# Scrapy架构图

Scrapy Engine: 这是引擎，负责Spiders、ItemPipeline、Downloader、Scheduler中间的通讯，信号、数据传递等等！

Scheduler(调度器): 它负责接受引擎发送过来的requests请求，并按照一定的方式进行整理排列，入队、并等待Scrapy Engine(引擎)来请求时，交给引擎。

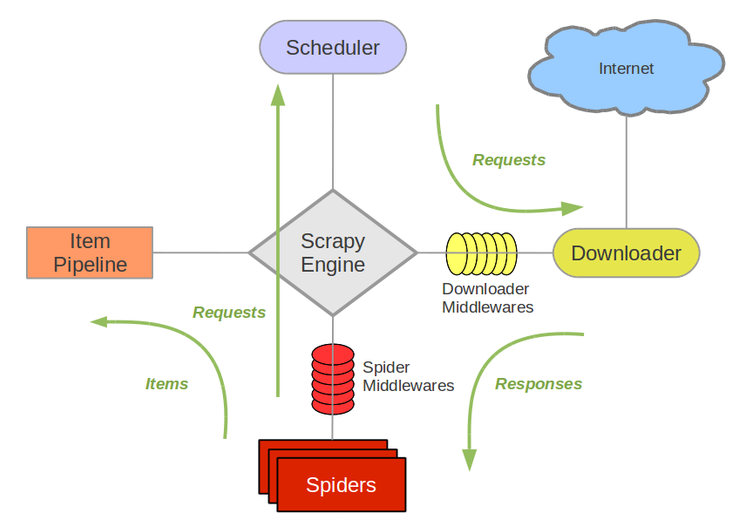
Downloader（下载器）：负责下载Scrapy Engine(引擎)发送的所有Requests请求，并将其获取到的Responses交还给Scrapy Engine(引擎)，由引擎交给Spiders来处理。

Spiders：它负责处理所有Responses,从中分析提取数据，获取Item字段需要的数据，并将需要跟进的URL提交给引擎，再次进入Scheduler(调度器)。

Item Pipeline：它负责处理Spiders中获取到的Item，并进行处理，比如去重，持久化存储（存数据库，写入文件，总之就是保存数据用的）。

Downloader Middlewares（下载中间件）：你可以当作是一个可以自定义扩展下载功能的组件。

Spider Middlewares（Spider中间件）：你可以理解为是一个可以自定扩展和操作引擎和Spiders中间‘通信‘的功能组件（比如进入Spiders的Responses;和从Spiders出去的Requests）。



# 数据在整个Scrapy的流向

程序运行的时候，

**引擎**：Hi！Spider, 你要处理哪一个网站？

**Spiders**：我要处理23wx.com

**引擎**：你把第一个需要的处理的URL给我吧。

**Spiders**：给你第一个URL是XXXXXXX.com

**引擎**：Hi！调度器，我这有request你帮我排序入队一下。

**调度器**：好的，正在处理你等一下。

**引擎**：Hi！调度器，把你处理好的request给我，

**调度器**：给你，这是我处理好的request

**引擎**：Hi！下载器，你按照下载中间件的设置帮我下载一下这个request

**下载器**：好的！给你，这是下载好的东西。（如果失败：不好意思，这个request下载失败，然后引擎告诉调度器，这个request下载失败了，你记录一下，我们待会儿再下载。）

**引擎**：Hi！Spiders，这是下载好的东西，并且已经按照Spider中间件处理过了，你处理一下（注意！这儿responses默认是交给def parse这个函数处理的）

**Spiders**：（处理完毕数据之后对于需要跟进的URL），Hi！引擎，这是我需要跟进的URL，将它的responses交给函数 def  xxxx(self, responses)处理。还有这是我获取到的Item。

**引擎**：Hi ！Item Pipeline 我这儿有个item你帮我处理一下！调度器！这是我需要的URL你帮我处理下。然后从第四步开始循环，直到获取到你需要的信息，

注意！只有当调度器中不存在任何request了，整个程序才会停止，（也就是说，对于下载失败的ＵＲＬ，Scrapy会重新下载。）

以上就是Scrapy整个流程了。