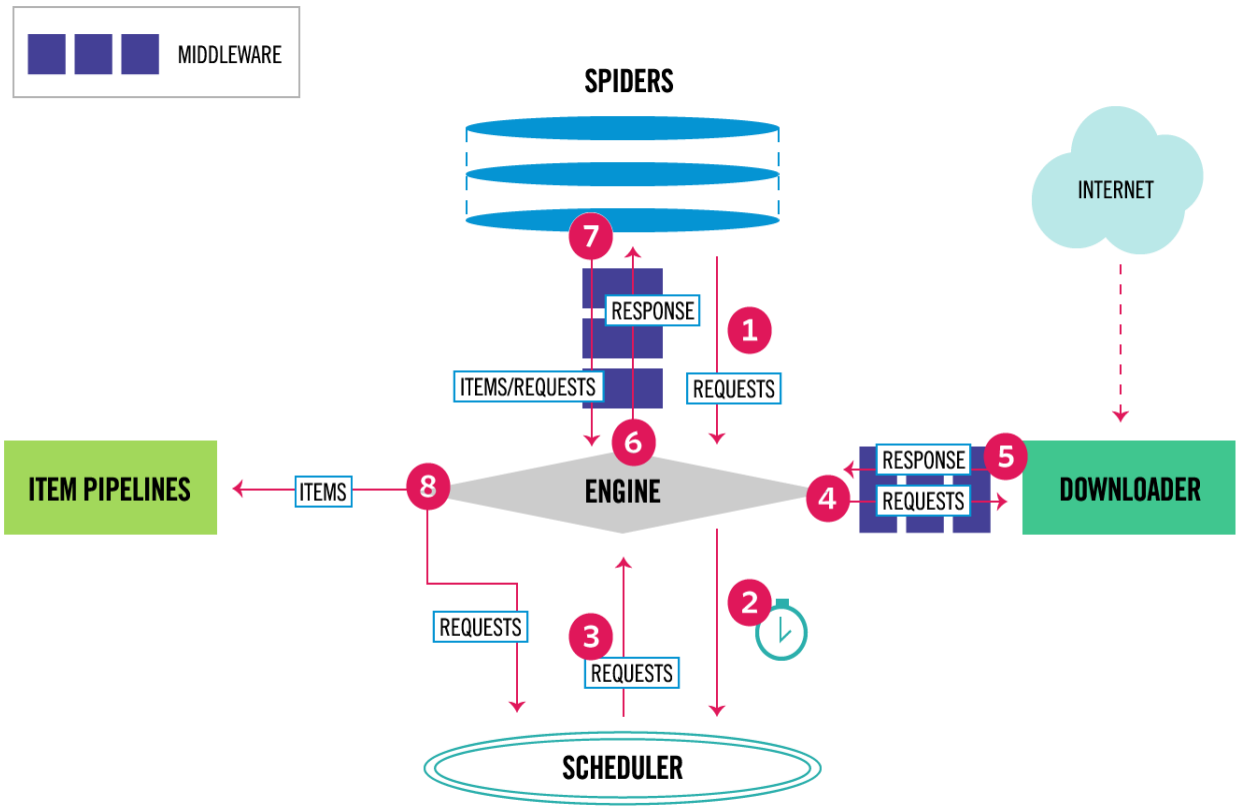
架构概述

本文档描述了Scrapy的体系结构及其组件的交互方式。

概述

下图显示了Scrapy体系结构及其组件的概述，以及系统内部发生的数据流的概述（由红色箭头显示）。下面包含组件的简要说明，并提供链接以获取有关它们的更多详细信息。数据流也在下面描述。

数据流



Scrapy中的数据流由执行引擎控制，如下所示：

1. 该引擎获得初始请求从抓取 蜘蛛。
2. 该引擎安排在请求 调度程序和要求下一个请求抓取。
3. 该计划返回下一请求的引擎。
4. 该引擎发送请求到 下载器，通过 下载器中间件（见 `process_request()` ）。

5. 页面完成下载后， Downloader会生成一个Response（带有该页面）并将其发送到Engine，并通过 Downloader Middlewares（请参阅 参考资料 `process_response()` ）。
6. 该引擎接收来自响应 下载器并将其发送到所述 蜘蛛进行处理，通过蜘蛛中间件（见 `process_spider_input()` ）。
7. 该蜘蛛处理响应并返回刮下的项目和新的要求（跟随）的 引擎，通过 蜘蛛中间件（见 `process_spider_output()` ）。
8. 该引擎发送处理的项目，以 项目管道，然后把处理的请求的调度，并要求今后可能要求抓取。
9. 该过程重复（从步骤1开始），直到调度程序不再有请求为止 。

组件

Scrapy引擎

引擎负责控制系统所有组件之间的数据流，并在发生某些操作时触发事件。有关详细信息，请参阅上面的 数据流部分。

调度程序

调度程序接收来自引擎的请求，并在引擎请求它们时将它们排入队列以便稍后（也发送到引擎）。

下载器

Downloader负责获取网页并将其提供给引擎，引擎又将它们提供给蜘蛛。

蜘蛛

蜘蛛是由Scrapy用户编写的自定义类，用于解析响应并从中提取项目（也称为已删除项目）或其他要遵循的请求。有关更多信息，请参阅蜘蛛。

项目管道

物品管道负责在物品被蜘蛛提取（或刮除）后处理物品。典型的任务包括清理，验证和持久性（如将项目存储在数据库中）。有关更多信息，请参阅项目管道。

下载中间件

下载中间件是位于Engine和Downloader之间的特定钩子，当它们从Engine传递到Downloader时处理请求，以及从Downloader传递到Engine的响应。

如果您需要执行以下操作之一，请使用Downloader中间件：

- 在将请求发送到下载程序之前处理请求（即在Scrapy将请求发送到网站之前）；
- 在将它传递给蜘蛛之前改变收到的响应;

- 发送新的请求，而不是将收到的响应传递给蜘蛛;
- 在没有抓取网页的情况下将响应传递给蜘蛛;
- 默默地放弃一些请求。

有关更多信息，请参阅下载器中间件。

蜘蛛中间件

Spider中间件是位于Engine和Spider之间的特定钩子，能够处理蜘蛛输入（响应）和输出（项目和请求）。

如果需要，请使用Spider中间件

- 蜘蛛回调的后期处理输出 - 更改/添加/删除请求或项目;
- 后处理start_requests;
- 处理蜘蛛异常;
- 根据响应内容调用errback而不是回调某些请求。

有关更多信息，请参阅Spider Middleware。

事件驱动的网络

Scrapy是用Twisted编写的，Twisted是一个流行的事件驱动的Python网络框架。因此，它使用非阻塞（也称为异步）代码实现并发。

有关异步编程和Twisted的更多信息，请参阅以下链接：

- 扭曲中的延迟介绍
- Twisted - 你好，异步编程
- 扭曲的介绍 - Krondo