract()
[u'<a href="image1.html">Name: My image 1 <br><img src="image1_
 ↪thumb.jpg"></a>',
 u'<a href="image2.html">Name: My image 2 <br><img src="image2_
 ↪thumb.jpg"></a>',
```

```
 u'<a href="image3.html">Name: My image 3 <br><img src="image3_
 ↪thumb.jpg"></a>',
 u'<a href="image4.html">Name: My image 4 <br><img src="image4_
 ↪thumb.jpg"></a>',
 u'<a href="image5.html">Name: My image 5 <br><img src="image5_
 ↪thumb.jpg"></a>']

>>> for index, link in enumerate(links):
...     args = (index, link.xpath('@href').extract(), link.xpath(
 ↪'img/@src').extract())
...     print 'Link number %d points to url %s and image %s' % args

Link number 0 points to url [u'image1.html'] and image [u'image1_
 ↪thumb.jpg']
Link number 1 points to url [u'image2.html'] and image [u'image2_
 ↪thumb.jpg']
Link number 2 points to url [u'image3.html'] and image [u'image3_
 ↪thumb.jpg']
Link number 3 points to url [u'image4.html'] and image [u'image4_
 ↪thumb.jpg']
Link number 4 points to url [u'image5.html'] and image [u'image5_
 ↪thumb.jpg']
```

4.4 ä¡£çŤÍæ■čåĹŹèąĺè¿¿åijŔ

SelectoræIJL'äÿĂäÿłre()æŰźæşŢéĂŽè£Ğæ■čåĹŹèąĺè¿¿åijŔæŔŘåŔŰæŢřæ■őïijŇåőČè£ŤåŽđçŽĎæŸŕu

```
>>> response.xpath('//a[contains(@href, "image")]/text()').re(r
 ↪'Name:\s*(.*)')
[u'My image 1',
 u'My image 2',
 u'My image 3',
 u'My image 4',
 u'My image 5']

>>> response.xpath('//a[contains(@href, "image")]/text()').re_
 ↪first(r'Name:\s*(.*)')
u'My image 1'
```

4.5 XPathçŻÿåŕźèůŕå¿Ď

å¡Şä¡ăåţŇåěŮä¡£çŤÍXPathæŮűåĂŹïijŇäÿ■èęĄä¡£çŤÍ/åijĂåd't'çŽĎïijŇåŽăäÿžè£ŹäÿłäijŽçŻÿåŕźæŰĞæąčæă

```
>>> divs = response.xpath('//div')
>>> for p in divs.xpath('.//p'):  # extracts all <p> inside
...     print p.extract()
```

```
# æĹŰèĂĚäÿŃéİćè£ŹäÿłçŻt’æŐëä¡£çŤÍpäź§åŘŕäžě
>>> for p in divs.xpath('p'):
...     print p.extract()
```

4.6 XPathåżžèőő

ä¡£çŤÍtextä¡IJäÿžæİąäżűæŮű

éĄ£åĚ■ä¡£çŤÍ.//text()ˌçŻt’æŐëä¡£çŤÍ.

```
>>> sel.xpath("//a[contains(., 'Next Page')]").extract()
[u'<a href="#">Click here to go to the <strong>Next Page</strong></
 ↪a>']
```

//node[1]åŠŇ(//node)[1]åŇžåĹń

- //node[1]: éĂL’æŃl’æL’ĂæIJL’ä¡■äžŐçňňäÿĂäÿłå■ŘèŁĆçĆźä¡■ç¡őçŽĎnodeèŁĆçĆź
- (//node)[1]: éĂL’æŃl’æL’ĂæIJL’çŽĎnodeèŁĆçĆźïijŇçĎűåŘŐè£ŤåŻđçżŞæđIJäÿ■çŽĎçňňäÿĂäÿłnodeè

éĂŽè£Ğclassæ§ěæL’¿æŮűäijŸåĚĹèĂČèŹŚCSS

```
>> from scrapy import Selector
>>> sel = Selector(text='<div class="hero shout"><time datetime=
 ↪"2014-07-23 19:00">Special date</time></div>')
>>> sel.css('.shout').xpath('./time/@datetime').extract()
[u'2014-07-23 19:00']
```

5 ScrapyæŢŹçÍŃ05- Itemèŕęèğč

ItemæŸŕä£Ìå■ŸçżŞæđĎæŢřæ■őçŽĎåIJřæŰźïijŇScrapyåŘŕäžěåřĘèğčæđŘçżŞæđIJäžěå■ŮåĚÿå¡ćåijŔè£ŤåŻđ

ItemæŘŘä¿ŻäžĘçśżå■ŮåĚÿçŽĎAPIïijŇåźűäÿŤåŘŕäžěå¿ĹæŰźä¿£çŽĎåčřæŸŐå■ŮæőţïijŇå¿Ĺåd’ŽScrapyçżĎ

5.1 åőŽäźL’Item

åőŽäźL’IteméÌđåÿÿçőĂå■ŢïijŇåŘłéIJĂèęĄçżğæL’£scrapy.ItemçśżïijŇåźűåřĘæL’ĂæIJL’å■Ůæőţé Č¡åőŽä
Fieldçśżåđ Ńå■şåŘŕ

```
import scrapy

class Product(scrapy.Item):
```

```
    name = scrapy.Field()
    price = scrapy.Field()
    stock = scrapy.Field()
    last_updated = scrapy.Field(serializer=str)
```

5.2 Item Fields

Fieldåŕźèśąå Ŕŕç ŤÍæİěåŕźæŕŔäÿłå■Ůæőţæ ŇĞåőŽåĚČæŢřæ■őãĂĆä¿ŃåęĆäÿŁéİć last_updated çŽĎåžŔ

5.3 Itemä¡£çŤÍçď žä¿Ń

ä¡ăäijŽçIJŃåĹřItemçŽĎä¡£çŤÍèů§Pythonäÿ■çŽĎå■ŮåĚÿAPIéİđåÿÿçśżäijij

åĹŻåżžItem

```
>>> product = Product(name='Desktop PC', price=1000)
>>> print product
Product(name='Desktop PC', price=1000)
```

èŐůåŔŰåĂij

```
>>> product['name']
Desktop PC
>>> product.get('name')
Desktop PC

>>> product['price']
1000

>>> product['last_updated']
Traceback (most recent call last):
    ...
KeyError: 'last_updated'

>>> product.get('last_updated', 'not set')
not set

>>> product['lala'] # getting unknown field
Traceback (most recent call last):
    ...
KeyError: 'lala'

>>> product.get('lala', 'unknown field')
'unknown field'
```

```
>>> 'name' in product  # is name field populated?
True

>>> 'last_updated' in product  # is last_updated populated?
False

>>> 'last_updated' in product.fields  # is last_updated a declared␣
↪field?
True

>>> 'lala' in product.fields  # is lala a declared field?
False
```

èő¿ç¡őåĂij

```
>>> product['last_updated'] = 'today'
>>> product['last_updated']
today

>>> product['lala'] = 'test' # setting unknown field
Traceback (most recent call last):
    ...
KeyError: 'Product does not support field: lala'
```

èő£éŮőæĽĂæIJĽçŽĎåĂij

```
>>> product.keys()
['price', 'name']

>>> product.items()
[('price', 1000), ('name', 'Desktop PC')]
```

5.4 Item Loader

Item Loaderäÿžæ ĹŚäžňæŔŘä¿ŻäžĘçŤ§æĹŘItemçŽĎçŻÿå¡Şä¿£åĹ'çŽĎæŰźæşŢãĂĆItemäÿžæŁŞåŔŰçŽĎæ LoaderåŔŕäžěèőľæĹŚäžňéÍđåÿÿæŰźä¿£çŽĎåřĘè¿ŞåĚěåąńåĚĚåĹřåőźåŹĺäÿ■ãĂĆ

äÿŃéÍćæĹŚäžňéĂŽè£ĞäÿĂäÿłä¿Ńå■ŘæÍěåśŢçd'žäÿĂèĹňä¡£çŤĺæŰźæşŢïijŽ

```
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()
```

æşÍæĎŔäÿŁéİćçŽĎnameå■Ůæőţæ ŸŕäžŐäÿďäÿłxpathèůŕå¿Ďæůžçţ'ŕåŁäåŘŐå¿ŮåĹřãĂĆ

5.5 è¿ŞåĚě/è¿ŞåĞžåď'ĎçŘĘåŹÍ

æŕŔäÿłItem Loaderåŕźæ ŕŔäÿłFieldéČ¡æIJL'äÿĂäÿłè¿ŞåĚěåď'ĎçŘĘåŹÍåŠŇäÿĂäÿłè¿ŞåĞžåď'ĎçŘĘåŹÍãĂĆ
load_item()æŮűåĘ■æL'ğèąŇè¿ŞåĞžåď'ĎçŘĘåŹÍïijŇè£ŤåŻđæIJĂçżĹçżŞæđIJãĂĆ

```
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)
```

æL'ğèąŇæţAçÍŃæŸŕè£ŹæăůçŽĎïijŽ

1. xpath1äÿ■çŽĎæŢřæ■őèćńæŔŘåŔŰåĞžæİěïijŇçĎűåŘŐäijăè¿ŞåĹřnameå■Ůæőţ çŽĎè¿ŞåĚěåď'ĎçŘĘ LoaderéĞŇéİć(è£ŹæŮűåĂŹæšąæIJL'èţŃåĂijçżZitem)
2. xpath2æŢřæ■őèćńæŔŘåŔŰåĞžæİěïijŇçĎűåŘŐäijăè¿ŞçżŹ(1)äÿ■åŘŇæăůçŽĎè¿ŞåĚěåď'ĎçŘĘåŹÍïijŇ
3. èů§2äÿĂæăů
4. èů§3äÿĂæăůïijŇäÿ■è£Ğè£ŹæňąæŸŕçŻt'æŐěçŽĎå■ŮéİćåŮçňęäÿšåĂijïijŇåĚĹè¡ňæ■ćæĹŘäÿĂäÿłå■Ţ
5. äÿŁéİć4æ■ěçŽĎæŢřæ■őèćńäijăè¿ŞçżŹnameçŽĎè¿ŞåĞžåď'ĎçŘĘåŹÍïijŇářĘæIJĂçżĹçŽĎçżŞæđIJèţŃå

5.6 èĞłåőŽäźL'Item Loader

ä¡£çŤÍçśżåőŽäźL'èŕ■æşŢïijŇäÿŃéİćæŸŕäÿĂäÿłä¿Ńå■Ř

```
from scrapy.loader import ItemLoader
from scrapy.loader.processors import TakeFirst, MapCompose, Join

class ProductLoader(ItemLoader):

    default_output_processor = TakeFirst()

    name_in = MapCompose(unicode.title)
    name_out = Join()

    price_in = MapCompose(unicode.strip)
```

```
    # ...
```

éĂŽè£Ğ_inåŠŇ_outåŘŐçijĂæİěåőŽäźL'è¿ŞåĚěåŠŇè¿ŞåĞžåď'ĎçŘĘåŹÍïijŇåźűäÿŤè£ŸåŔŕäžěåőŽäźL'éżŸ
default_input_processoråŠŇItemLoader.default_input_processor.

5.7 ålJÍFieldåőŽäźL'äÿ■ačřæŸŐè¿ŞåĚě/è¿ŞåĞžåď'ĎçŘĘåŹÍ

è£ŸæIJL'äÿłåIJřæŰźåŔŕäžěéİđåÿÿæŰźä¿£çŽĎæůžåŁäè¿ŞåĚě/è¿ŞåĞžåď'ĎçŘĘåŹÍïijŇéĆčářśæŸŕçŻť'æŐěåIJ

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

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

class 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(),
    )
```

äijŸåĚĹçžğïijŽ

1. åIJÍItem Loaderäÿ■åőŽäźL'çŽĎfield_inåŠŇfield_out
2. FiledåĚČæŢřæ■ő(input_processoråŠŇoutput_processoråĚşéŤőå■Ů)
3. Item Loaderäÿ■çŽĎéżŸèőď'çŽĎ

TipsïijŽäÿĂèĹňæİěèőšïijŇåřĘè¿ŞåĚěåď'ĎçŘĘåŹÍåőŽäźL'åIJÍItem LoaderçŽĎåőŽäźL'äÿ■field_inïijŇçĎűåŘŐåřĘè¿ŞåĞžåď'ĎçŘĘåŹÍåőŽäźL'åIJÍFieldåĚČæŢřæ■őäÿ■

5.8 Item LoaderäÿŁäÿŃæŰĞ

Item LoaderäÿŁäÿŃæŰĞèćńæL'ĂæIJL'è¿ŞåĚě/è¿ŞåĞžåď'ĎçŘĘåŹÍåĚśäžńïijŇærŤåęĆä¡ăæIJL'äÿĂäÿłèğčæđŘ

```
def parse_length(text, loader_context):
    unit = loader_context.get('unit', 'm')
    # ... length parsing code goes here ...
    return parsed_length
```

åĹİåğŃåŇŰåŠŇä£őæŤźäÿŁäÿŃæŰĞçŽĎåĂij

```
loader = ItemLoader(product)
loader.context['unit'] = 'cm'

loader = ItemLoader(product, unit='cm')

class ProductLoader(ItemLoader):
    length_out = MapCompose(parse_length, unit='cm')
```

5.9 åĘĚç¡őçŽĎåď'ĎçŘĘåŹÍ

1. `Identity` åŢťäź§äÿ■åĄŽ
2. `TakeFirst` è£ŤåŻđçňňäÿĂäÿłéÍđçl'žåĂijïijŇéĂŽåÿÿçŤÍä¡IJè¿ŞåĞžåď'ĎçŘĘåŹÍ
3. `Join` åřĘçżŞæđIJè£đèţůæÍěïijŇéżŸèőď'ä¡£çŤÍçl'žæäij' '
4. `Compose` åřĘåĞ¡æŢřéŞ¿æŐèèţůæÍěå¡ćæĹŘçőąéĄŞæţĄïijŇäžğçŤ§æIJĂåŘŐçŽĎè¿ŞåĞž
5. `MapCompose` èů§äÿŁéÍćçŽĎ`Compose`çśżäijijïijŇåŇžåĹńåIJÍäžŐåĘĚéČÍçżŞæđIJåIJÍåĞ¡æŢřäÿ■çŽĎ
   åőČçŽĎè¿ŞåĚěåĂijæŸŕåŘŕè£■äžčçŽĎïijŇéęŰåĚĹářĘçňňäÿĂäÿłåĞ¡æŢřä¿Íæňąä¡IJçŤÍäžŐæL'ĂæIJL'å
6. `SelectJmes` ä¡£çŤÍjsonèůŕå¿ĎæÍěæ§ěèŕćåĂijåźűè£ŤåŻđçżŞæđIJ

6 ScrapyæŢŹçÍŃ06- Item Pipeline

å¡ŞäÿĂäÿłitemèćńèIJŸèŻŻçĹňåŘŰåĹřäźŃåŘŐäijŽèćńåŘŚéĂĄçżŹItem PipelineïijŇçĎűåŘŐåď'ŽäÿłçżĎäžűæŇL'çĚğéąžåžŘåď'ĎçŘĘè£ŹäÿłitemãĂĆ æŕŘäÿłItem PipelineçżĎäžűåĚűåőđåřśæŸŕäÿĂäÿłåőđçŐřäžĘäÿĂäÿłçőĂå■ŢæŰźæşŢçŽĎPythonçśżãĂĆäžŰäžňæŐěåŘŮä

ä¡£çŤÍItem PipelineçŽĎåÿÿçŤÍåIJžæŹŕïijŽ

- æÿĚçŘĘHTMLæŢřæ■ő
- élŇèŕĄèćńæŁŞåŘŰçŽĎæŢřæ■ő(æčĂæ§ěitemæŸŕåŘęåŇĚåŘńæ§ŘäžŻå■Ůæőţ)
- éĞ■åď'■æĂğæčĂæ§ě(çĎűåŘŐäÿćåijČ)
- åřĘæŁŞåŘŰçŽĎæŢřæ■őå■ŸåĆÍåĹřæŢřæ■őåžŞäÿ■

6.1 çijŰåĘŹèĞłåůśçŽĎPipeline

åőŽäźL'äÿĂäÿłPythonçśżïijŇçĎűåŘŐåőđçŐřæŰźæşŢ`process_item(self, item, spider)`å■şåŘŕïijŇè£ŤåŻđäÿĂäÿłå■ŮåĚÿæĹŰItemïijŇæĹŰèĂĚæŁŻåĞž`DropItem`åijĆåÿÿäÿćåijČè£ŹäÿæĹŰèĂĚè£ŸåŘŕäžěåőđçŐřäÿŃéÍćåĞääÿłæŰźæşŢïijŽ

- `open_spider(self, spider)` èIJŸèŻŻæL'ŞåijĂçŽĎæŮűæL'ğèąŇ
- `close_spider(self, spider)` èIJŸèŻŻåĚşéŮ■æŮűæL'ğèąŇ

- `from_crawler(cls, crawler)` åŘŕèő£éŮőæäÿå£ČçżĎäžűæŕŤåęĆéĚ■ç¡őåŠŇä£ąåŔůïijŇåźűæş

6.2 Item Pipelineçď˙žä¿Ń

äžůæäijéłŇèŕĄ

æĹŚäžňéĂŽè£ĞäÿĂäÿłäžůæäijéłŇèŕĄä¿Ńå■ŘæİěçIJŃçIJŃæĂŐæåůä¡£çŤĺ

```
from scrapy.exceptions import DropItem

class PricePipeline(object):

    vat_factor = 1.15

    def process_item(self, item, spider):
        if item['price']:
            if item['price_excludes_vat']:
                item['price'] = item['price'] * self.vat_factor
            return item
        else:
            raise DropItem("Missing price in %s" % item)
```

åřĘitemåĘŹåĚějsonæŰĞäžű

äÿŃéİćçŽĎè£ŹäÿłPipelineåřĘæĽ˘ĂæIJL˘çŽĎitemåĘŹåĚěåĹřäÿĂäÿłå■ŢçŃňçŽĎjsonæŰĞäžűïijŇäÿĂèąŇäÿĂ

```
import json

class JsonWriterPipeline(object):

    def __init__(self):
        self.file = open('items.jl', 'wb')

    def process_item(self, item, spider):
        line = json.dumps(dict(item)) + "\n"
        self.file.write(line)
        return item
```

åřĘitemå■ŸåĆĺåĹřMongoDBäÿ■

è£Źäÿłä¿Ńå■Řä¡£çŤĺpymongoæİěæijŢçd'žæĂŐæåůèőšitemä£Ìå■ŸåĹřMongoDBäÿ■ãĂĆ
MongoDBçŽĎåIJřåİĂåŠŇæŢřæ■őåžŞåŘ■åIJĺéĚ■ç¡őäÿ■æŇĞåőŽïijŇè£ŹäÿłäżNå■ŘäÿżèęĄæŸřåŘŚä¡ăåśŢç

```
import pymongo

class MongoPipeline(object):
```

```
    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(dict(item))
        return item
```

éĞ■åď■è£Ğæżď'åŹÍ

åĄĞèő¿æĹŚäżňçŽĎiteméĞŇéİćçŽĎidå■ŮåĚÿæŸŕåŤŕäÿĂçŽĎïijŇä¡ĘæŸŕæĹŚäżňçŽĎèIJŸèŻŻè£ŤåŻđäžĘå

```
from scrapy.exceptions import DropItem

class DuplicatesPipeline(object):

    def __init__(self):
        self.ids_seen = set()

    def process_item(self, item, spider):
        if item['id'] in self.ids_seen:
            raise DropItem("Duplicate item found: %s" % item)
        else:
            self.ids_seen.add(item['id'])
            return item
```

6.3 æ£Äæť'žäÿÄäÿłItem PipelineçżĎäžű

ä¡åå£ĚéążåIJĺéĚ■ç¡őæŰĞäžűäÿ■åřĘä¡äéIJÄèęĄæ£Äæť'żçŽĎPiplineçżĎäžűæůżåŁäåĹřITEM_PIPELINESä

```
ITEM_PIPELINES = {
    'myproject.pipelines.PricePipeline': 300,
    'myproject.pipelines.JsonWriterPipeline': 800,
}
```

åŘŐéİćçŽĎæŢřå■Ůèąĺçď žåőČçŽĎæL'ğèąŇéąžåžŔïijŇäžŐä¡ŐåĹřéńŸæL'ğèąŇïijŇèŇČåŻť 0-1000

6.4 Feed exports

è£ŹéĞŇéąžä¿£æŔŘäÿŃFeed exportsïijŇäÿĂèĹňæIJL'çŽĎçĹňèŹńçŻť'æŐěářEçĹňåŔŰçżŞæđIJåžŘåĹŮåŇŰå

```
* FEED_FORMAT= json # json|jsonlines|csv|xml|pickle|marshal
* FEED_URI= file:///tmp/export.csv|ftp://user:pass@ftp.example.com/
 ↪path/to/export.csv|s3://aws_key:aws_secret@mybucket/path/to/
 ↪export.csv|stdout:
* FEED_EXPORT_FIELDS = ["foo", "bar", "baz"] #␣
 ↪è£Źäÿłåİ Jĺåŕijå Ğžcsvç ŽĎæŮűåĂŹæIJL'çŤĺ
```

6.5 èŕůæśĆåŠŇåŞ■åžŤ

Scrapyä¡£çŤĺRequeståŠŇResponseåŕźèśąæİěçĹňåŔŰç¡ŚçńŹãĂĆRequeståŕźèśąèćńèIJŸèŻŽçŤ§æĹŘïi

çžŹåŻđèřČåĞ¡æŢřäijäéĂŠéćĺåď ŰçŽĎåŔĆæŢř

Requeståŕźèśąç Ť§æĹŘçŽĎæŮűåĂŹäijŽéĂŽè£ĞåĚşéŤőå■ŮåŔĆæŢřcallbackæŇĞåőŽåŻđèřČåĞ¡æŢřïi meta

```
def parse_page1(self, response):
    item = MyItem()
    item['main_url'] = response.url
    request = scrapy.Request("http://www.example.com/some_page.html
 ↪",
                             callback=self.parse_page2)
    request.meta['item'] = item
    return request

def parse_page2(self, response):
    item = response.meta['item']
    item['other_url'] = response.url
    return item
```

Requestå■Řçśż

Scrapyäÿžå ŘĎçğ■äÿ■åŘŇçŽĎåIJžæŹŕåĘĚç¡őäžĘå¿Ĺåd'ŽRequestå■Řçśżïij Ňä¡ãè£ŸåŔŕäžěçżğæL'£åőČèĞłåć

FormRequestè£ŹäÿłäÿŞéŮĺäÿžformèąĺå■Ţèő¿èőąïijŇæĺąæŃ§èąĺå■ŢæŔŘäžd'çŽĎçd'žä¿Ń

```
return [FormRequest(url="http://www.example.com/post/action",
                    formdata={'name': 'John Doe', 'age': '27'},
                    callback=self.after_post)]
```

æĹŚäžňåĘ■æİěäÿĂäÿłä¿Ńå■ŘæĺąæŃ§çŤĺæĹůçŹżå¡ŢïijŇä¡£çŤĺäžĘFormRequest.from_response()

```
import scrapy

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):
        # check login succeed before going on
        if "authentication failed" in response.body:
            self.logger.error("Login failed")
            return

        # continue scraping with authenticated session...
```

Responseå■Řçśż

äÿĂäÿłscrapy.http.ResponseårźèśąäžčèąĺäžĘäÿĂäÿłHTTPçŻÿåžŤïijŇéĂŽåÿÿæŸŕèćńäÿŃè¡¡åŹĺäÿŃè

- TextResponse åIJĺå§žæIJňResponseçśżå§žçąĂäźŃäÿŁáćđåŁääžĘçijŰçăĄåŁ§èČ¡ïijŇäÿŞéŮĺçŤĺäžŐä
- HtmlResponse æ■d'çśżæŸŕTextResponseçŽĎå■Řçśżïij ŇéĂŽè£Ğæ§ěèŕćHTMLçŽĎmeta http-equivåśđæĂğåőđçŐřäžĘçijŰçăĄèĞłåŁĺåŔŚçŐř
- XmlResponse æ■d'çśżæŸŕTextResponseçŽĎå■Řçśżïij ŇéĂŽè£Ğæ§ěèŕćXMLåčřæŸŐåőđçŐřçijŰçă

7 ScrapyæŢŹçĺŃ07- åĘĚç¡őælJ■åŁą

Scrapyä¡£çŤĺPythonåĘĚç¡őçŽĎçŽĎæŮěå£Ůçżçż§æİěèőřå¡ŢäžŃäžűæŮěå£ŮãĆć æŮěå£ŮéĚ■ç¡ő

```
LOG_ENABLED = true
LOG_ENCODING = "utf-8"
LOG_LEVEL = logging.INFO
LOG_FILE = "log/spider.log"
LOG_STDOUT = True
LOG_FORMAT = "%(asctime)s [%(name)s] %(levelname)s: %(message)s"
LOG_DATEFORMAT = "%Y-%m-%d %H:%M:%S"
```

ä¡£çŤĺäź§å¿ĹçőĂå■Ţ

```
import logging
logger = logging.getLogger(__name__)
logger.warning("This is a warning")
```

åęĆæđIJåIJĺSpideréĞŇéİćä¡£çŤĺïijŇéĆčåřśæŻť´çőĂå■ŢäžĘïijŇåŻääÿžloggerårśæŸŕåőČçŽĎäÿĂäÿłåőđä¿Ňå

```
import scrapy

class MySpider(scrapy.Spider):

    name = 'myspider'
    start_urls = ['http://scrapinghub.com']

    def parse(self, response):
        self.logger.info('Parse function called on %s', response.
↪url)
```

7.1 åŔŚéĂĄemail

ScrapyåŔŚéĂĄemailå§žäžŐTwisted non-blocking IOåőđçŐřïijŇåŔłéIJĂåĞääÿłçőĂå■ŢéĚ■ç¡őå■şåŔřãĂĆ åĹİåğŃåŇŰ

```
mailer = MailSender.from_settings(settings)
```

åŔŚéĂĄäÿ■åŇĚåŘńéŹĎäżű

```
mailer.send(to=["someone@example.com"], subject="Some subject",␣
↪body="Some body", cc=["another@example.com"])
```

éĚ■ç¡ő

```
MAIL_FROM = 'scrapy@localhost'
MAIL_HOST = 'localhost'
MAIL_PORT = 25
MAIL_USER = ""
MAIL_PASS = ""
MAIL_TLS = False
MAIL_SSL = False
```

7.2 åŘŇäÿĂäÿłè£ŻçÍŃè£ŘèąŇåď'ŽäÿłSpider

```
import scrapy
from scrapy.crawler import CrawlerProcess

class MySpider1(scrapy.Spider):
    # Your first spider definition
    ...

class MySpider2(scrapy.Spider):
    # Your second spider definition
    ...

process = CrawlerProcess()
process.crawl(MySpider1)
process.crawl(MySpider2)
process.start() # the script will block here until all crawling␣
↪jobs are finished
```

7.3 åĹĘåÿČåijŔçĹňèŹń

ScrapyåźűæšąæIJL'æŔŘä¿ŻåĘĚç¡őçŽĎåĹĘåÿČåijŔæŁŞåŔŰåŁ§èČ¡ïijŇäÿ■è£ĞæIJL'å¿Ĺåď'ŽæŰźæşŢåŔŕäżěåęĆæđIJä¡ăæIJL'å¿Ĺåď'ŽäÿłspiderïijŇæIJĂçŐĂå■ŢçŽĎæŰźåijŔåřśæŸŕåŘŕåŁĺåď'Žäÿłscrapydåőđä¿ŃïijŇçåęĆæđIJä¡ăæČşåď'ŽäÿłæIJžåŹĺè£ŘèąŇåŘŇäÿĂäÿłspiderïijŇåŔŕäżěåřĘurlåĹĘçL'ĞåŘŐäžď'çżŹæŕŔäÿłæIJžåŹ

```
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
```

çĎűåŘŐè£ŘèąŇ3äÿłscrapydåőđä¿ŃïijŇåĹĘåĹńåŘŕåŁĺåőČäžňïijŇåźűäijăéĂŠpartåŔĆæŢř

```
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
```

7.4 éŸšæ■ćèćńåřĄçŽĎç■ŰçŢě

äÿĂäžŻç¡ŚçńŹåőđçŐřäžĘäÿĂäžŻç■ŰçŢěæÌěçęÆæ■ćçĹňèŹńæÌěçĹňåŔŰåőČäžňçŽĎç¡Śéąţã ĂĆæIJL'çŽĎærĎäÿŃéÌćæŸŕåŕźäžŐè£ŹäžŻç¡ŚçńŹçŽĎäÿĂäžŻæIJL'çŤĺçŽĎåžžèőőïijŽ

- ä¡£çŤĺuser agentæśăăĂĆäź§åřśæŸŕærŔæňąåŔŚéĂĄçŽĎæŮűåĂŹéŽŔæIJžäżŐæśăäÿ■éĂL'æŃľäÿ■äÿĂ

- çęĄæ■ćCookieïijŇæ§ŘäžŻç¡ŚçńŹäij ŽéĂŽè£ĞCookieèŕĘåĹńçŤĺæĹůèžńäż¡ïijŇçęĄçŤĺåŘŐä¡£å¿ŮæIJ■
- èő¿ç¡ődownload_delayäÿŃè¡¡åžűè£§ïijŇæŢřå■Ůèő¿ç¡őäÿž5çğŠïijŇèűŁåď ğèűŁåőĽåĚĺ
- åęĆæđIJæIJL’åŘŕèČ¡çŽĎèŕÍår¡éĞŔä¡£çŤĺGoogle cacheèŐůåŔŰç¡Śéąţ ïijŇèĂŇäÿ■æŸŕçŻť’æŐěèő£éŮő
- ä¡£çŤĺäÿĂäÿłè¡őè¡ňIPæśăïijŇä¿ŃåęĆåĚ■èť’źçŽĎTor projectæĹŰèĂĚæŸŕäžŸèť’źçŽĎProxyMesh
- ä¡£çŤĺåď ğåđŃåĹĘåÿČåijŔäÿŃè¡¡åŹĺïijŇè£Źæăůåřśè Č¡åőŇåĚĺéĄ£åĚ■èćńářĄäžĘïijŇåŘłéIJÀèęĄåĚşæ

åęĆæđIJè£ŹäžŻè£ŸæŸŕæŮăæşŢéĄ£åĚ■èćńçęĄïijŇåŘŕäžěèĂČèŹŚåŢĘäÿŽæŤŕæŇĄ

8 ScrapyæŢŹçĺŃ08- æŰĞäžűäÿŐåŻ¿çĽĞ

Scrapyäÿžæ ĹŚäžňæŔŘä¿ŻäžĘåŘŕéĞ■çŤĺçŽĎitem pipelinesäÿžæ§Řäÿłç Ľźåő ŽçŽĎItemåŐžäÿŃè¡¡æŰĞäžűã
éĂŽåÿÿæÍěèŕť’ä¡ääijŽéĂĽ’æŃľ’ä¡£çŤĺFiles PipelineæĹŰImages PipelineãĂĆ

è£Źäÿd’äÿłçőąéĄŞéČ¡åőđçŐřäžĘïijŽ

- éĄ£åĚ■éĞ■åd’■äÿŃè¡¡
- åŘŕäžěæŇĞåőŽäÿŃè¡¡åŘŐä£Íå■ŸçŽĎåIJřæŰź(æŰĞäžűçşżçż§çŻőå¡Ţäÿ■,Amazon S3äÿ■)

Images Pipelineäÿžåď ĎçŘĘåŻ¿çĽĞæŔŘä¿ŻäžĘéćÍåď ŰçŽĎåŁ§èČ¡ïijŽ

- åřĘæĽ’ĂæIJL’äÿŃè¡¡çŽĎåŻ¿çĽĞæăijåijŔè¡ňæ■ćæĹŘæŹőéĂŽçŽĎJPGåźűä¡£çŤĺRGBéćIJèĽ’šæĺąåijŔ
- çŤ§æĹŘçijľçŢěåŻ¿
- æčĂæ§ěåŻ¿çĽĞçŽĎåő¡åžęåŠŇéńŸåžęçąőä£ÍåőČäžňæżąèűşæIJĂåřŔçŽĎåřžåŕÿéŹŘåĹű

çőąéĄŞåŘŇæŮűäijŽåIJÍåĘĚéČÍä£Íå■ŸäÿĂäÿłèćńèřČåžęäÿŃè¡¡çŽĎURLåĹŮèąÍïijŇçĎűåŘŐářĘåŇĚåŘńçŻÿ

8.1 ä¡£çŤĺFiles Pipeline

äÿĂèĹňæĹŚäžňäijŽæŇĽ’çĚğäÿŃéÍćçŽĎæ■ěélď’æÍěä¡£çŤĺæŰĞäžűçőąéĄŞïijŽ

1. åIJÍæ§Řäÿł Spideräÿ■ïijŇä¡äçĹňåŔŰäÿĂäÿłitemåŘŐïijŇářĘçŻÿåžŢçŽĎæŰĞäžűURLæŢ¿åĚě`file_u`
2. itemèćńè£ŤåŻđäźŃåŘŐářśäijŽè¡ňäžd’çżŹitem pipeline
3. å¡Şè£Źäÿłitemåĺřè¿¿`FilesPipeline`æŮűïijŇåIJÍ`file_urls`å■Ůæőţäÿ■çŽĎURLåĹŮèąÍäijŽéĂŽ
4. å¡ŞæŰĞäžűèćńäÿŃè¡¡åőŇäźŃåŘŐïijŇçżŞæđIJäijŽèćńètŃåĂijçżŹåŖęäÿĂäÿł`files`å■Ůæőţã ĆČè£Źäÿ

8.2 ä¡£çŤĺImages Pipeline

`ImagesPipeline`èů§`FilesPipeline`çŽĎä¡£çŤĺåůőäÿ■åď ŽïijŇäÿ■è£Ğä¡£çŤĺçŽĎå■Ůæőţå Ř■äÿ■äÿĂ
ä¡£çŤĺ`ImagesPipeline`çŽĎåě¡åď ĎæŸŕä¡ăåŔŕäžěéĂŽè£ĞéĚ■ç¡őæÍěæŔŘä¿ŻéćÍåď ŰçŽĎåŁ§èČ¡ïijŇæŕŤ
`ImagesPipeline`ä¡£çŤĺPillowæÍěçŤ§æĹŘçijľçŢěåŻ¿äžěåŘŁè¡ňæ■ćæĹŘæăĞåĞęçŽĎJPEG/RGBæăijåijĺ

8.3 ä¡£çŤĺä¿Ńå■Ř

èęĄä¡£çŤĺåłŠä¡Şçőąé ĄŞïijŇèŕůåĚĹåIJĺéĚ■ç¡őæŰĞäžűäÿ■æĽŞåijĂåőČ

```
# åŘŇæŮűä¡£çŤĺåŽ¿çL'ĞåŠŇæŰĞäžűçőąéĄŞ
ITEM_PIPELINES = {
                  'scrapy.pipelines.images.ImagesPipeline': 1,
                  'scrapy.pipelines.files.FilesPipeline': 2,
                 }
FILES_STORE = '/path/to/valid/dir'  # æŰĞäžűå■ŸåĆĺèůŕå¿Ď
IMAGES_STORE = '/path/to/valid/dir' # åŽ¿çL'Ğå■ŸåĆĺèůŕå¿Ď
# 90 days of delay for files expiration
FILES_EXPIRES = 90
# 30 days of delay for images expiration
IMAGES_EXPIRES = 30
# åŽ¿çL'Ğçijl'çŢěåŽ¿
IMAGES_THUMBS = {
    'small': (50, 50),
    'big': (270, 270),
}
# åŽ¿çL'Ğè£Ğæżd'åŹĺïijŇæIJĂåřRéńŸåžęåŠŇåő¡åžę
IMAGES_MIN_HEIGHT = 110
IMAGES_MIN_WIDTH = 110
```

äÿĂäÿłä¡£çŤĺäžEçijl'çŢěåŻ¿çŽĎäÿŃè¡¡ä¿Ńå■ŘäijŽçŤ§æĹŘåęĆäÿŃåŻ¿çL'ĞïijŽ

```
<IMAGES_STORE>/full/63bbfea82b8880ed33cdb762aa11fab722a90a24.jpg
<IMAGES_STORE>/thumbs/small/
↪63bbfea82b8880ed33cdb762aa11fab722a90a24.jpg
<IMAGES_STORE>/thumbs/big/63bbfea82b8880ed33cdb762aa11fab722a90a24.
↪jpg
```

çĎűåŘŐïijŇæ§ŘäÿłItemè£ŤåŻđæŮűïijŇæIJL'`file_urls`æĹŰ`image_urls`ïijŇåźűäÿŤå■ŸåIJĺçŻÿåžŢçŽ

```
import scrapy

class MyItem(scrapy.Item):

    # ... other item fields ...
    image_urls = scrapy.Field()
    images = scrapy.Field()
```

8.4 èĞłåőŽäźĽåłŠä¡ŞçőąéĄŞ

åęĆæđIJä¡äè£ŸéIJĂèęĄæŻť'åŁååď'■æİĆçŽĎåŁ§èČ¡ïijŇæČşèĞłåőŽäźĽäÿŃè¡¡åłŠä¡ŞéĂżè¿ŚïijŇèŕůåŔĆèĂČ

äÿ■çőąæŸŕæĽľ'åśŢ`FilesPipeline`è£ŸæŸŕ`ImagesPipeline`,éČ¡åŔłéIJĂéĞ■åĘŹäÿŃéİćäÿd'äÿłæŰźa

- `get_media_requests(self, item, info)`,è£ŤåŻđäÿĂäÿł`Request`årźèśą

- `item_completed(self, results, item, info)`,å¡Şäÿ Łé Ů Íç ŽĎ Request täÿ Ńè¡¡ åő Ň æ Ĺ Řå Ř

äÿ Ńé Ić æ Ÿŕ äÿ Ă äÿ łæ L' l' åś Ţ `ImagesPipeline` çŽĎ ä¿Ńå■Řïij Ň æ Ĺ Śå Ŕ łå Ŕ Ű pathä£ąæ Ą ŕïij Ň åźűår Ę åő Čę Ţ Ńç ż

```
import scrapy
from scrapy.pipelines.images import ImagesPipeline
from scrapy.exceptions import DropItem

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")
        item['image_paths'] = image_paths
        return item
```

9 ScrapyæŢŹçÍŃ09- éČÍç¡š

æIJňçŕĞäÿżèęĄäżŃçż■äÿd'çğ■éČÍç¡šçĹňèŹńçŽĎæŰźæąĹãĂĆåęĆæđIJäżĚäżĚåIJÍåijĂåŔŚèřČèŕŢçŽĎæŮűåĂ

- Scrapyd åijĂæžŘæŰźæąĹ
- Scrapy Cloud äžŚæŰźæąĹ

9.1 éČÍç¡šåĹřScrapyd

ScrapydæŸŕäÿĂäÿłåijĂæžŘè¡ŕäżűïijŇçŤÍæİèè£ŘèąŇèIJŸèŻŽçĹňèŹńãĂĆåőČæŔŘä¿ŻäžĘHTTP APIçŽĎæIJ■åŁąåŹÍïijŇè£ŸèČ¡è£ŘèąŇåŠŇçŻŚæŐğScrapyçŽĎèIJŸèŻŽ

èęĄéČÍç¡šçĹňèŹńåĹřScrapydïijŇéIJĂèęĄä¡£çŤÍåĹřscrapyd-clientéČÍç¡šåůěåĚůéŻĘïijŇäÿŃéIćæĹŚæijŤçd'žä

ScrapydéĂŽåÿÿäżěåőĹæŁd'è£ŽçÍŃdaemonå¡ćåijŔè£ŘèąŇïijŇçŻŚåŘňspiderçŽĎèŕůæśĆïijŇçĎůåŘŐäÿžæŕŔ `crawl myspider`,åŘŇæŮűScrapydè£ŸèČ¡äžěåd'Žè£ŽçÍŃæŰźåijŔåŘŕåĹÍïijŇéĂŽè£ĞéĚ■ç¡ő`max_proc`

åőL'èčĚ

ä¡£çŤÍpipåőL'èčĚ

```
pip install scrapyd
```

åIJÍubuntuçşżçż§äÿŁéIć

```
apt-get install scrapyd
```

éĚ■ç¡ő

éĚ■ç¡őæŰĞäżűåIJřåİĂïijÑäijŸåĚĹçžğäżŐä¡ŐåĹřéńŸ

- /etc/scrapyd/scrapyd.conf (Unix)
- /etc/scrapyd/conf.d/* (in alphabetical order, Unix)
- scrapyd.conf
- ~/.scrapyd.conf (users home directory)

åĚůä¡ŞåŔĆæŢřåŔĆèĂČscrapydéĚ■ç¡ő

çőĂå■ŢçŽĎä¿Ńå■Ř

```
[scrapyd]
eggs_dir    = eggs
logs_dir    = logs
items_dir   =
jobs_to_keep = 5
dbs_dir     = dbs
max_proc    = 0
max_proc_per_cpu = 4
finished_to_keep = 100
poll_interval = 5
bind_address = 0.0.0.0
http_port   = 6800
debug       = off
runner      = scrapyd.runner
application = scrapyd.app.application
launcher    = scrapyd.launcher.Launcher
webroot     = scrapyd.website.Root

[services]
schedule.json     = scrapyd.webservice.Schedule
cancel.json       = scrapyd.webservice.Cancel
addversion.json   = scrapyd.webservice.AddVersion
listprojects.json = scrapyd.webservice.ListProjects
listversions.json = scrapyd.webservice.ListVersions
listspiders.json  = scrapyd.webservice.ListSpiders
delproject.json   = scrapyd.webservice.DeleteProject
delversion.json   = scrapyd.webservice.DeleteVersion
listjobs.json     = scrapyd.webservice.ListJobs
daemonstatus.json = scrapyd.webservice.DaemonStatus
```

éČÍç¡š

ä¡£çŤÍscrapyd-clientæIJĂæŰźä¿£ïijŇ Scrapyd-clientæŸŕscrapydçŽĎäÿĂäÿłåőćæĹůçńŕïijŇåőČæŔŘä¿ŻäžĘs

éĂŽåÿÿårĘä¡ăçŽĎåůěçÍŃéČÍç¡šåĹřScrapydéIJĂèęĄäÿd'äÿłæ■ěéłd'ïijŽ

1. årĘåůěçÍŃæL'ŞåŇĚæĹŘpythonèŻŃïijŇä¡ăéIJĂèęĄåőL'èčĚsetuptools
2. éĂŽè£ĞaddversionjsonçźĹçńŕårĘè§ŠèŻĞèŻŃäÿŁäijăèĞşScrapdæIJ■åŁąåŹÍ

ä¡ääŔŕäžěåIJÍä¡ăçŽĎåůěçÍŃéĚ■ç¡őæŰĞäžűscrapy.cfgåőŽäźL'ScrapydçŻőæăĞ

```
[deploy:example]
url = http://scrapyd.example.com/api/scrapyd
username = scrapy
password = secret
```

åĹŮåĞžæL'ĂæIJL'åŔŕçŤÍçŻőæăĞä¡£çŤÍåŚ¡äžd'

```
scrapyd-deploy -l
```

åĹŮåĞžæ§Řäÿ łçŻőæăĞäÿŁéİćæL'ĂæIJL'åŔŕè£ŘèąŇçŽĎåůěçÍŃïijŇæL'ğèąŇåŚ¡äžd'

```
scrapyd-deploy -L example
```

åĚĹcdåĹřåůěçÍŃæăźçŻőå¡ŢïijŇçĎűåŘŐä¡£çŤÍåęĆäÿŃåŚ¡äžd'æİěéČÍç¡šïijŽ

```
scrapyd-deploy <target> -p <project>
```

ä¡ăè£ŸåŔŕäžěåőŽäźL'éżŸèőd'çŽĎtargetåŠŇprojectïijŇçIJAçŽĎä¡ăærŔæňąéČ¡åŐżæŢšäžčçăĄ

```
[deploy]
url = http://scrapyd.example.com/api/scrapyd
username = scrapy
password = secret
project = yourproject
```

è£Źæăůä¡ăářśçŻť'æŐěåŘŰæL'ğèąŇ

```
scrapyd-deploy
```

åęĆædIJä¡ăæIJL'åd'ŽäÿłtargetïijŇéĆčäźĹåŔŕäžěä¡£çŤÍäÿŃéİćåŚ¡äžd'ářĘprojectéČÍç¡šåĹřåd'Žäÿłtargetæ IJ■å

```
scrapyd-deploy -a -p <project>
```

9.2 éČÍç¡šåĹřScrapy Cloud

Scrapy CloudæŸŕäÿĂäÿłæL'Ÿçőąç ŽĎäžŚæIJ■åŁąåŹÍïijŇçŤśScrapyèČŇåŘŐçŽĎåĚňåŔýScrapinghubçżť'æŁ

åőČåĚ■éŹd'äžĘåőL'èčĚåŠŇçŻŚæŐğæIJ■åŁąåŹÍçŽĎéIJĂèęĄïijŇåźűæŔŘä¿ŻäžĘéİđåÿÿç¿ŐèğĆçŽĎUIæİěçĉ

ä¡ääŔŕäžěä¡£çŤĺshubåŠ¡äžď'èąŇåůěåĚůæİěèőšspideréČĺç¡šåĹřScrapy CloudãĂĆæŻť'åď'ŽèŕůåŔĆèĂČåőŸæŰźæŰĞæąč

Scrapy CloudåŠŇScrapydæŸŕåĚijåőźçŽĎïijŇä¡ääŔŕäžěæäźæ■őéIJĂèęĄåIJĺäÿď'èĂĚäźŃåL'■åĹĞæ■ćïijŇéĚ■ cfgïijŇèů§scrapyd-deployèŕżåŔŰçŽĎæŸŕäÿĂæăůçŽĎãĂĆ

10 ScrapyæŢŹçĺŃ10- åŁĺæĂĄéĚ■ç¡őçĹňèŹń

æIJL'å¿Ĺåď'ŽæŮűåĂŹæĹŚäżňéIJĂèęĄäżŐåď'Žäÿłç¡ŚçńŹçĹňåŔŰæL'ĂéIJĂèęĄçŽĎæŢřæ■őïijŇærŤåęĆæĹŚ åĚűåőđäÿ■éIJĂèęĄïijŇæĹŚäżňåŔŕäžěéĂŽè£Ğçżť'æŁď'äÿĂäÿłèğĎåĹŹéĚ■ç¡őèąĺæĹŰèĂĚäÿĂäÿłèğĎåĹŹéĚ

èęĄè£ŹæăůåĄŽïijŇæĹŚäżňåřśäÿ■èČ¡åĘ■ä¡£çŤĺåL'■éİćçŽĎscrapy crawl testè£Źçğ■åŠ¡äžď'äžĘïijŇæĹŚäżňéIJĂèęĄä¡£çŤĺçijŰçĺŃçŽĎæŰźåijŔè£ŘèąŇScrapy spiderïijŇåŔĆèĂČåőŸæŰźæŰĞæąč

10.1 èĎŽæIJňè£ŘèąŇScrapy

åŔŕäžěåĹľçŤĺscrapyæŔŘä¿ŻçŽĎæăÿå£ČAPIéĂŽè£ĞçijŰçĺŃæŰźåijŔåŘŕåŁĺscrapyïijŇäżčæŻ£äijäçż§çŽĎs crawlåŘŕåŁĺæŰźåijŔãĂĆ

ScrapyæđĎåżžäžŐTwistedåijĆæ■ěç¡ŚçżIJæąĘæđűå§žçąĂäźŃäÿŁïijŇåŻăæ■ď'ä¡äéIJĂèęĄåIJĺTwisted reactoréĞŇéİćè£ŘèąŇãĂĆ

éęŰåĚĹä¡ääŔŕäžěä¡£çŤĺscrapy.crawler.CrawlerProcessè£Źäÿłçśżæİěè£ŘèąŇä¡äçŽĎspiderïijŇè£ reactorïijŇåźűèČ¡éĚ■ç¡őä¡äçŽĎæŮěå£ŮåŠŇshutdownåď'ĎçŘĘåŹĺãĂĆæL'ĂæIJL'çŽĎscrapyåŠ¡äžď'éČ¡ä¡£ç

```
import scrapy
from scrapy.crawler import CrawlerProcess
from scrapy.utils.project import get_project_settings

process = CrawlerProcess(get_project_settings())

process.crawl(MySpider)
process.start() # the script will block here until the crawling is␣
↪finished
```

çĎűåŘŐä¡ääŕśåŔŕäžěçŻť'æŐěæL'ğèąŇè£ŹäÿłèĎŽæIJň

```
python run.py
```

åŔęåď'ŰäÿĂäÿłåŁ§èČ¡æŻť'åijžåď'ğçŽĎçśżæŸŕscrapy.crawler. CrawlerRunnerïijŇæŐĺè■Řä¡ää¡£çŤĺè£Źäÿł

```
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
```

10.2 åŘŇäÿÄè£ŻçÍŃè£ŘèąŇåď'Žäÿłspider

éżŸèőď'æČĚåĘţå¡Şä¡ăæŕŔæňąæL'ğèąŇscrapy crawlåŚ¡äžď'æŮűäijŽåĹŻåžžäÿĂäÿłæŰřçŽĎè£ŻçÍŃãĂĆ

```
import scrapy
from twisted.internet import reactor
from scrapy.crawler import CrawlerRunner
from scrapy.utils.log import configure_logging

class MySpider1(scrapy.Spider):
    # Your first spider definition
    ...

class MySpider2(scrapy.Spider):
    # Your second spider definition
    ...

configure_logging()
runner = CrawlerRunner()
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
```

10.3 åőŽäźL'èğĎåĹŹèąĺ

åě¡äžĘèÍĂå¡Šæ■čäijăïijŇæIJL'äžĘåL'■éİćçŽĎèĎŽæIJňåŘŕåŁĺå§žçąĂïijŇåřśåŔŕäžěåijĂåğŇæĹŚäžňçŽĎåŁĺæ
æĹŚäžňçŽĎéIJĂæśĆæŸŕè£ŹæăůçŽĎïijŇäžŐäÿď'äÿłäÿ■åŘŇçŽĎç¡ŚçńŹçĹňåŔŰæĹŚäžňæL'ĂéIJĂèęĄçŽĎæ

éęŰåĚĹæĹŚäžňéIJĂèęĄåőŽäźL'èğĎåĹŹèąĺåŠŇæŰĞçńăèąĺïijŇéĂŽè£ĞåŁĺæĂĄçŽĎåĹŻåžžèIJŸèŻŽçśžïijŇ

```python
#!/usr/bin/env python
# -*- encoding: utf-8 -*-
"""
Topic: åőŽäźL'æŢřæ■őåžŞæÍąåđŃåőđä¡Ş
Desc :
"""
import datetime

from sqlalchemy.engine.url import URL
from sqlalchemy.ext.declarative import declarative_base
from sqlalchemy import create_engine, Column, Integer, String, Text,
↪ DateTime
from coolscrapy.settings import DATABASE

Base = declarative_base()

class ArticleRule(Base):
    """èĞłåőŽäźL'æŰĞçńäçĹňåŔŰèğĎåĹŹ"""
    __tablename__ = 'article_rule'

    id = Column(Integer, primary_key=True)
    # èğĎåĹŹåŘ■çğř
    name = Column(String(30))
    # è£ŘèąŇçŽĎå$$åŘ■åĹŮèąÍïijŇéĂŮåŔůéŽŤåijĂ
    allow_domains = Column(String(100))
    # åijĂåğŃURLåĹŮèąÍïijŇéĂŮåŔůéŽŤåijĂ
    start_urls = Column(String(100))
    # äÿŃäÿĂéąţçŽĎxpath
    next_page = Column(String(100))
    # æŰĞçńäéŞ¿æŐëæ■čåĹŹèąÍè¿¿åijŔ(å■Řäÿš)
    allow_url = Column(String(200))
    # æŰĞçńäéŞ¿æŐëæŔŘåŔŰåŇžå$$xpath
    extract_from = Column(String(200))
    # æŰĞçńäæăĞéćŸxpath
    title_xpath = Column(String(100))
    # æŰĞçńääĘĚåőźxpath
    body_xpath = Column(Text)
    # åŔŚåÿČæŮűéÛt'xpath
    publish_time_xpath = Column(String(30))
    # æŰĞçńäæİěæžŘ
    source_site = Column(String(30))
    # èğĎåĹŹæŸŕåŘęçŤ$æŢĹ
    enable = Column(Integer)


class Article(Base):
    """æŰĞçńäçśż"""
    __tablename__ = 'articles'

    id = Column(Integer, primary_key=True)
```

```
    url = Column(String(100))
    title = Column(String(100))
    body = Column(Text)
    publish_time = Column(String(30))
    source_site = Column(String(30))
```

10.4 åőŽäźĽæŰĞçńălItem

è£Źäÿłå¿ĹçőÅå■ŢäžĘïijŇæšąäżĂäźĹéIJÄèęĄèŕť æŸŐçŽĎ

```
import scrapy


class Article(scrapy.Item):
    title = scrapy.Field()
    url = scrapy.Field()
    body = scrapy.Field()
    publish_time = scrapy.Field()
    source_site = scrapy.Field()
```

10.5 åőŽäźĽArticleSpider

æŐěäÿŇæİěæĹŚäžňářĘåőŽäźĽçĹňåŔŰæŰĞçńăçŽĎèIJŸèŻŻïijŇè£ŹäÿłspideräijŽä¡£çŤĺäÿĂäÿłRuleåőđä¿Ń

```
from coolscrapy.utils import parse_text
from scrapy.spiders import CrawlSpider, Rule
from scrapy.linkextractors import LinkExtractor
from coolscrapy.items import Article


class ArticleSpider(CrawlSpider):
    name = "article"

    def __init__(self, rule):
        self.rule = rule
        self.name = rule.name
        self.allowed_domains = rule.allow_domains.split(",")
        self.start_urls = rule.start_urls.split(",")
        rule_list = []
        # æůżåŁă`äÿŃäÿĂéąţ`çŽĎèğĎåĹŹ
        if rule.next_page:
            rule_list.append(Rule(LinkExtractor(restrict_
↪xpaths=rule.next_page)))
        # æůżåŁăæŁ¡åŔŰæŰĞçńăéŞ¿æŐěçŽĎèğĎåĹŹ
        rule_list.append(Rule(LinkExtractor(
            allow=[rule.allow_url],
            restrict_xpaths=[rule.extract_from]),
```

```
            callback='parse_item'))
        self.rules = tuple(rule_list)
        super(ArticleSpider, self).__init__()

    def parse_item(self, response):
        self.log('Hi, this is an article page! %s' % response.url)

        article = Article()
        article["url"] = response.url

        title = response.xpath(self.rule.title_xpath).extract()
        article["title"] = parse_text(title, self.rule.name, 'title
↪')

        body = response.xpath(self.rule.body_xpath).extract()
        article["body"] = parse_text(body, self.rule.name, 'body')

        publish_time = response.xpath(self.rule.publish_time_xpath).
↪extract()
        article["publish_time"] = parse_text(publish_time, self.
↪rule.name, 'publish_time')

        article["source_site"] = self.rule.source_site

        return article
```

èęĄæşÍæĎŔçŽĎæŸŕstart_urlsïijŇrulesç■L'éČ¡åĹİåğŃåŇŰæĹŘäžĘåŕźèśąçŽĎåśđæĂğïijŇéČ¡çŤśäijååĚěçŽĎ

10.6 çijŰåĘŹpipelineå■ŸåĆĺåĹřæŢřæ■őåžŞäÿ■

æĹŚäžňè£ŸæŸŕä¡£çŤĺSQLAlchemyæİěåřĘæŰĞçńăItemæŢřæ■őå■ŸåĆĺåĹřæŢřæ■őåžŞäÿ■

```
@contextmanager
def session_scope(Session):
    """Provide a transactional scope around a series of operations."
↪""
    session = Session()
    try:
        yield session
        session.commit()
    except:
        session.rollback()
        raise
    finally:
        session.close()


class ArticleDataBasePipeline(object):
    """ä£İå■ŸæŰĞçńăåĹřæŢřæ■őåžŞ"""
```

```
    def __init__(self):
        engine = db_connect()
        create_news_table(engine)
        self.Session = sessionmaker(bind=engine)

    def open_spider(self, spider):
        """This method is called when the spider is opened."""
        pass

    def process_item(self, item, spider):
        a = Article(url=item["url"],
                    title=item["title"].encode("utf-8"),
                    publish_time=item["publish_time"].encode("utf-8
↪"),
                    body=item["body"].encode("utf-8"),
                    source_site=item["source_site"].encode("utf-8"))
        with session_scope(self.Session) as session:
            session.add(a)

    def close_spider(self, spider):
        pass
```

10.7 ä£őæŤźrun.pyåŘŕåŁĺèĎŽæIJň

æĹŚäžňåřĘäÿŁéİćçŽĎrun.pyçĺ■ä¡IJä£őæŤźå■şåŔŕåőŽåĹűæĹŚäžňçŽĎæŰĞçńăçĹňèŹńåŘŕåŁĺèĎŽæIJň

```
import logging
from spiders.article_spider import ArticleSpider
from twisted.internet import reactor
from scrapy.crawler import CrawlerRunner
from scrapy.utils.project import get_project_settings
from scrapy.utils.log import configure_logging
from coolscrapy.models import db_connect
from coolscrapy.models import ArticleRule
from sqlalchemy.orm import sessionmaker

if __name__ == '__main__':
    settings = get_project_settings()
    configure_logging(settings)
    db = db_connect()
    Session = sessionmaker(bind=db)
    session = Session()
    rules = session.query(ArticleRule).filter(ArticleRule.enable ==␣
↪1).all()
    session.close()
    runner = CrawlerRunner(settings)

    for rule in rules:
```

```
        # stop reactor when spider closes
        # runner.signals.connect(spider_closing, signal=signals.
↪spider_closed)
        runner.crawl(ArticleSpider, rule=rule)

    d = runner.join()
    d.addBoth(lambda _: reactor.stop())

    # blocks process so always keep as the last statement
    reactor.run()
    logging.info('all finished.')
```

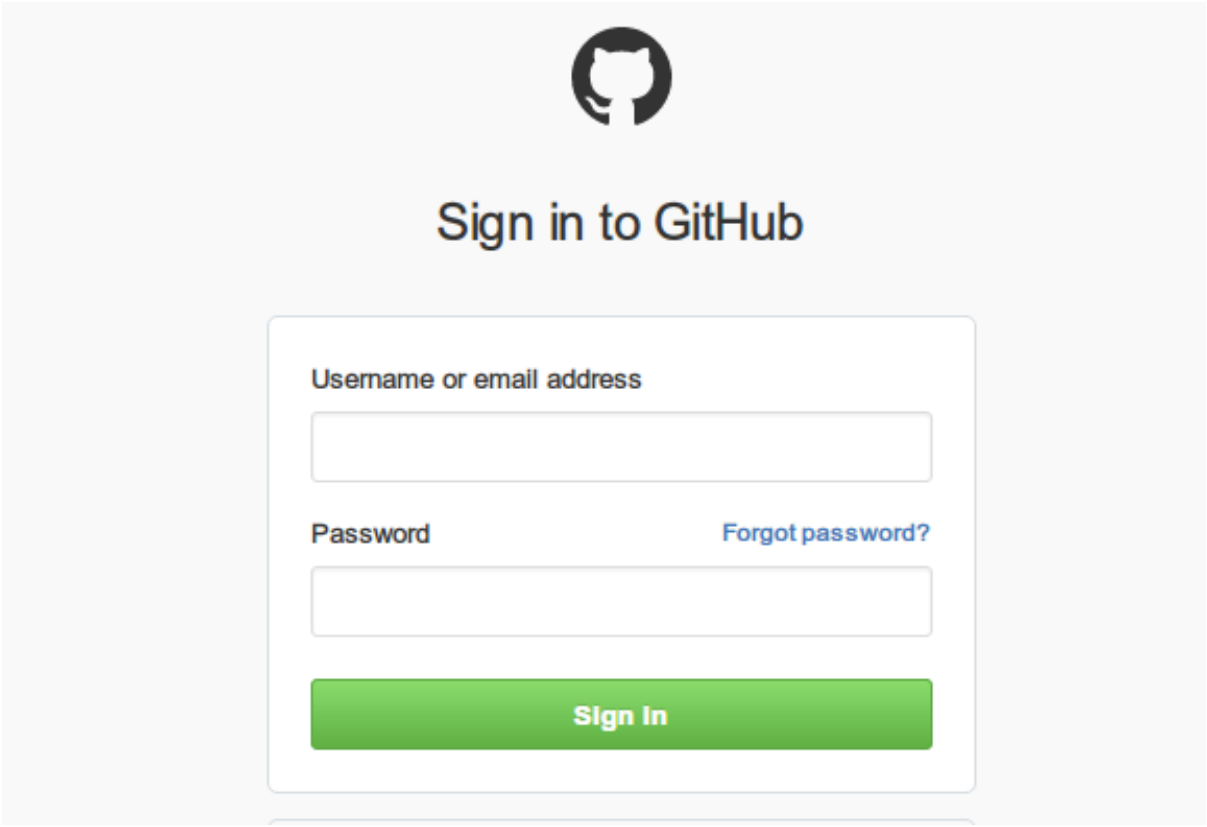
OKïijÑäÿĂåĹĞæŘđåőŽãĂĆçŐřåIJÍæĹŚäžñåŘŕäżěå¿ĂArticleRuleèąĺäÿ■åŁäåĚěæĹŘçŹ¿äÿŁå■Čäÿłç¡ŚçńŹçŽ
å¡ŞçĎűä¡ääőŇåĚÍåŘŕäżěåĄŽäÿĂäÿłWebåL'■çńŕæÍěåőŇæĹŘçżť'æŁd'ArticleRuleèąĺçŽĎäżżåŁąãĂĆå¡ŞçĎűA

ä¡ääŘŕäżěåIJÍGitHubäÿŁçIJŃåĹřæIJňæŰĞçŽĎåőŇæŢt'éąźçŻőæžŘçăĄãĂĆ

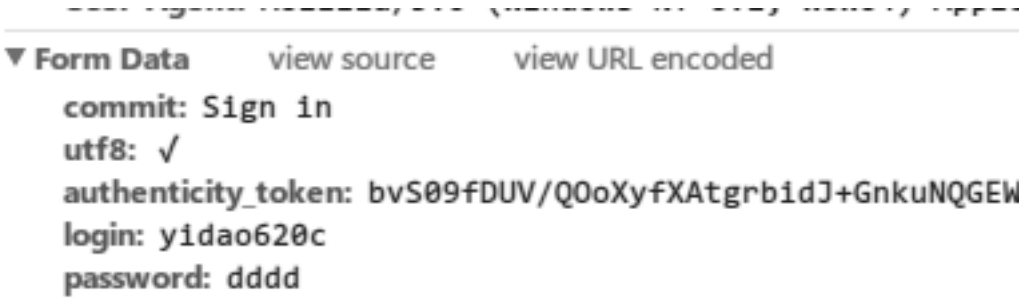
11 ScrapyæŢŹçÍŃ11- æĺąæŃ§çŹżå¡Ţ

æIJL'æŮűåĂŹçĹňåŘŰç¡ŚçńŹçŽĎæŮűåĂŹéIJĂèęĄçŹżå¡Ţïij ÑåIJÍScrapyäÿ■åŘŕäżěéĂŽè£ĞæĺąæŃ§çŹżå¡Ţä

èęĄæČşåőđçŐřçŹżå¡ŢåřšéIJĂèęĄèąĺå■ŢæŘŘäžď'ïijÑåĚĹéĂŽè£ĞætŔèğĹåŹÍèő£éŮőgithubçŽĎçŹżå¡Ţéątéİd
//github.com/loginïijÑ çĎűåŘŐä¡£çŤÍætŔèğĹåŹÍèřČèŕŢåůěåĚůæİěå¿ŮåĹřçŹżå¡ŢæŮűéIJĂèęĄæŘŘäžď'äżĂä



æĹŚè£ŹéĞÑä¡£çŤÍchromeætŔèğĹåŹÍçŽĎèřČèŕŢåůěåĚůïijÑF12æL'ŞåijĂåŘŐéĂL'æÑl'NetworkïijÑåźűårĘP
logåÑ¿äÿŁãĂĆ æĹŚæŢĚæĎŔè¿ŞåĚěéŤŹèŕŕçŽĎçŤÍæĹůåŘ■åŠÑårĘçăĄïijÑå¿ŮåĹřåőČæŘŘäžď'çŽĎformèąĹ



åŐžæ§ěçIJŃhtmlæžŘçăĄäijŽåŔŠçŐřèąĺå■ŢéĞŇéİćæIJL'äÿłéŽŘèŮŔçŽĎauthenticity_tokenåĂijïijŇ

11.1 éĞ■åĘŹstart_requestsæŰźæşŢ

èęĄä¡£çŤĺcookieïijŇçňňäÿĂæ■ěå¿ŮæL'ŞåijĂåőČåŚĂïijŇéżŸèőd'scrapyä¡£çŤĺCookiesMiddlewareäÿ■é

```
COOKIES_ENABLES = True
```

æĹŚäžňåĚĹèęĄæL'ŞåijĂçŹżå¡Ţéąţéİćïij ŇèŐůåŔŰauthenticity_tokenåĂijïijŇè£ŹéĞŇæĹŚéĞ■åĘŹä

```
# éĞ■åĘŹäžĘçĹňèŹńçśżçŽĎæŰźæşŢ, åőđçŐřäžĘèĞłåőŽäźL'èŕůæśĆ,␣
↪è£ŘèąŇæĹŘåŁŞåŘŐäijžèřČçŤĺcallbackåŻđèřČåĞ¡æŢř
def start_requests(self):
    return [Request("https://github.com/login",
                    meta={'cookiejar': 1}, callback=self.post_
↪login)]

# FormRequeset
def post_login(self, response):
    # åĚĹåŐżæŃ£éŽŘèŮŔçŽĎèąĺå■ŢåŔĆæŢřauthenticity_token
    authenticity_token = response.xpath(
        '//input[@name="authenticity_token"]/@value').extract_
↪first()
    logging.info('authenticity_token=' + authenticity_token)
    pass
```

start_requestsæŰźæşŢæŇĞåőŽäžĘåŻđèřČåĞ¡æŢřijŇçŤĺæİèèŐůåŔŰéŽŘèŮŔèąĺå■ŢåĂijauthenti

11.2 ä¡£çŤÍFormRequest

ScrapyäÿžæĹŚäžňåĞĘåď'ĞäžĘFormRequestçśžäÿŞéŮÍçŤÍæİěè£ŻèąŇFormèąĺå■ŢæŔŘäžď'çŽĎ

```
# äÿžäžĘæĺąæŃ$æţŕèğĹåŹĺïijŇæĹŚäżňåőŽäźL'httpheader
post_headers = {
    "Accept": "text/html,application/xhtml+xml,application/xml;q=0.
↪9,image/webp,*/*;q=0.8",
    "Accept-Encoding": "gzip, deflate",
    "Accept-Language": "zh-CN,zh;q=0.8,en;q=0.6",
    "Cache-Control": "no-cache",
    "Connection": "keep-alive",
    "Content-Type": "application/x-www-form-urlencoded",
    "User-Agent": "Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/
↪537.36 (KHTML, like Gecko) Chrome/49.0.2623.75 Safari/537.36",
    "Referer": "https://github.com/",
}
# ä¡£çŤÍFormRequesetæĺąæŃ$èąĺå■ŢæŔŘäžd'
def post_login(self, response):
    # åĚĹåŐżæŃ£éŽŘèŮŔçŽĎèąĺå■ŢåŔĆæŢřauthenticity_token
    authenticity_token = response.xpath(
        '//input[@name="authenticity_token"]/@value').extract_
↪first()
    logging.info('authenticity_token=' + authenticity_token)
    # FormRequeset.from_responseæŸŕŕScrapyæŔŘä¿ŽçŽĎäÿĂäÿłåĞ¡æŢř,␣
↪çŤÍäžŐpostèąĺå■Ţ
    # çŹżéŹĘæĹŘåŁ$åŘŐ, äijŽèřČçŤÍafter_
↪loginåŻđèřČåĞ¡æŢřïijŇåęĆædIJurlèů$Requestéąţéİćç
ŽĎäÿĂæăůåřšçIJĄçŢěæŐL'
    return [FormRequest.from_response(response,
                                      url='https://github.com/
↪session',
                                      meta={'cookiejar': response.
↪meta['cookiejar']},
                                      headers=self.post_headers,  #␣
↪æşĺæĎŔæ■d'åd'ĎçŽĎheaders
                                      formdata={
                                          'utf8': '✓',
                                          'login': 'yidao620c',
                                          'password': '******',
                                          'authenticity_token':␣
↪authenticity_token
                                      },
                                      callback=self.after_login,
                                      dont_filter=True
                                      )]

def after_login(self, response):
    pass
```

FormRequest.from_response()æŰźæşŢèőľä¡ăæŇĞåőŽæŔŘäžď'çŽĎurlïijŇèřůæśĆåď't'è£ŸæIJL'for

```
def after_login(self, response):
    # çŹżå¡ŢäźŃåŘŐïijŇåijÄåğŃè£ŻåĚěæĹŚèęĄçĹňåŔŰçŽĎçğĄä£ąéąţéİć
    for url in self.start_urls:
        #␣
↪åŻääÿžæĹŚäżñäÿŁéİćåőŽäźL’äžĘRuleïijŇæL’ÄäżěåŔłéIJÄèęĄçőÄå■ŢçŽĎçŤŞæĹŘåĹİåğŃçĹňå
        yield Request(url, meta={'cookiejar': response.meta[
↪'cookiejar']})
```

è£ŹéĞŇæĹŚéĂŽè£Ğstart_urlsåőŽäźL’äžĘåijÄåğŃéąţéİćïijŇçĎűåŘŐçŤ§æĹŘRequestïijŇåĚůä¡ŞçĹňåŔ

11.3 éĞ■åĘŹ_requests_to_follow

æIJL’äÿłéŮőécŸåĹŽåijÄåğŃåŻřæL’řæĹŚå¿ĹäźĚářśæŸŕè£ŹéĞŇæĹŚåőŽäźL’çŽĎspiderçżğæL’£èĞłCrawlSpi

```
def _requests_to_follow(self, response):
    """éĞ■åĘŹåŁäåĚěcookiejarçŽĎæŻt’æŰř"""
    if not isinstance(response, HtmlResponse):
        return
    seen = set()
    for n, rule in enumerate(self._rules):
        links = [l for l in rule.link_extractor.extract_
↪links(response) if l not in seen]
        if links and rule.process_links:
            links = rule.process_links(links)
        for link in links:
            seen.add(link)
            r = Request(url=link.url, callback=self._response_
↪downloaded)
            # äÿŃéİćè£ŹåŘěæŸŕæĹŚéĞ■åĘŹçŽĎ
            r.meta.update(rule=n, link_text=link.text,␣
↪cookiejar=response.meta['cookiejar'])
            yield rule.process_request(r)
```

11.4 éąţéİćåď’ĎçŘĘæŰźæşŢ

åIJĺèğĎåĹŹRuleéĞŇéİćæĹŚåőŽäźL’äžĘæŕŘäÿłéŞ¿æŐěçŽĎåŻđèřČåĞ¡æŢřparse_pageïijŇåřśæŸŕæIJĂçż

```
def parse_page(self, response):
    ""
↪"è£ŹäÿłæŸŕä¡£çŤĺLinkExtractorèĞłåŁĺåd’ĎçŘĘéŞ¿æŐëäżěåŘŁ`äÿŃäÿĂéąţ`"
↪""
    logging.info(u'--------------æűĹæĄŕåĹĘåL’šçž£-----------------')
    logging.info(response.url)
    issue_title = response.xpath(
        '//span[@class="js-issue-title"]/text()').extract_first()
    logging.info(u'issue_titleïijŽ' + issue_title.encode('utf-8'))
```

11.5 åőŇæŢťæžŘçăĄ

```
#!/usr/bin/env python
# -*- encoding: utf-8 -*-
"""
Topic: çŹžå¡ŢçĹňèŹń
Desc : æĺąæŃ§çŹžå¡Ţhttps://github.
↪comåŘŐåřĘèĞłåůśçŽĎissueåĚĺéČĺçĹňåĞžæİě
tipsïijŽä¡£çŤĺchromeèřČèŕŢpostèąĺå■ŢçŽĎæŮűåĂŹåŃ¿éĂL’Preserve␣
↪logåŠŇDisable cache
"""
import logging
import re
import sys
import scrapy
from scrapy.spiders import CrawlSpider, Rule
from scrapy.linkextractors import LinkExtractor
from scrapy.http import Request, FormRequest, HtmlResponse

logging.basicConfig(level=logging.INFO,
                    format='%(asctime)s %(filename)s[line:
↪%(lineno)d] %(levelname)s %(message)s',
                    datefmt='%Y-%m-%d %H:%M:%S',
                    handlers=[logging.StreamHandler(sys.stdout)])


class GithubSpider(CrawlSpider):
    name = "github"
    allowed_domains = ["github.com"]
    start_urls = [
        'https://github.com/issues',
    ]
    rules = (
        # æűĹæĄŕåĹŮèąĺ
        Rule(LinkExtractor(allow=('/issues/\d+',),
                           restrict_xpaths='//ul[starts-with(@class,
↪ "table-list")]/li/div[2]/a[2]'),
             callback='parse_page'),
        # äÿŇäÿĂéąţ, If callback is None follow defaults to True,␣
↪otherwise it defaults to False
        Rule(LinkExtractor(restrict_xpaths='//a[@class="next_page"]
↪')),
    )
    post_headers = {
        "Accept": "text/html,application/xhtml+xml,application/xml;
↪q=0.9,image/webp,*/*;q=0.8",
        "Accept-Encoding": "gzip, deflate",
        "Accept-Language": "zh-CN,zh;q=0.8,en;q=0.6",
        "Cache-Control": "no-cache",
        "Connection": "keep-alive",
```

```
        "Content-Type": "application/x-www-form-urlencoded",
        "User-Agent": "Mozilla/5.0 (Windows NT 6.1; WOW64)␣
↪AppleWebKit/537.36 (KHTML, like Gecko) Chrome/49.0.2623.75 Safari/
↪537.36",
        "Referer": "https://github.com/",
    }

    # éĞ■åĘźäžĘçĹňèŹńçśżçŽĎæŰźæşŢ, åőđçŐřäžĘèĞłåőŽäźL'èŕůæśĆ,␣
↪è£ŘèąŇæĹŘåŁŞåŘŐäijŽèřČçŤĺcallbackåŻđèřČåĞ¡æŢř
    def start_requests(self):
        return [Request("https://github.com/login",
                        meta={'cookiejar': 1}, callback=self.post_
↪login)]

    # FormRequeset
    def post_login(self, response):
        # åĚĹåŐżæŃ£éŽŘèŮŔçŽĎèąĺå■ŢåŔĆæŢřauthenticity_token
        authenticity_token = response.xpath(
            '//input[@name="authenticity_token"]/@value').extract_
↪first()
        logging.info('authenticity_token=' + authenticity_token)
        # FormRequeset.from_responseæŸŕScrapyæŔŘä¿ŻçŽĎäÿĂäÿłåĞ¡æŢř,␣
↪çŤĺäžŐpostèąĺå■Ţ
        # çŹżéŹĘæĹŘåŁŞåŘŐ, äijŽèřČçŤĺafter_
↪loginåŻđèřČåĞ¡æŢřïijŇåęĆæđIJurlèů$Requestéąţéİćç ŽĎäÿĂæăůåřšçIJĄçŢěæŐL'
        return [FormRequest.from_response(response,
                                          url='https://github.com/
↪session',
                                          meta={'cookiejar':␣
↪response.meta['cookiejar']},
                                          headers=self.post_headers,
↪  # æşĺæĎŔæ■d'åd'ĎçŽĎheaders
                                          formdata={
                                              'utf8': '✓',
                                              'login': 'yidao620c',
                                              'password': '******',
                                              'authenticity_token':␣
↪authenticity_token
                                          },
                                          callback=self.after_login,
                                          dont_filter=True
                                          )]

    def after_login(self, response):
        for url in self.start_urls:
            #␣
↪åŻääÿžæĹŚäżñäÿŁéİćåőŽäźL'äžĘRuleïijŇæL'ĂäżěåŔłéIJĂèęĄçőĂå■ŢçŽĎçŤ$æĹŘåĹİåğŃçĹňå
            yield Request(url, meta={'cookiejar': response.meta[
↪'cookiejar']})
```

```
    def parse_page(self, response):
        ""
↪"è£ŹäÿłæŸŕä¡£çŤĺLinkExtractorèĞłåŁĺåd'ĎçŘĘéŞ¿æŐěäżěåŔŁ`äÿŃäÿĂéąţ`"
↪""
        logging.info(u'--------------æűĹæĄŕåĹĘåL'šçž£---------------
↪--')
        logging.info(response.url)
        issue_title = response.xpath(
            '//span[@class="js-issue-title"]/text()').extract_
↪first()
        logging.info(u'issue_titleïijŽ' + issue_title.encode('utf-8
↪'))

    def _requests_to_follow(self, response):
        """éĞ■åĘŹåŁäåĚěcookiejarçŽĎæŻt'æŰř"""
        if not isinstance(response, HtmlResponse):
            return
        seen = set()
        for n, rule in enumerate(self._rules):
            links = [l for l in rule.link_extractor.extract_
↪links(response) if l not in seen]
            if links and rule.process_links:
                links = rule.process_links(links)
            for link in links:
                seen.add(link)
                r = Request(url=link.url, callback=self._response_
↪downloaded)
                # äÿŃéİćè£ŹåŘěæŸŕæĹŚéĞ■åĘŹçŽĎ
                r.meta.update(rule=n, link_text=link.text,␣
↪cookiejar=response.meta['cookiejar'])
                yield rule.process_request(r)
```

ä¡ääŔŕäžěåIJĺGitHubäÿŁçIJŃåĹřæIJňæŰĞçŽĎåőŇæŢť'éąźçŽőæžŘçăĄïijŇè£ŸæIJL'åŘęåd'ŰäÿĂäÿłèĞłåŁĺç

12 ScrapyæŢŹçĺŃ12- æŁŞåŔŰåŁĺæĂĄç¡Śçńź

åL'■éĺćæĹŚäżñäżŃçż■çŽĎéČ¡æŸŕåŐżæŁŞåŔŰéİŹæĂĄçŽĎç¡ŚçńŹéąţéİćïijŇäź§åřśæŸŕèŕť'æĹŚäżňæL'Şåij
ä¡ĘæŸŕåęĆäżŁçŽĎäžŠèĄŢç¡Śåd'ğéČĺåĹĘçŽĎwebéąţéİćéČ¡æŸŕåŁĺæĂĄçŽĎïijŇçżŔåÿÿéĂŻçŽĎç¡ŚçńŹä¿Ń
äÿŃè¡¡æ§ŘäÿłéŞ¿æŐěå¿ŮåĹřçŽĎéąţéİćéĞŇéİćåŘńæIJL'åijĆæ■ěåŁäè¡¡çŽĎåĘĚåőźïijŇè£ŹæăůåĘ■ä¡£çŤĺäźĺ

ä¡£çŤĺJavascriptæÿšæ§ŞåŠŇåd'ĎçŘĘç¡ŚéąţæŸŕçğ■éĺđåÿÿåÿÿèğĄçŽĎåĄŻæşŢïijŇåęĆä¡Ţåd'ĎçŘĘäÿĂäÿłåd
è£ŹćŕĞæŰĞçńăåřĘèŕť'æŸŐåęĆä¡ŢåIJĺScrapyçĹňèŹńäÿ■ä¡£çŤĺscrapy-
splashæĺěåd'ĎçŘĘéąţéİćäÿ■å¿ŮJavascriptãĂĆ

12.1 scrapy-splashçőÄäżŃ

scrapy-splashåĹľçŤÍSplashåřĘjavascriptåŠŇScrapyéŻĘæĹŘèţůæİěïijŇä¡£å¿ŮScrapyåŔŕäžěæŁŞåŔŰåŁĺæĂA

SplashæŸŕäÿĂäÿłjavascriptæÿšæ§ŞæIJ■åŁąïijŇæŸŕåőđçŐřäžĘHTTP APIçŽĎè¡žéĞŔçžğæţŔèğĹåŹĺïijŇåžŢåśĆå§žäžŐTwistedåŠŇQTæąĘæđűïijŇPythonèŕ■èÍĂçijŰåĘŹãĂĆæL'Ă

12.2 åőL'èčĚdocker

åőŸç¡Śåżžèőöä¡£çŤÍdockeråőźåŹĺåőL'èčĚæŰźåijŔSplashãĂĆéĆčäźĹéęŰåĚÍä¡ăå¿ŮåĚÍåőL'èčĚdocker

åŔĆèĂČåőŸæŰźåőL'èčĚæŰĞæąčïijŇè£ŹéĞŇæĹŚéĂL'æŃl' Ubuntu 12.04 LT-SçL'ĹæIJňåőL'èčĚ

å■ĞçžğåĘĚæäÿçL'ĹæIJňïijŇdockeréIJĂèęĄ3.13åĘĚæäÿ

```
$ sudo apt-get update
$ sudo apt-get install linux-image-generic-lts-trusty
$ sudo reboot
```

åőL'èčĚCAèőd'èŕĄ

```
$ sudo apt-get install apt-transport-https ca-certificates
```

åćđåŁăæŰřçŽĎGPGkey

```
$ sudo apt-key adv --keyserver hkp://p80.pool.sks-keyservers.net:80␣
↪--recv-keys 58118E89F3A912897C070ADBF76221572C52609D
```

æL'ŞåijĂ/etc/apt/sources.list.d/docker.listïijŇåęĆæđIJæšąæIJL'åřśåĹŻåžžäÿĂäÿłïijŇçĎűå

```
deb https://apt.dockerproject.org/repo ubuntu-precise main
```

æŻt'æŰřAPT

```
$ sudo apt-get update
$ sudo apt-get purge lxc-docker
$ apt-cache policy docker-engine
```

åőL'èčĚ

```
$ sudo apt-get install docker-engine
```

åŘŕåŁĺdockeræIJ■åŁą

```
$ sudo service docker start
```

élŇèŕĄæŸŕåŘęåŘŕåŁĺæĹŘåŁ§

```
$ sudo docker run hello-world
```

äÿŁéÍćè£ŹæÍąåŚ¡äžďäijŽäÿŃè¡¡äÿĂäÿłæţŃèŕŢéŢIJåČŔåźűåIJÍåőźåŹÍäÿ■è£ŘèąŇåőČïijŇåőČäijŽæĽŞå■řäÿĂ

12.3 åőĽèčĚSplash

æŃĽåŔŰéŢIJåČŔäÿŃæÍě

```
$ sudo docker pull scrapinghub/splash
```

åŘŕåŁÍåőźåŹÍ

```
$ sudo docker run -p 5023:5023 -p 8050:8050 -p 8051:8051␣
↪scrapinghub/splash
```

çŐřåIJÍåŔŕäžěéĂŽè£Ğ0.0.0.0:8050(http),8051(https),5023 (telnet)æÍèèő£éŮőSplashäžĘãĂĆ

12.4 åőĽèčĚscrapy-splash

ä¡£çŤÍpipåőĽèčĚ

```
$ pip install scrapy-splash
```

12.5 éĚ■ç¡őscrapy-splash

åIJÍä¡ăçŽĎscrapyåůěçÍŃçŽĎéĚ■ç¡őæŰĞäžűsettings.pyäÿ■æůżåŁă

```
SPLASH_URL = 'http://192.168.203.92:8050'
```

æůżåŁăSplashäÿ■éŮťäžűïijŇè£ŸæŸŕåIJÍsettings.pyäÿ■éĂŽè£ĞDOWNLOADER_MIDDLEWARESæŇĞå

```
DOWNLOADER_MIDDLEWARES = {
    'scrapy_splash.SplashCookiesMiddleware': 723,
    'scrapy_splash.SplashMiddleware': 725,
    'scrapy.downloadermiddlewares.httpcompression.
↪HttpCompressionMiddleware': 810,
}
```

éżŸèőď'æČĚåĘţäÿŃïijŇHttpProxyMiddlewareçŽĎäijŸåĚĹçžğæŸŕ750ïijŇèęĄæŁŁåőČæŤ¿åIJÍSplashäÿ■éŮt

èő¿ç¡őSplashèĞłåůśçŽĎåŐżéĞ■è£Ğæžď'åŹÍ

```
DUPEFILTER_CLASS = 'scrapy_splash.SplashAwareDupeFilter'
```

åęĆæđIJä¡ää¡£çŤÍSplashçŽĎHttpçijŞå■ŸïijŇéĆčäźĹè£ŸèęĄæŇĞåőŽäÿĂäÿłèĞłåőŽäźĽçŽĎçijŞå■ŸåŘŐåŕ
splashæŔŘä¿ŻäžĘäÿĂäÿłscrapy.contrib.httpcache.
FilesystemCacheStorageçŽĎå■Řçśż

```
HTTPCACHE_STORAGE = 'scrapy_splash.SplashAwareFSCacheStorage'
```

åęĆæđIJä¡ãèęĄä¡£çŤĺåĔűäżŰçŽĎçijŞå■Ÿå■ŸåĆĺïijŇéĆčäźĹéIJĂèęĄçżğæĽ£è£ŹäÿłçśżåźűäÿŤåřĘæĽĂæIJL
`util.request.request_fingerprint`èřČçŤĺæŽ£æ■ćæĹŘ`scrapy_splash.splash_request_fingerprint`

12.6 ä¡£çŤĺscrapy-splash

SplashRequest

æIJĂçőĂå■ŢçŽĎæÿšæ§ŞèŕůæśĆçŽĎæŰźåijŔæŸŕä¡£çŤĺ`scrapy_splash.SplashRequest`ïijŇéĂŽåÿÿä¡ãåžŤèŕéĕĂĽæŃľä¡£çŤĺè£Źäÿł

```
yield SplashRequest(url, self.parse_result,
    args={
        # optional; parameters passed to Splash HTTP API
        'wait': 0.5,

        # 'url' is prefilled from request url
        # 'http_method' is set to 'POST' for POST requests
        # 'body' is set to request body for POST requests
    },
    endpoint='render.json', # optional; default is render.html
    splash_url='<url>',     # optional; overrides SPLASH_URL
    slot_policy=scrapy_splash.SlotPolicy.PER_DOMAIN,  # optional
)
```

åŔęåď ŰïijŇä¡ãè£ŸåŔŕäžĕåIJĺæŹőéĂŽçŽĎscrapyèŕůæśĆäÿ■äijäéĂŠ`splash`èŕůæśĆmetaåĚşéŤőå■Ůè¿¿åĹř

```
yield scrapy.Request(url, self.parse_result, meta={
    'splash': {
        'args': {
            # set rendering arguments here
            'html': 1,
            'png': 1,

            # 'url' is prefilled from request url
            # 'http_method' is set to 'POST' for POST requests
            # 'body' is set to request body for POST requests
        },

        # optional parameters
        'endpoint': 'render.json',  # optional; default is render.
↪json
        'splash_url': '<url>',      # optional; overrides SPLASH_URL
        'slot_policy': scrapy_splash.SlotPolicy.PER_DOMAIN,
        'splash_headers': {},       # optional; a dict with headers␣
↪sent to Splash
        'dont_process_response': True, # optional, default is False
```

```
        'dont_send_headers': True,  # optional, default is False
        'magic_response': False,    # optional, default is True
    }
})
```

Splash APIèŕťæŸŐïijŇä¡£çŤÍSplashRequestæŸŕäÿĂäÿłéÍđåÿÿä¿£åĹ'çŽĎåůěåĚůæÍěåąńåĚĚrequest.meta['splash']éĞŇçŽĎæŢřæ■ő

- meta[‘splash’][‘args’] åŇĚåŘńäžĘåŔŚå¿ĂSplashçŽĎåŔĆæŢřãĂĆ
- meta[‘splash’][‘endpoint’] æŇĞåőŽäžĘSplashæĽĂä¡£çŤÍçŽĎendpointïijŇézŸèőď'æŸŕrender.html
- meta[‘splash’][‘splash_url’] èęĘçŽŰäžĘsettings.pyæŰĞäżűäÿ■éĚ■ç¡őçŽĎSplash URL
- meta[‘splash’][‘splash_headers’] è£ŘèąŇä¡ăåćđåŁăæĹŰä£őæŤźåŔŚå¿ĂSplashæIJ■åŁąåŹÍçŽĎHTTPå
- meta[‘splash’][‘dont_send_headers’] åęĆæđIJä¡ăäÿ■æČşäijăéĂŠheadersçżŹSplashïijŇářĘåőČèő¿ç¡őæÍ
- meta[‘splash’][‘slot_policy’] èőĺ'ä¡äèĞłåőŽäźĽ'SplashèŕůæśĆçŽĎåŘŇæ■ěèő¿ç¡ő
- meta[‘splash’][‘dont_process_response’] å¡Şä¡äèő¿ç¡őæĹŘTrueåŘŐïijŇSplashMiddlewareäÿ■äijŽResponseèŕůæśĆãĂĆéżŸèőď'æŸŕäijŽè£ŤåŻđSplashResponseå■ŘçśżåŞ■åžŤærŤåęĆSplashTe
- meta[‘splash’][‘magic_response’] éżŸèőď'äÿžTrueïijŇS-plashäijŽèĞłåŁÍèő¿ç¡őResponseçŽĎäÿĂäžŽåśđæĂğïijŇærŤåęĆresponse.headers,response.bodyç■Ľ'

åęĆæđIJä¡äæČşéĂŽè£ĞSplashæÍěæŔŘäžd'FormèŕůæśĆïijŇåŕŕäžěä¡£çŤÍscrapy_splash.SplashFormRequestïijŇåőČèů§SplashRequestä¡£çŤÍæŸŕäÿĂæăůçŽĎãĂĆ

Responses

åŕźäžŐäÿ■åŘŇçŽĎSplashèŕůæśĆïijŇscrapy-splashè£ŤåŻđäÿ■åŘŇçŽĎResponseå■Řçśżżŕ

- SplashResponse äžŇè£ŻåĹűåŞ■åžŤïijŇærŤåęĆåŕź/render.pngçŽĎåŞ■åžŤ
- SplashTextResponse æŰĞæIJňåŞ■åžŤïijŇærŤåęĆåŕź/render.htmlçŽĎåŞ■åžŤ
- SplashJsonResponse JSONåŞ■åžŤïijŇærŤåęĆåŕź/render.jsonæĹŰä¡£çŤÍLuaèĎŽæIJňçŽĎ/executeçŽĎ

åęĆæđIJä¡ääŔłæČşä¡£çŤÍæăĞåĞEçŽĎResponseåŕźèśąïijŇářśèő¿ç¡őmeta['splash']['dont_proces

æĽĂæIJĽè£ŹäžŻResponseäijŽæŁŁresponse.urlèő¿ç¡őæĹŘåŐ§ågŃèŕůæśĆURL(äź§ářśæŸŕä¡äèęĄæÿš endpointçŽĎURLåIJřåÌĂãĂĆåőđéŹĚåIJřåÌĂéĂŽè£Ğresponse.real_urlå¿ŮåĹř

SessionçŽĎåď'ĎçŘĘ

SplashæIJňèžńæŸŕæŮăçŁűæĂĄçŽĎïijŇéĆčäźĹäÿžäžĘæŤŕæŇĄscrapy-splashçŽĎsessionå£ĚéążçijŭåĘŹLuaèĎŽæIJňïijŇä¡£çŤÍ/execute

```
function main(splash)
    splash:init_cookies(splash.args.cookies)

    -- ... your script

    return {
        cookies = splash:get_cookies(),
        -- ... other results, e.g. html
    }
end
```

èĂŇæăĞåĞĘçŽĎscrapy sessionåŔĆæŢřåŔŕäžěä¡£çŤĺSplashRequeståřĘcookieæůżåŁäåĹřå¡ŞåĽ■Splash cookiejaräÿ■

12.7 ä¡£çŤĺåőđä¿Ń

æŐěäÿŇæĬěæĹŚéĂŽè£ĞäÿĂäÿłåőđéŹĚçŽĎä¿Ňå■ŘæĬěæijŤçd'žæĂŐæăůä¡£çŤĺïijŇæĹŚéĂL'æŃl'çĹňåŔŰäž

äžňäÿIJç¡ŚæL'Şåij Äé ęŰéąţçŽĎæŮűåĂŹåŔłäijŽåřĘåŕijèĹłèŔIJå■ŢåŁäè¡¡åĞžæĬěïijŇåĚűäżŮåĚůä¡ŞéęŰéąţåĘ æĹŚçŐřåIJĺåřśéĂŽè£ĞçĹňåŔŰè£Źäÿŀ"çŇIJä¡ăåŰIJæňć"è£ŹåŻŻäÿłå■ŮæĬěèŕť æŸŐäÿŇæŹőéĂŽçŽĎScrapy

éęŰåĚĹæĹŚäžňåĘŹäÿłçőĂå■ŢçŽĎæţŃèŕŢSpiderïijŇäÿ■ä¡£çŤĺsplashïijŽ

```
class TestSpider(scrapy.Spider):
    name = "test"
    allowed_domains = ["jd.com"]
    start_urls = [
        "http://www.jd.com/"
    ]

    def parse(self, response):
        logging.info(u'---------
↪æĹŚè£ŹäÿłæŷŕçőĂå■ŢçŽĎçŻt'æŐěèŐůåŔŰäžňäÿIJç¡Śéę Ůéąţæţ ŃèŕŢ---------')
        guessyou = response.xpath('//div[@id="guessyou"]/div[1]/h2/
↪text()').extract_first()
        logging.info(u"findïijŽ%s" % guessyou)
        logging.info(u'---------------success----------------')
```

çĎűåŘŐè£ŘèąŇçżŞædIJïijŽ

```
2016-04-18 14:42:44 test_spider.py[line:20] INFO ---------
↪æĹŚè£ŹäÿłæŷŕçőĂå■ŢçŽĎçŻt'æŐěèŐůåŔŰäžňäÿIJç¡Śéę Ůéąţæţ ŃèŕŢ---------
2016-04-18 14:42:44 test_spider.py[line:22] INFO findïijŽNone
2016-04-18 14:42:44 test_spider.py[line:23] INFO ---------------
↪success----------------
```

æĹŚæL'¿äÿ■åĹřéĆčäÿł"çŇIJä¡ăåŰIJæňć"è£ŹåŻŻäÿłå■Ů

æŐěäÿŇæĬěæĹŚä¡£çŤĺsplashæĬěçĹňåŔŰ

```
import scrapy
from scrapy_splash import SplashRequest


class JsSpider(scrapy.Spider):
    name = "jd"
    allowed_domains = ["jd.com"]
    start_urls = [
        "http://www.jd.com/"
    ]

    def start_requests(self):
        splash_args = {
            'wait': 0.5,
        }
        for url in self.start_urls:
            yield SplashRequest(url, self.parse_result, endpoint=
↪'render.html',
                                args=splash_args)

    def parse_result(self, response):
        logging.info(u'----------
↪ä¡£çŤÍsplashçĹňåŔŰäžňäÿIJç¡Śéę ŮéąţåijĆæ■ěåŁăè¡¡åĘĚåőź----------')
        guessyou = response.xpath('//div[@id="guessyou"]/div[1]/h2/
↪text()').extract_first()
        logging.info(u"findïijŽ%s" % guessyou)
        logging.info(u'---------------success----------------')
```

è£ŘèąŇçżŞæđIJïijŽ

```
2016-04-18 14:42:51 js_spider.py[line:36] INFO ----------
↪ä¡£çŤÍsplashçĹňåŔŰäžňäÿIJç¡Śéę ŮéąţåijĆæ■ěåŁăè¡¡åĘĚåőź----------
2016-04-18 14:42:51 js_spider.py[line:38] INFO␣
↪findïijŽçŇIJä¡ăåŰIJæňć
2016-04-18 14:42:51 js_spider.py[line:39] INFO ---------------
↪success----------------
```

åŔŕäżěçIJŇåĞžçżŞæđIJéĞŇéĺćåůšçżŔæL’¿åĹřäžĘè£Źäÿł”çŇIJä¡ăåŰIJæňć”ïijŇèŕť æŸŐåijĆæ■ěåŁăè¡¡åĘĚåő

13 èĄŤçşżæĹŚ

- EmailïijŽ yidao620@gmail.com
- å■Žåőćïij Ž https://www.xncoding.com/
- GitHubïijŽ https://github.com/yidao620c

å¿ő㣤æL’ńäÿĂæL’ńïijŇèŕůå■ŽäÿżåŰĺæĺŕåŠŰåŢą

掃描上面的QR Code，加我WeChat。