项目管道

在一个项目被蜘蛛抓取之后，它被发送到项目管道，该项目管道通过顺序执行的几个组件处理它。

每个项目管道组件（有时简称为“项目管道”）是一个实现简单方法的Python类。他们收到一个项目并对其执行操作，同时决定该项目是否应该继续通过管道或被丢弃并且不再处理。

项目管道的典型用途是：

- 清理HTML数据
- 验证已删除的数据（检查项目是否包含某些字段）
- 检查重复项（并删除它们）
- 将已删除的项目存储在数据库中

编写自己的项目管道

每个项管道组件都是一个必须实现以下方法的Python类：

process_item（*自我，项目，蜘蛛*）

为每个项目管道组件调用此方法。 `process_item()` 必须要么：返回带数据的dict，返回一个 `Item` （或任何后代类）对象，返回Twisted Deferred或引发 `DropItem` 异常。丢弃的项目不再由其他管道组件处理。

参数：
- **item**（ `Item` 对象或字典） - 刮取的项目
- **蜘蛛**（ `Spider` 物体） - 刮掉物品的蜘蛛

此外，他们还可以实现以下方法：

open_spider（*自我，蜘蛛*）

打开蜘蛛时会调用此方法。

参数： **蜘蛛**（ `Spider` 对象） - 被打开的蜘蛛

close_spider（*自我，蜘蛛*）

当蜘蛛关闭时调用此方法。

参数： **蜘蛛**（`Spider`物体）- 被关闭的蜘蛛

from_crawler(*cls* , *crawler*)

如果存在，则调用此类方法以从a创建管道实例`Crawler`。它必须返回管道的新实例。Crawler对象提供对所有Scrapy核心组件的访问，如设置和信号; 它是管道访问它们并将其功能挂钩到Scrapy的一种方式。

参数： crawler（`Crawler` object）- 使用此管道的爬网程序

项目管道示例

价格验证和丢弃物品没有价格

让我们看看下面的假设管道，它调整`price`那些不包含增值税（`price_excludes_vat`属性）的项目的属性，并删除那些不包含价格的项目：

```
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)
```

将项目写入JSON文件

以下管道将所有已删除的项目（来自所有蜘蛛）存储到一个`items.jl`文件中，每行包含一个以JSON格式序列化的项目：

```
import json

class JsonWriterPipeline(object):

    def open_spider(self, spider):
        self.file = open('items.jl', 'w')

    def close_spider(self, spider):
        self.file.close()

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

JsonWriterPipeline的目的只是介绍如何编写项目管道。如果您确实要将所有已删除的项目存储到JSON文件中，则应使用Feed导出。

将项目写入

在这个例子中，我们将使用pymongo将项目写入MongoDB。MongoDB地址和数据库名称在Scrapy设置中指定; MongoDB集合以item类命名。

这个例子的要点是展示如何使用 `from_crawler()` 方法以及如何正确地清理资源：

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

截取项目的截图

此示例演示如何从方法返回Deferred `process_item()` 。它使用Splash渲染项目URL的屏幕截图。Pipeline向本地运行的Splash实例发出请求。下载请求并延迟回调激活后，它会将项目保存到文件并将文件名添加到项目中。

```
import scrapy
import hashlib
from urllib.parse import quote


class ScreenshotPipeline(object):
    """Pipeline that uses Splash to render screenshot of
    every Scrapy item."""

    SPLASH_URL = "http://localhost:8050/render.png?url={}"

    def process_item(self, item, spider):
        encoded_item_url = quote(item["url"])
        screenshot_url = self.SPLASH_URL.format(encoded_item_url)
        request = scrapy.Request(screenshot_url)
        dfd = spider.crawler.engine.download(request, spider)
        dfd.addBoth(self.return_item, item)
        return dfd

    def return_item(self, response, item):
        if response.status != 200:
            # Error happened, return item.
            return item

        # Save screenshot to file, filename will be hash of url.
        url = item["url"]
        url_hash = hashlib.md5(url.encode("utf8")).hexdigest()
        filename = "{}.png".format(url_hash)
        with open(filename, "wb") as f:
            f.write(response.body)

        # Store filename in item.
        item["screenshot_filename"] = filename
        return item
```

重复过滤

一个过滤器，用于查找重复项目，并删除已处理的项目。假设我们的项目具有唯一ID，但我们的蜘蛛会返回具有相同ID的多个项目：

```
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
```

激活项目管道组件

要激活Item Pipeline组件，必须将其类添加到 `ITEM_PIPELINES` 设置中，如下例所示：

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

您在此设置中为类分配的整数值决定了它们运行的顺序：项目从较低值到较高值类进行。习惯上在0-1000范围内定义这些数字。