Link Extractors

Link extractors are objects whose only purpose is to extract links from web pages (`scrapy.http.Response` objects) which will be eventually followed.

There is `scrapy.linkextractors.LinkExtractor` available in Scrapy, but you can create your own custom Link Extractors to suit your needs by implementing a simple interface.

The only public method that every link extractor has is `extract_links`, which receives a `Response` object and returns a list of `scrapy.link.Link` objects. Link extractors are meant to be instantiated once and their `extract_links` method called several times with different responses to extract links to follow.

Link extractors are used in the `CrawlSpider` class (available in Scrapy), through a set of rules, but you can also use it in your spiders, even if you don't subclass from `CrawlSpider`, as its purpose is very simple: to extract links.

Built-in link extractors reference

Link extractors classes bundled with Scrapy are provided in the `scrapy.linkextractors` module.

The default link extractor is `LinkExtractor`, which is the same as `LxmlLinkExtractor`:

```
from scrapy.linkextractors import LinkExtractor
```

There used to be other link extractor classes in previous Scrapy versions, but they are deprecated now.

LxmlLinkExtractor

class `scrapy.linkextractors.lxmlhtml.LxmlLinkExtractor`(allow=(), deny=(), allow_domains=(), deny_domains=(), deny_extensions=None, restrict_xpaths=(), restrict_css=(), tags=('a', 'area'), attrs=('href',), canonicalize=False, unique=True, process_value=None, strip=True)

LxmlLinkExtractor is the recommended link extractor with handy filtering options. It is implemented using lxml's robust HTMLParser.

Parameters:

- **allow** (a regular expression (or list of)) – a single regular expression (or list of regular expressions) that the (absolute) urls must match in order to be extracted. If not given (or empty), it will match all links.
- **deny** (a regular expression (or list of)) – a single regular expression (or list of regular expressions) that the (absolute) urls must match in order to be excluded (ie. not extracted). It has precedence over the `allow` parameter. If not given (or empty) it won't exclude any links.
- **allow_domains** (str or list) – a single value or a list of string containing domains which will be considered for extracting the links
- **deny_domains** (str or list) – a single value or a list of strings containing domains which won't be considered for extracting the links
- **deny_extensions** (list) – a single value or list of strings containing extensions that should be ignored when extracting links. If not given, it will default to the `IGNORED_EXTENSIONS` list defined in the scrapy.linkextractors package.
- **restrict_xpaths** (str or list) – is an XPath (or list of XPath's) which defines regions inside the response where links should be extracted from. If given, only the text selected by those XPath will be scanned for links. See examples below.
- **restrict_css** (str or list) – a CSS selector (or list of selectors) which defines regions inside the response where links should be extracted from. Has the same behaviour as `restrict_xpaths`.
- **tags** (str or list) – a tag or a list of tags to consider when extracting links. Defaults to `('a', 'area')`.
- **attrs** (list) – an attribute or list of attributes which should be considered when looking for links to extract (only for those tags specified in the `tags` parameter). Defaults to `('href',)`
- **canonicalize** (boolean) – canonicalize each extracted url (using w3lib.url.canonicalize_url). Defaults to `False`. Note that canonicalize_url is meant for duplicate checking; it can change the URL visible at server side, so the response can be different for requests with canonicalized and raw URLs. If you're using LinkExtractor to follow links it is more robust to keep the default `canonicalize=False`.
- **unique** (boolean) – whether duplicate filtering should be applied to extracted links.
- **process_value** (callable) –

  a function which receives each value extracted from the tag and attributes scanned and can modify the value and return a new one, or return `None` to ignore the link altogether. If not given, `process_value` defaults to `lambda x: x`.

  For example, to extract links from this code:

```
<a href="javascript:goToPage('../other/page.html'); return false">Link text</a>
```

You can use the following function in `process_value`:

```
def process_value(value):
    m = re.search("javascript:goToPage\('(.*?)'", value)
    if m:
        return m.group(1)
```

- **strip** (boolean) – whether to strip whitespaces from extracted attributes. According to HTML5 standard, leading and trailing whitespaces must be stripped from `href` attributes of `<a>`, `<area>` and many other elements, `src` attribute of `<img>`, `<iframe>` elements, etc., so LinkExtractor strips space chars by default. Set `strip=False` to turn it off (e.g. if you're extracting urls from elements or attributes which allow leading/trailing whitespaces).

Previous Next