SEP	4
Title	Library-Like API for quick scraping
Author	Pablo Hoffman
Created	2009-07-21
Status	Archived

SEP-004: Library API

> **Note**
>
> the library API has been implemented, but slightly different from proposed in this SEP. You can run a Scrapy crawler inside a Twisted reactor, but not outside it.

Introduction

It would be desirable for Scrapy to provide a quick, "light-weight" mechanism for implementing crawlers by just using callback functions. That way you could use Scrapy as any standard library (like you would use os.walk) in a script without the overhead of having to create an entire project from scratch.

Proposed API

Here's a simple proof-of-concept code of such script:

```
#!/usr/bin/env python
from scrapy.http import Request
from scrapy import Crawler

# a container to hold scraped items
scraped_items = []

def parse_start_page(response):
    # collect urls to follow into urls_to_follow list
    requests = [Request(url, callback=parse_other_page) for url in urls_to_follow]
    return requests

def parse_other_page(response):
    # ... parse items from response content ...
    scraped_items.extend(parsed_items)

start_urls = ["http://www.example.com/start_page.html"]

cr = Crawler(start_urls, callback=parse_start_page)
cr.run() # blocking call - this populates scraped_items

print "%d items scraped" % len(scraped_items)
# ... do something more interesting with scraped_items ...
```

The behaviour of the Scrapy crawler would be controller by the Scrapy settings, naturally, just like any typical Scrapy project. But the default settings should be sufficient so as to not require adding any specific setting. But, at the same time, you could do it if you need to, say, for specifying a custom middleware.

It shouldn't be hard to implement this API as all this functionality is a (small) subset of the current Scrapy functionality. At the same time, it would provide an additional incentive for newcomers.

Crawler class

The Crawler class would have the following instance arguments (most of them have been singletons so far):

- engine
- settings
- spiders
- extensions

Spider Manager

The role of the spider manager will be to "resolve" spiders from URLs and domains. Also, it should be moved outside scrapy.spider (and only BaseSpider left there).

There is also the `close_spider()` method which is called for all closed spiders, even when they weren't resolved first by the spider manager. We need to decide what to do with this method.