Scrapy Library

Patrick J. O'Brien - @obdit
PhillyPUG - Monetate
09-13-2011

What is it?

- Screen scraper / web crawler
- 100% Python
- Asynchronous
- Event Driven
- Well tested

Featured Services

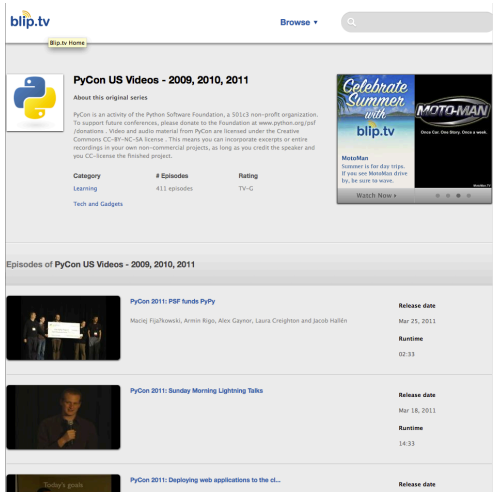
- Logging: critical, error, warning, info, debug
- Stats: key/value implementation
- Email Notification
- Monitor: telnet, web

Plan of Attack

- Pick a site
- Create spider
- Target the data
- Extract the data
- Assist the spider
- Launch spider

Pick a site

- PyCon US videos from blip.tv
  - http://blip.tv/pycon-us-videos-2009-2010-2011



monetate

Create Project

- `scrapy startproject pyconvideos`

```
foo - 15:02  $ ls -R
pyconvideos

./pyconvideos:
pyconvideos     scrapy.cfg

./pyconvideos/pyconvideos:
__init__.py     items.py        pipelines.py    settings.py     spiders

./pyconvideos/pyconvideos/spiders:
__init__.py
```

Target the Data

- Extract structured data
- Scrap items are dictionary-like

```python
# Module for PyCon Video items
from scrapy.item import Item, Field

class PyconvideosItem(Item):
    title = Field()
    description = Field()
    date = Field()
    runtime = Field()
    url = Field()
    pass
```

Extract the Data

- Scrapy uses XPath selectors to select data
- Other options

```
from lxml.cssselect import \
CSSSelector
```

```
<ul class="EpisodeList">
  <li class="clearfix">
  <li class="clearfix">
  <li class="clearfix">
  <li class="clearfix">
  <li class="clearfix">
  <li class="clearfix">
  <li class="clearfix">
  <li class="clearfix">
  <li class="clearfix">
  <li class="clearfix">
</ul>
```

```
>>> episode_sel = CSSSelector('ul.EpisodeList li.clearfix')
>>> episode_sel(lx)
[<Element li at 9d498fc>, <Element li at 9d4989c>, <Element li a
t 9d498cc>, <Element li at 9d4995c>, <Element li at 9d4992c>, <E
lement li at 9d4986c>, <Element li at 9d4998c>, <Element li at 9
d499bc>, <Element li at 9d499ec>, <Element li at 9d49a1c>]
>>> len(episode_sel(lx))
10
```

monetate

More selection

- Inspect markup and decide best approach

```
title_sel = CSSSelector('#TheaterLite h2')
link_sel = CSSSelector('div.Description h3 a')
desc_sel = CSSSelector('div.About p')
info_sel  = CSSSelector('ul.MetaDataPairs li h6')
```

monetate

Assist the Spider

- Which urls to visit
- Filter visited urls
- Create HTTP Requests
  - url, method, body, header, cookies, encoding, priority, dont_filter, callback, errback
- Handle HTTP Responses
  - url, headers, status, body, meta, flags

Assist the Spider

- Go to video details
- DO IT!

```
def parse(self, response):
    lx = lxml.html.fromstring(response.body_as_unicode())
    episodes = episode_sel(lx)
    for episode in episodes:
        url = link_sel(episode)[0]
        url = urljoin_rfc(self.start_urls[0], url.attrib['href'])
        yield Request(url=url, callback=self.parse_video_page)
```

monetate

Assist the Spider

- Store the items

```
def parse_video_page(self, response):
    """ Collect video information """
    lx = lxml.html.fromstring(response.body_as_unicode())
    title = title_sel(lx)[0].text.strip()
    desc = desc_sel(lx)[0].text.strip()
    date, runtime = [info.text for info in info_sel(lx)]

    video = PyconvideosItem()
    video['title'] = title
    video['description'] = desc
    video['date'] = date
    video['runtime'] = runtime
    video['url'] = response.url

    return video
```

Assist the Spider

- Pagination: better take a closer look
  - XHR
  - Are we finished?

Assist the Spider

- OK, let's paginate

```
# Simulate pagination
if episodes:
    current = url_query_parameter(response.url, 'page')
    if not current:
        current = '2' # XHR request starts at page 2
    url = "http://blip.tv/pr/show_get_full_episode_list?"
    url += "users_id=348873&lite=1&esi=1&page=%s"
    url = url % str(int(current)+ 1)

    yield Request(url=url, callback=self.parse)
```

Launch Spider

- Review the code

```
scrapy crawl pycon --set FEED_URI=pyconvideos.json\
--set FEED_FORMAT=json
```

- Enjoy the winning

Thanks!

- Tonight's talk available https://github.com/pjob/pyconscrape