SEP	20
Title	Bulk Item Loader
Author	Steven Almeroth
Created	2012-02-24
Status	Draft

SEP-020: Bulk Item Loader

Introduction

Just as Item Loaders "provide a convenient mechanism for populating scraped Items", the Bulk Item Loader provides a convenient mechanism for populating Item Loaders.

Rationale

There are certain markup patterns that lend themselves quite nicely to automated parsing, for example the `<table>` tag outlines such a pattern for populating a database table with the embedded `<tr>` elements denoting the rows and the further embedded `<td>` elements denoting the individual fields.

One pattern that is particularly well suited for auto-populating an Item Loader is the definition list:

```
<div class="geeks">
    <dl>
        <dt> hacker
        <dd> a clever programmer

        <dt> nerd
        <dd> technically bright but socially inept person
    </dl>
</div>
```

Within the `<dl>` each `<dt>` would contain the Field *name* and the following `<dd>` would contain the Field *value*.

How it works

Without a bulk loader a programmer needs to specifically hardcode all the entries that are needed. With the bulk loader on the other hand, just a seed point is required.

Before

```
xpath = '//div[@class="geeks"]/dl/dt[contains(text(),"%s")]/following-sibling::dd[1]//text()'
gl = XPathItemLoader(response=response, item=dict())
gl.default_output_processor = Compose(TakeFirst(), lambda v: v.strip())
gl.add_xpath('hacker', xpath % 'hacker')
gl.add_xpath('nerd', xpath % 'nerd')
```

After

```
bil = BulkItemLoader(response=response)
bil.parse_dl('//div[@class="geeks"]/dl')
```

Code Proposal

This is a working code sample that covers just the basics.

```
from scrapy.contrib.loader import XPathItemLoader
from scrapy.contrib.loader.processor import MapCompose

class BulkItemLoader(XPathItemLoader):
    """ Item loader based on specified pattern recognition
    """
    default_item_class = dict
    base_xpath = '//body'
    ignore = ()

    def _get_label(self, entity):
        """ Pull the text label out of selected markup

        :param entity: Found markup
        :type entity: Selector
        """
        label = ' '.join(entity.xpath('.//text()').extract())
        label = label.encode('ascii', 'xmlcharrefreplace') if label else ''
        label = label.strip('&#160;') if '&#160;' in label else label
        label = label.strip(':') if ':' in label else label
        label = label.strip()
        return label

    def _get_entities(self, xpath):
        """ Retrieve the list of selectors for a given sub-pattern

        :param xpath: The xpath to select
        :type xpath: String
        :return: The list of selectors
        :rtype: list
        """
        return self.selector.xpath(self.base_xpath + xpath)

    def parse_dl(self, xpath=u'//dl'):
        """ Look for the specified definition list pattern and store all found
        values for the enclosed terms and descriptions.

        :param xpath: The xpath to select
        :type xpath: String
        """
        for term in self._get_entities(xpath + '/dt'):
            label = self._get_label(term)
            if label and label not in self.ignore:
                value = term.xpath('following-sibling::dd[1]//text()')
                if value:
                    self.add_value(label, value.extract(),
                        MapCompose(lambda v: v.strip()))
```

Example Spider

This spider uses the bulk loader above.

Spider code

```
from scrapy.spider import BaseSpider
from scrapy.contrib.loader.bulk import BulkItemLoader

class W3cSpider(BaseSpider):
    name = "w3c"
    allowed_domains = ["w3.org"]
    start_urls = ('http://www.w3.org/TR/html401/struct/lists.html',)

    def parse(self, response):
        el = BulkItemLoader(response=response)
        el.parse_dl('//dl[2]')
        item = el.load_item()

        from pprint import pprint
        pprint(item)
```

Log Output

```
2012-11-19 14:21:22-0600 [scrapy] INFO: Scrapy 0.17.0 started (bot: scrapy-loader)
2012-11-19 14:21:22-0600 [scrapy] DEBUG: Enabled extensions: LogStats, TelnetConsole, CloseSpider, WebService, CoreStats, SpiderState
2012-11-19 14:21:22-0600 [scrapy] DEBUG: Enabled downloader middlewares: HttpAuthMiddleware, DownloadTimeoutMiddleware, UserAgentMiddlewa
2012-11-19 14:21:22-0600 [scrapy] DEBUG: Enabled spider middlewares: HttpErrorMiddleware, OffsiteMiddleware, RefererMiddleware, UrlLength
2012-11-19 14:21:22-0600 [scrapy] DEBUG: Enabled item pipelines:
2012-11-19 14:21:22-0600 [w3c] INFO: Spider opened
2012-11-19 14:21:22-0600 [w3c] INFO: Crawled 0 pages (at 0 pages/min), scraped 0 items (at 0 items/min)
2012-11-19 14:21:22-0600 [scrapy] DEBUG: Telnet console listening on 0.0.0.0:6023
2012-11-19 14:21:22-0600 [scrapy] DEBUG: Web service listening on 0.0.0.0:6080
2012-11-19 14:21:22-0600 [w3c] DEBUG: Crawled (200) <GET http://www.w3.org/TR/html401/struct/lists.html> (referer: None) ['cached']
{'Notes': [u'The recipe may be improved by adding raisins.'],
 'The ingredients': [u'',
                     u'100 g. flour',
                     u'',
                     u'10 g. sugar',
                     u'',
                     u'1 cup water',
                     u'',
                     u'2 eggs',
                     u'',
                     u'salt, pepper',
                     u''],
 'The procedure': [u'',
                   u'Mix dry ingredients thoroughly.',
                   u'',
                   u'Pour in wet ingredients.',
                   u'',
                   u'Mix for 10 minutes.',
                   u'',
                   u'Bake for one hour at 300 degrees.',
                   u'']}
```

Notes

Other parsers can also be dropped in such as:

- `parse_table ()` with column designations for key and value,
- `parse_ul ()` with a key/value separator designation,
- `parse_ol ()` with a key/value separator designation,
- `parse ()` with a designation for key/value tags.

Actually this touches on the subject of *embedded intelligence* as it would be possible, with a little bootstrapping for what goes where, for a general parser to just go out and grab all of the above.