基于Scrapy的自定义框架，按公司需求，下载DOM源码保存到文件服务器。将Scrapy分为原生流程与开发流程。

运行环境分为线上和线下。均使用Redis作为中间站，进行对文件id的操作。

线上环境使用Redis集群，分布式爬虫，由公司大佬负责维护。

线下环境就没有集群，由我维护。

基于Scrapy框架，将爬虫分为爬取，下载，解析三个部分。

爬取过程：可以保存超链接或直接保存DOM源码

下载过程：从MongoDB中将文件id推到Redis，从Redis进行获取。再反到mongo中进行获取超链接，下载。一般为GET下载，POST请求的JSON等数据一般直接存储到文件服务器。

解析过程：从文件服务器，获取原生DOM，模拟Scrapy的Response结构，模拟Response并进行开发人员的解析。

个人版本：进行了优化，

1、在支持文件服务器的同时，也增加原生Scrapy解析流程。通过参数parse\_item参数进行控制。

2、支持动态加载，使用元类，在初始化对象之前对custom\_settings参数进行获取并设置，实现按name的动态加载。其中name为xxx-xxx样式，获取分别代表mongo数据库和collection库

当参数parse\_item为True时，使用原生scrapy流程进行存储数据，不适用文件服务器。

原生流程：start\_urls获取目标入口，通过parse函数对response主体进行解析，从中获取到数据结构体item，通过yield将数据传到管道pipeline进行后序处理。

当参数parse\_item为False时（默认），使用文件服务器。通过FileManger对获取到的主体进行构造，样式为{‘request’:’’, ‘fid’:’’, ‘size’:’’}

当为超链接是存储到requet中，其余为空，当主体为源码时，直接上传到文件服务器，获取对应的fid和size