

Scrapyd 常用功能与 API

上一个小节中，我们学会了如何将项目打包并部署，这一节我们来熟悉 Scrapyd 的常用功能以及学习如何通过 Scrapyd 提供的 API 对爬虫发起指令。

Scrapyd-web

[Scrapyd 官方文档](#)中有介绍，Scrapyd 为使用者提供了 Web 与 JSON 两种访问方式，其中 `web` 用于监视运行进程和爬虫日志、`json` 用于向爬虫发送操作指令。

Web 中有首页、Jobs、Logs 三个主要页面：

- 首页：服务入口，展示已部署的项目名称
- Jobs：爬虫运行状态及运行记录的主要观察页面
- Logs：爬虫运行日志文本文件资源目录

Jobs

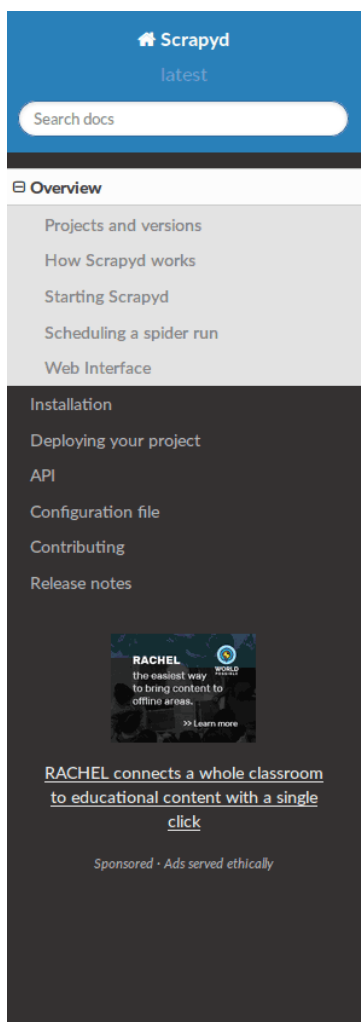
[Go back](#)

Project	Spider	Job	PID	Start	Runtime	Finish	Log
Pending							
Fabias	fabias	cc261beec7ba11e8815d00e070785d37					
Running							
Fabias	quinns	c4ac52cac7ba11e8828200e070785d37	4459	2018-10-04 17:49:23	0:00:12		Log
Fabias	fabias	caad2212c7ba11e8a84900e070785d37	4463	2018-10-04 17:49:33	0:00:02		Log
Finished							

由上图可以看到，Jobs 页面中包含了爬虫与运行记录、项目名称、爬虫名称、jobID、进程 ID、启动时间、运行时长、结束时间以及日志链接。根据以上信息，我们可以对爬虫的状态做个大致的了解。

Scrapyd-json

Scrapyd 为使用者提供了很多的 API，在文档中称为[资源](#)。



Overview

Projects and versions

Scrapy can manage multiple projects and each project can have multiple versions uploaded, but only the latest one will be used for launching new spiders.

A common (and useful) convention to use for the version name is the revision number of the version control tool you're using to track your Scrapy project code. For example: `r23`. The versions are not compared alphabetically but using a smarter algorithm (the same `distutils` uses) so `r10` compares greater to `r9`, for example.

How Scrapy works

Scrapy is an application (typically run as a daemon) that listens to requests for spiders to run and spawns a process for each one, which basically executes:

```
scrapy crawl myspider
```

Scrapy also runs multiple processes in parallel, allocating them in a fixed number of slots given by the `max_proc` and `max_proc_per_cpu` options, starting as many processes as possible to handle the load.

In addition to dispatching and managing processes, Scrapy provides a [JSON web service](#) to upload new project versions (as eggs) and schedule spiders. This feature is optional and can be disabled if you want to implement your own custom Scrapy. The components are pluggable and can be changed, if you're familiar with the [Twisted Application Framework](#) which Scrapy is implemented in.

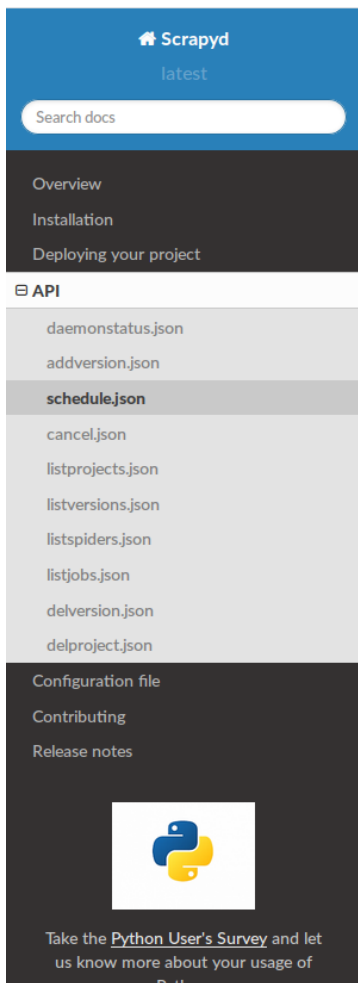
API 的作用介绍

API 的具体内容、用法和返回信息在官方文档中都有详尽的介绍，这里我做个归纳：

- `daemonstatus.json` - 用以检查 Scrapy 服务的爬虫状态及对应数量。
- `addversion.json` - 用以为项目添加版本，如果项目不存在则创建项目。
- `schedule.json` - 用以启动指定的爬虫。
- `cancel.json` - 用以取消指定的爬虫。
- `listprojects.json` - 用以查看当前已部署的项目名称。
- `listversions.json` - 用以查看指定项目的版本号。
- `listspiders.json` - 用以查看指定项目的爬虫名称。
- `listjobs.json` - 用以查看指定项目下爬虫的运行状态。
- `delversion.json` - 用以删除指定项目的指定版本，如版本不存在则将项目删除。
- `delproject.json` - 用以删除指定项目及其已上传的所有版本。

API 的使用示例

官方示例中以 `cURL` 作为请求工具，向服务的 API 发起请求



schedule.json

Schedule a spider run (also known as a job), returning the job id.

- Supported Request Methods: **POST**
- Parameters:
 - **project** (string, required) - the project name
 - **spider** (string, required) - the spider name
 - **setting** (string, optional) - a Scrapy setting to use when running the spider
 - **jobid** (string, optional) - a job id used to identify the job, overrides the default generated UUID
 - **_version** (string, optional) - the version of the project to use
 - any other parameter is passed as spider argument

Example request:

```
$ curl http://localhost:6800/schedule.json -d project=myproject -d spider=somespider
```

Example response:

```
{"status": "ok", "jobid": "6487ec79947edab326d6db28a2d86511e8247444"}
```

Example request passing a spider argument (**arg1**) and a setting (**DOWNLOAD_DELAY**):

```
$ curl http://localhost:6800/schedule.json -d project=myproject -d spider=somespider -d setting=DOWNLO
```

Note

Spiders scheduled with scrapyd should allow for an arbitrary number of keyword arguments as scrapyd sends internally generated spider arguments to the spider being scheduled

如启动指定爬虫:

```
1 $ curl http://localhost:6800/schedule.json -d project=myproject -d spider=somespider
2
```

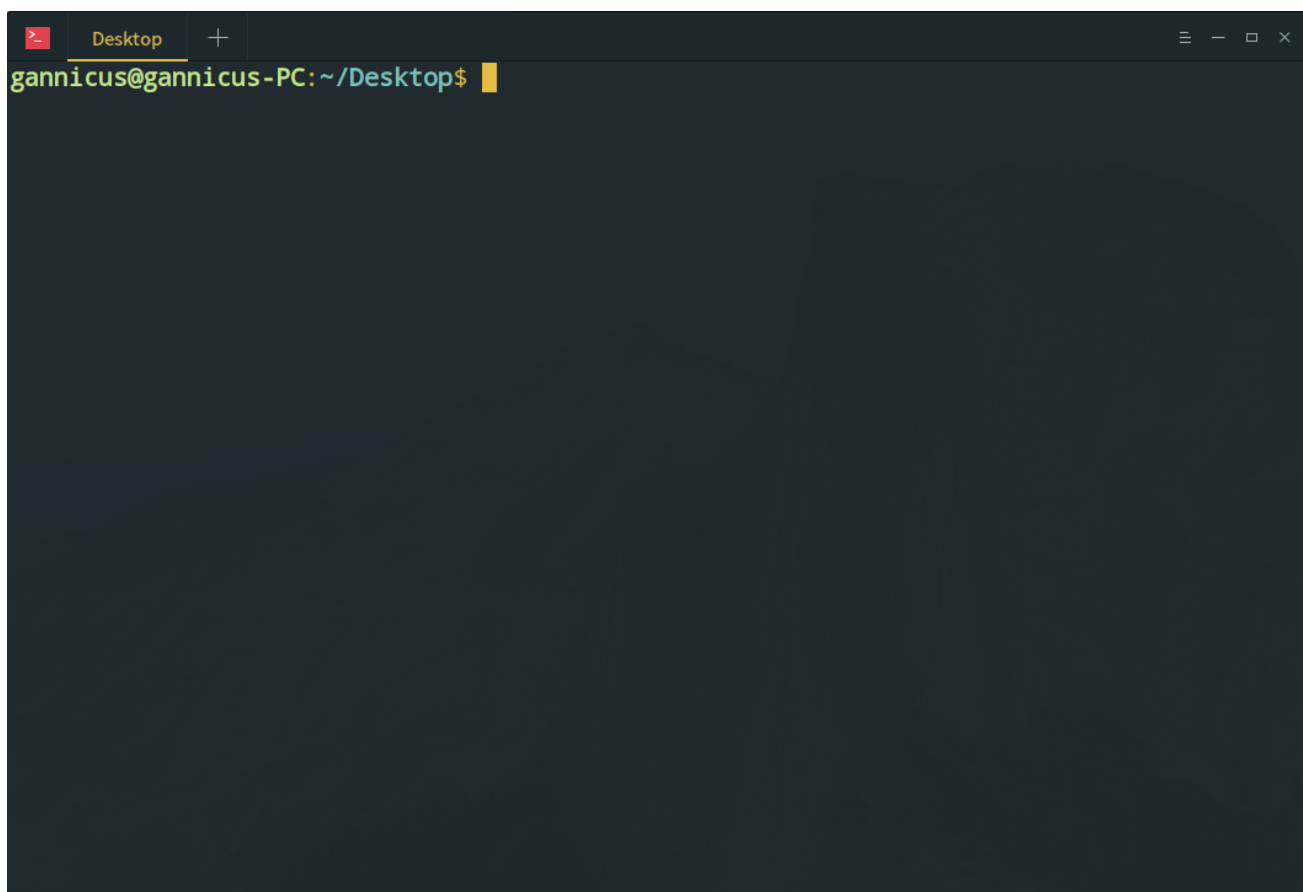
当然, 如果你想在启动的时候传递指定的参数, 也可以这么写:

```
1 $ curl http://localhost:6800/schedule.json -d project=myproject -d spider=somespider -d
2   arg1=val1
```

官方返回示例为:

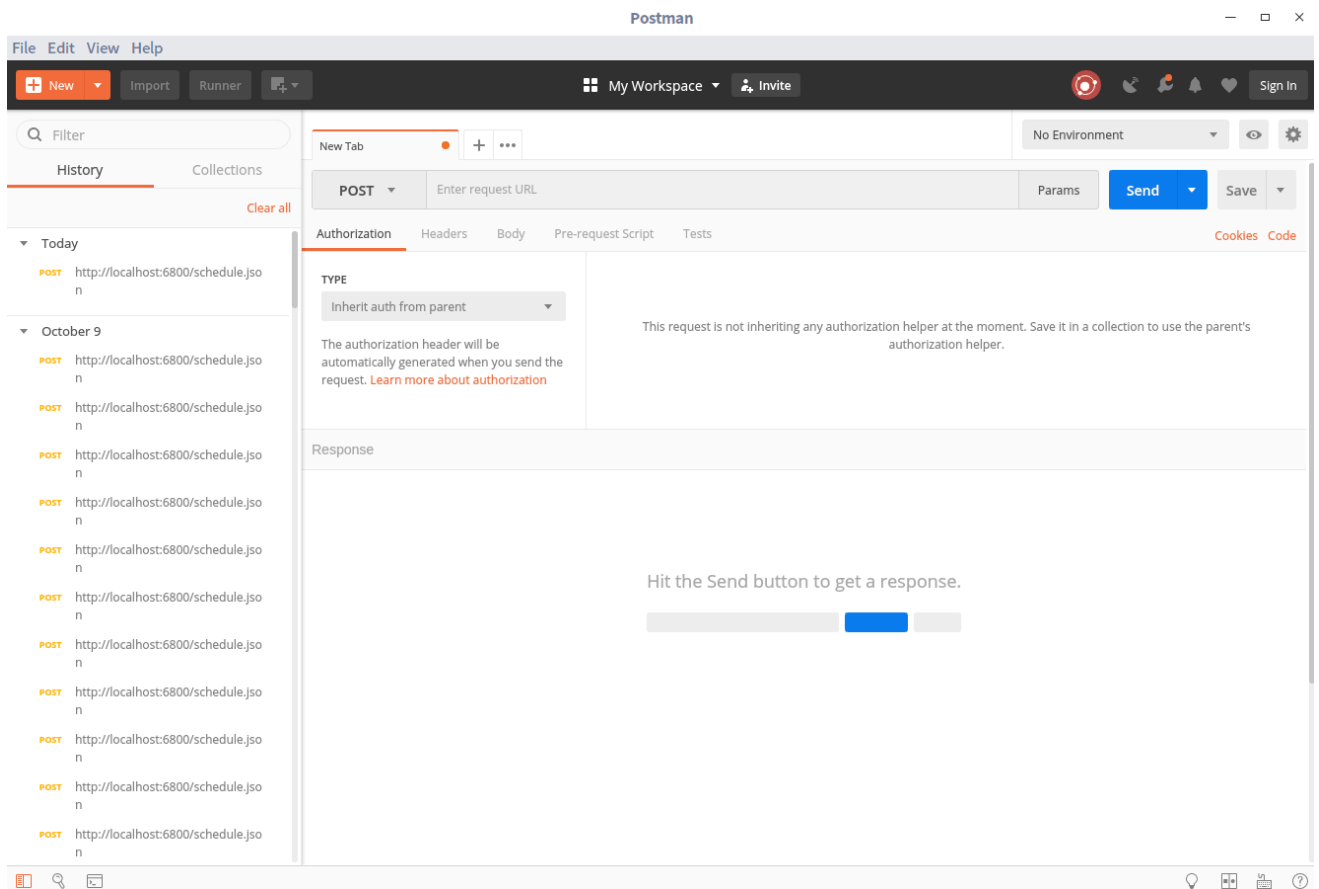
```
1 {"status": "ok", "jobid": "6487ec79947edab326d6db28a2d86511e8247444"}
2
```

cURL 动图演示:



使用 Postman 工具启动爬虫

当然，除了官方推荐的 cURL，我们还可以使用 Postman 来作为我们的请求工具，同样是启动指定的爬虫，Postman 的设置如下所示：



设置好请求地址和对应的参数后，点击 Send 发送请求，我们会得到 Scrapyd 中 Schedule 视图类提供的返回结果：

```
1 {
2   "node_name": "node-name",
3   "status": "ok",
4   "jobid": "ede04d0ac7be11e8b70d00e070785d37"
5 }
6
```

代表此次请求发送成功且能够启动对应的爬虫。

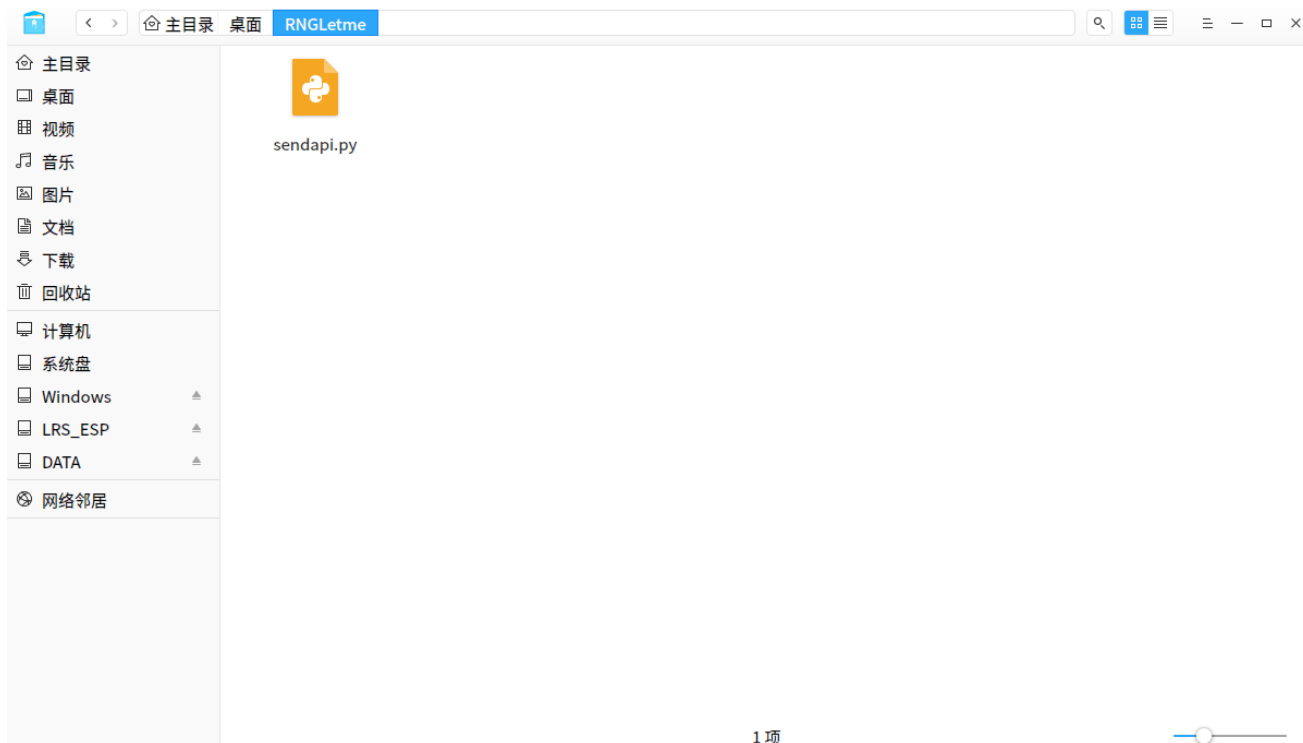
使用 Python 代码启动爬虫

既然它是通过网络请求来发送指令，那我们就可以使用 requests 来模拟请求的发起，新建 `sendapi.py` 文件并编写代码：

```
1 import requests
2
3
4 url = "http://localhost:6800/schedule.json"
5 params = {"project": "arts", "spider": "tips"}
6 resp = requests.post(url, data=params)
7 print(resp.text)
8
```

通过命令 `python sendapi.py` 运行得到返回结果：

```
1 {"node_name": "gannicus-PC", "status": "ok", "jobid": "51bdd2bad1ac11e89ed954ee75c0e204"}
2
```



Scrapyd 日志

爬虫的运行信息都会记录在日志文件当中，在 Jobs 页面，可以找到爬虫任务对应的日志，也可以通过 Logs 查看某个项目或某个爬虫的日志，日志内容如下图所示：



Scrapyd

Available projects: **Fabias**

- [Jobs](#)
- [Logs](#)
- [Documentation](#)

How to schedule a spider?

To schedule a spider you need to use the API (this web UI is only for monitoring)

Example using [curl](#):

```
curl http://localhost:6800/schedule.json -d project=default -d spider=somespider
```

For more information about the API, see the [Scrapyd documentation](#)

小结

本小节结合 Scrapyd 文档与示例代码，我们了解到 Scrapyd API 的用法，并且通过网络请求工具 Postman 发起请求，最后使用 Python 代码向 Scrapyd 服务发起请求，成功启动指定爬虫，完成了从文档到实践的跳跃。